Ceres Germanna Braga Morais

Ensino e aprendizagem de programação: estudo de caso no Ensino Superior

UMinho | 2022

Universidade do Minho
Instituto de Educação

Ceres Germanna Braga Morais

Ensino e aprendizagem de programação: estudo de caso no Ensino Superior

junho de 2022

Universidade do Minho
Instituto de Educação

Ceres Germanna Braga Morais

Ensino e aprendizagem de programação: estudo de caso no Ensino Superior

Tese de Doutoramento
Doutoramento em Ciências da Educação
Especialidade em Tecnologia Educativa

Trabalho efetuado sob a orientação do
Professor António José Meneses Osório
e do
Professor Francisco Milton Mendes Neto

junho de 2022

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho

Atribuição-NãoComercial-SemDerivações
CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/4.0/

AGRADECIMENTOS

À Deus, por conduzir-me com sabedoria e integridade durante todo doutoramento e mesmo diante das adversidades enfrentadas, me deu ânimo e paciência para que eu pudesse concluir esse percurso.

À minha mãe, Raimunda Braga, que deixou tudo para viver por mim, para mim e para a minha família. Ao meu pai, José Avelino, por todo amor e mesmo longe se faz sempre presente em minha vida.

Ao meu esposo, Joamar Neto, meu amigo, companheiro e maior incentivador para que eu fizesse este doutoramento e aos meus amados filhos, Emanuela, Mateus e Heitor, que mesmo sem entenderem o percurso, foram fundamentais que eu chegasse até aqui. A vocês, meus filhos, dedico este título de Doutora em Ciências da Educação.

Aos meus familiares por todo amor e torcida para que eu conseguisse ter êxito nesse percurso. À Dona Evaldite, Fafá e Jaianny, em nome das quais agradeço a todos da família de Joamar, pelo apoio e compreensão das nossas ausências nos últimos anos.

À Luisa, Joaquim e Hannah, por todo apoio em nossa vinda e estada em Portugal.

Ao meu orientador, Professor Osório, por todos os ensinamentos, resiliência, paciência e atenção a mim dedicados. Sua competência e rigor na orientação foram fundamentais para que eu conseguisse me tornar doutora. Ao Professor Milton Mendes, meu segundo orientador, pela atenção, conhecimentos partilhados e disponibilidade nessa jornada.

Aos professores que compuseram o júri e partilharam conhecimentos e contribuições enriquecedoras para que eu pudesse melhorar.

À Professora Altina Ramos, ao Professor Bento Silva e ao Professor Leandro Almeida, em nome dos quais agradeço a todos os professores do Instituto de Educação. À Patrícia Capelo e Natália Rodrigues, secretárias do Doutoramento em Ciências da Educação, que tanto me ajudaram durante este percurso sem medir esforços. Muito obrigada!

À Professora Cicilia Raquel e ao Professor Pedro Fernandes, em nome dos quais agradeço a UERN, essa Universidade Pública, Gratuita, de Qualidade e Socialmente referenciada, da qual tenho imenso orgulho de ser fruto, produzir frutos e colher frutos. Ao Professor Maximiliano Lopes, em nome de quem agradeço a todos os que fazem o Departamento de Informática. Sou eternamente grata pelo apoio incondicional de todos durante meu doutoramento. Meu muito obrigada!

Ao estimado Almir de Castro, do setor de Capacitação da PROPEG, pela dedicação, paciência e cuidado que teve comigo durante todo meu percurso no doutoramento.

À Professora Regina Marques e ao Jesaías Silva pelo grande apoio no desenvolvimento do projeto submetido ao Comitê de Ética em Pesquisa. Aos alunos e professores pela disponibilidade voluntária na participação das recolhas de dados, fundamentais para a tese.

À Eveline e Eliana, que me deram todo o apoio, com dicas e recomendações de como ir para Portugal. À Adriana Aleixo, amiga que o doutoramento me deu e que entrou, permaneceu e finalizou esta jornada junto comigo, o que para mim fez toda a diferença! À Gabryella, amiga que a orientação do Professor Osório me deu, que me ajudou na RSL e nas leituras dos meus capítulos. Vocês duas ficarão para sempre em meu coração! A João Paulo, amigo que a UERN me deu, por todo apoio nesse percurso, pelas orientações e por me ajudar também na nossa ida para Portugal.

Ao professor Leonel Morgado, em nome de quem agradeço a todos que fazem parte do projeto SCReLProg, do qual tive imenso prazer e orgulho de ser bolseira de investigação durante o período do doutoramento.

À Portugal e à Coimbra pela acolhida e por nos fazerem tão felizes enquanto lá vivemos.

Agradeço à Universidade do Estado do Rio Grande do Norte, pelo apoio financeiro concedido para a realização do Doutoramento em Ciências da Educação, na Universidade do Minho e pela concessão da Licença para Capacitação no Exterior.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO: ESTUDO DE CASO NO ENSINO SUPERIOR

Resumo

O processo de ensino e aprendizagem de programação é uma tarefa complexa que coloca desafios importantes a docentes e discentes. Ensinar a programar exige do professor uma forte demanda de interação a fim de atender, acompanhar, mediar e avaliar individualmente os alunos e suas atividades, e escolher os caminhos mais adequados para manter a motivação, o envolvimento e o bom desempenho destes. Aprender a programar é um processo complexo pois envolve diversas singularidades do domínio da programação, exige dos estudantes a prática constante e conhecimentos e habilidades específicos tais como interpretação e resolução de problemas. Esses aspectos podem dificultar o processo, causando desmotivação e frustração de docentes e discentes, bem como a desistência e reprovação nas Unidades Curriculares. Diante do exposto, nessa pesquisa tivemos como objetivo compreender 'como ocorre o processo de ensino e aprendizagem de programação no Ensino Superior', a fim de inventariar metodologias e ferramentas, caracterizar os fatores que influenciam na aprendizagem, explicitar os conhecimentos e habilidades necessários para aprender a programar e identificar as dificuldades enfrentadas por professores e alunos e o que estas dificuldades podem acarretar. A investigação é de natureza mista e teve como abordagem metodológica a realização de um estudo de caso único. Para tanto, selecionamos documentos e realizamos a aplicação de questionários e entrevistas a estudantes e professores de um curso superior na área de Informática no contexto brasileiro. Após a recolha de dados procedemos à análise e interpretação de conteúdo. Em seguida efetuamos a triangulação das fontes de evidências, categorizados à luz da fundamentação teórica e de acordo com os parâmetros 'O que', 'Quem', 'Quando', 'Onde', 'Porque', 'Como' e 'Quanto', baseados no *framework* 5W2H, a fim de compreender os sentidos dos dados e como eles respondem a questão central da tese. O estudo identifica contributos relacionados aos objetivos elencados, de forma que os resultados obtidos podem servir de suporte para a construção de novos conhecimentos para o desenvolvimento de estratégias que possam contribuir para o processo de ensino e aprendizagem de programação.

Palavras-chave: Desafios; Ensino de Informática ; Ensino e aprendizagem de programação; Ensino Superior; Estudo de Caso.

TEACHING AND LEARNING PROGRAMMING: A CASE STUDY IN HIGHER EDUCATION

Abstract

The process of teaching and learning programming is a complex task that poses important challenges to teachers and students. Teaching programming requires a strong demand for interaction from the teacher to attend, monitor, mediate, and evaluate individually the students and their activities, and to choose the most appropriate ways to maintain their motivation, involvement, and good performance. Learning to program is a complex process because it involves several singularities of the programming domain, demands from the students constant practice and specific knowledge and skills such as interpretation and problem solving. These aspects can hinder the process, causing demotivation and frustration for teachers and students, as well as dropout and failure in the Curricular Units. Considering the above, in this research we aimed to understand 'how the process of teaching and learning programming occurs in Higher Education', to inventory methodologies and tools, characterize the factors that influence learning, explain the knowledge and skills required to learn to program, and identify the difficulties faced by teachers and students and what these difficulties may entail. The research is mixed in nature and its methodological approach was a single case study. To do so, we selected documents and carried out the application of questionnaires and interviews to students and teachers of a higher education course in the area of Computer Science in the Brazilian context. After data collection we proceeded to content analysis and interpretation. Then we carried out the triangulation of the sources of evidence, categorized in the light of the theoretical foundation and according to the parameters 'What', 'Who', When', 'Where', 'Why', 'How' and 'How Much', based on the 5W2H framework, in order to understand the meanings of the data and how they answer the central question of the thesis. The study identifies contributions related to the listed objectives, so that the results obtained can serve as support for the construction of new knowledge for the development of strategies that can contribute to the process of teaching and learning programming.

Keywords: Case Study; Challenges; Computer Science Teaching; Higher Education; Teaching and Learning of Programming.

Índice

Lista de Abreviaturas e Siglas

ACM	Association for Computing Machinery
CD	Coding Dojo
CdP	Comunidade de Prática
CEP	Comitê de Ética em Pesquisa
CES	Câmara de Educação Superior
CNE	Conselho Nacional de Educação
DCN	Diretrizes Curriculares Nacionais
ENEM	Exame Nacional do Ensino Médio
FT	Fundamentação Teórica
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronic Engineers
IF	Instituto Federal
IFRN	Instituto Federal do Rio Grande do Norte
LMS	Learning Management System
MOOC	Massive Online Open Courses
PET	Programa de Educação Tutorial
PGCC	Programas Gerais dos Componentes Curriculares
PLA	Programming Learning Assesment
PLEM	Programming Learning Experience Model
POO	Programação Orientada a Objetos
PPC	Plano Pedagógico do Curso
QP	Questão de Partida
RSL	Revisão Sistemática de Literatura
SISU	Sistema de Seleção Unificada
TI	Tecnologia da Informação
UC	Unidade Curricular
UERN	Universidade do Estado do Rio Grande do Norte
UMinho	Universidade do Minho
XP	eXtreme Programming

Lista de Figuras

Lista de Tabelas

1. Introdução

A nossa pesquisa 'Ensino e aprendizagem de programação: estudo de caso no Ensino Superior' está inserida no âmbito de Ciências da Educação, do Instituto de Educação da Universidade do Minho, na especialidade Tecnologia Educativa, e tem por finalidade compreendermos como ocorre o processo de ensino e aprendizagem de programação, à luz da literatura e das percepções, vozes e palavras das nossas fontes de evidências.

O ensino e aprendizagem de programação estão inseridos nas diretrizes curriculares dos cursos de Computação e áreas correlatas (Rum & Ismail, 2017). No entanto, ensinar a programar exige do professor formação e estratégias adequadas para acompanhar, orientar, avaliar e apoiar seus alunos, geralmente com perfis e níveis de conhecimentos na área diferentes, que podem acarretar dificuldades na tarefa do professor. Para os alunos, aprender programação também é desafiador, uma vez que cada um possui competências e habilidades diferentes, ou a falta de algumas destas, fatores que o apoiam ou afetam sua aprendizagem, e ainda dificuldades individuais que podem gerar consequências para o seu percurso acadêmico.

Para tentar diminuir ou mitigar os desafios vivenciados ao longo do processo, professores precisam identificar as dificuldades para que possam dar o suporte devido a seus alunos (Kawaguchi et al., 2019), de maneira a garantir um melhor desempenho de cada um de forma individual, e da turma como um todo, e os alunos precisam reconhecer suas dificuldades, bem como adotar estratégias que os auxiliem a adquirir ou melhorara as competências e habilidades necessárias para aprender a programar.

Fazer uso de métodos e tecnologias que busquem maior interação entre aluno/professor/disciplina, e em que os alunos percebam sua evolução no contexto da sala de aula, podem ser diferenciais na aprendizagem, uma vez que quanto maior é o envolvimento do aluno na manipulação criativa, na pesquisa, na interação com o próprio conhecimento, na descoberta de novas formas de expressão de saberes, maior é a eficácia didática deste processo (Bento et al., 2016). Além disso, a "nova dinâmica social e tecnológica sugere uma transformação dos sistemas educativos e de formação em função dos novos desafios e das exigências da sociedade da informação" (Meirinhos & Osório, 2011, p. 39).

De acordo com Vygotsky & Cole (1978), a aprendizagem se dá por um longo processo de apropriação e transformação de conhecimentos que ocorre na atividade mediada, na relação com os outros. Freire (1987) enfatiza a importância desta na educação, pressupondo o diálogo, de forma que a educação seja humanista e problematizadora.

Muitas vezes, por não haver uma metodologia de ensino que garanta a interação professor-aluno, bem como o acompanhamento, nivelamento e avaliação mais concreta de cada aluno, os alunos acabam tendo dificuldade em adquirir os conceitos iniciais de programação, fundamentais para os passos posteriores, uma vez que a aprendizagem de programação é considerada iterativa e recursiva e, consequentemente, podem evadir ou reprovar nestas disciplinas (Prietch & Pazeto, 2010).

As dificuldades de aprender a programar se dão pelo fato de exigir não só o aprendizado de uma tecnologia, mas de um conjunto de habilidades, tais como o raciocínio lógico, a competência para resolver problemas e a capacidade de abstrair soluções fazendo uso de uma representação formal (A. J. Gomes, 2010). E para isso, é necessário que o aluno adote estratégias de estudo adequadas, que proporcionem a prática constante, e demonstrem compromisso e dedicação com o processo de aprendizagem.

O grande desafio do professor é conseguir atender todos os alunos de maneira geral e individual, de forma que todos sejam contemplados e adquiram 'igualmente' o conhecimento necessário para avançar na disciplina. No entanto, por se tratar de um contexto em que os envolvidos possuem diferentes níveis de conhecimento e interesses, bem como a existência de turmas numerosas, as metodologias comumente utilizadas em sala de aula acabam por não serem satisfatórias e não atenderem às expectativas dos alunos e docentes.

A fim de verificarmos a fundamentação teórica sobre o processo de ensino e aprendizagem de programação no Ensino Superior, realizamos pesquisas em diferentes bases de dados (Web of Science, IEEE, Scopus e RCAAP). Esse procedimento nos permitiu manipular e consultar grandes volumes de dados e nos ajudou a perceber que a grande maioria dos estudos abordavam as dificuldades vivenciadas pelos alunos ao longo do processo de aprendizagem de programação, o que se justifica pelo fato da complexidade inerente a esse processo

1.1. Motivações para a Pesquisa

A realização desta pesquisa pautou-se especialmente em três diferentes motivações:

A primeira, enquanto egressa do curso de Ciência Computação, em que vivenciei algumas dificuldades durante o processo de aprendizagem de programação, mas que devido a um conjunto de aspectos, consegui ultrapassá-las e obter êxito no meu percurso acadêmico, pessoal e profissional.

A segunda, enquanto professora de programação no Ensino Superior, em que vejo que meus alunos continuam passando por dificuldades e e desafios ao longo do percurso acadêmico, dificuldades estas causadas por diferentes fatores e que culminam em diferentes consequências. Percebo que não há receita de bolo para ensinar e aprender programação, e que isso depende das mais diversas

variáveis, condicionais e estruturas. Não é fácil ser professora de programação, assim como foi fácil ser aluna de programação. As angústias e inquietações que tenho em 'como ensinar os meus alunos a aprenderem a programar', me levaram, portanto para a minha terceira motivação.

A terceira motivação é enquanto pesquisadora, na tentativa de buscar respostas para saber como ocorre o processo de ensino e aprendizagem de programação, que aspectos são inerentes a este processo, de forma a explorá-los e deixá-los evidentes à comunidade acadêmica.

Essa tese tenta responder minha questão de partida, meus objetivos, mas principalmente trazer para alunos, professores e pesquisadores, assim como eu, a compreensão sobre 'como ocorre o processo de ensino e aprendizagem de programação no Ensino Superior'.

1.2. Questão de Partida e Objetivos

Diante do contexto e da minha relação com a aprendizagem de programação no Ensino Superior, enquanto aluna, e com o ensino de programação, enquanto docente, pude identificar a minha problemática de investigação para esta tese de doutoramento em Ciências da Educação, tendo como questão de partida: 'Como ocorre o processo de ensino e aprendizagem de programação no Ensino Superior'. Na busca de compreender e entender a questão proposta, desenvolvemos os seguintes objetivos de pesquisa:

- Inventariar metodologias e ferramentas usadas para o ensino e aprendizagem de programação;
- Caracterizar os fatores que podem influenciar para o aluno aprender a programar;
- Explicitar os conhecimentos e habilidades necessários para aprender a programar;
- Identificar as dificuldades enfrentadas por professores e alunos durante o processo de ensino e aprendizagem, e o que estas dificuldades acarretam.

Os objetivos definidos para essa investigação além de nos ajudar a dar respostas à questão de partida, nos orientam na definição do *design* da pesquisa, pois envolvem a justificativa e caracterização do uso das técnicas e instrumentos de recolha e análise dos dados para apresentação das informações posteriormente. De acordo com Oliveira (2020) "a metodologia não é escolhida ao acaso, pois ela está intrínseca à questão de investigação e presente, indiretamente, nos objetivos que pretendemos alcançar, ou seja, a opção metodológica não é uma escolha do pesquisador, mas determinada pelos seus objetivos" (p. 170).

1.3. Método da Investigação

A modalidade de uma pesquisa científica deve estar alinhada com a questão de partida e os objetivos, para garantir um melhor êxito do trabalho e a possibilidade de sua aplicação em outros

contextos. A abordagem mista que enquadra a nossa pesquisa possibilitou um melhor direcionamento para as atividades realizadas ao longo do desenvolvimento desta tese, além de impor credibilidade e coerência a recolha e análise dos dados (Richardson, 2017).

Entendemos que a pesquisa realizada sobre o processo de ensino e aprendizagem de programação tendo como caso um curso de Computação proporciona uma compreensão desse processo que é complexo e que abrange professores e alunos dos mais diversos cursos de Ensino Superior da área de Informática.

O estudo de caso foi considerado uma estratégia apropriada para nossa pesquisa e, para tanto, nos pautamos em Yin (2015), que define estudo de caso como uma investigação empírica que explora um fenômeno contemporâneo dentro do seu contexto da vida real, principalmente quando os limites existentes entre o fenômeno e o contexto não estão bem definidos.

Para a nossa pesquisa, desenvolvemos um estudo de caso único, tendo o propósito de investigar um caso de ensino e aprendizagem de programação no Ensino Superior. Para a escolha do caso, consideramos um curso superior cuja matriz curricular possuísse Unidades Curriculares de programação em diferentes níveis de competências e que as tenham de forma contínua ao longo do percurso acadêmico. A razão pela escolha do Curso deu-se por estar inserido no contexto brasileiro e pela possibilidade de realizar um estudo de campo em um curso que temos facilidade de acesso às fontes de evidência.

O plano de pesquisa desta investigação é a estratégia de triangulação de dados que, como definida por Stake (2005), é "considerada um processo de usar múltiplas percepções para esclarecer o significado, verificando a repetibilidade de uma observação ou interpretação" (p. 454, tradução nossa).

Para o nosso estudo, adotamos inicialmente uma perspectiva teórica que norteou a investigação para então iniciarmos o uso das diferentes abordagens que conduziram a triangulação dos resultados. A estratégia adotada foi necessária para que o rigor fosse garantido, de forma que pudéssemos encontrar resultados que contemplassem nossos objetivos e nossa questão de partida, especialmente diante da complexidade do contexto e do objeto de pesquisa.

Para o desenvolvimento do estudo de caso dessa tese, tivemos que optar por instrumentos de recolha que nos permitisse coletar os dados de forma remota, face ao contexto pandêmico que vivenciamos desde meados de março de 2020, mas sem perder a acurácia que é importante para esta modalidade de estudo.

Para cada uma das fontes de dados (questionários de alunos, entrevistas para alunos, entrevistas para professores), realizamos a categorização das unidades de registro com base nas

categorias definidas na fundamentação teórica, originando um arcabouço de dados organizadas a partir dos parâmetros 'O que', 'Quem', 'Onde', 'Quando', 'Porque', 'Como' e 'Quanto', do *framework* 5W2H, sobre o qual falamos mais adiante, que, ao serem sobrepostos, nos permitiram realizar a triangulação desses dados, para posterior interpretação e apresentação dos resultados.

1.4. Estrutura da Tese

A nossa pesquisa está estruturada em cinco capítulos: 'Introdução', 'Ensino e aprendizagem de programação no Ensino Superior', 'Enquadramento metodológico', 'Estudo de caso' e 'Considerações finais'.

No capítulo 'Introdução' apresentamos a motivação e justificativa para pesquisar sobre o processo de ensino e aprendizagem de programação. Apontamos a necessidade de se estudar a temática e como formulamos a questão de partida e os objetivos da pesquisa. Apresentamos ainda o método da investigação que conduziu toda a realização desta tese, desde a coleta de dados até sua análise, interpretação e apresentação dos resultados.

No capítulo 'Ensino e aprendizagem de programação no Ensino Superior' apresentamos a nossa fundamentação teórica que nos norteia nessa tese. Com o trabalho realizado, pudemos evidenciar os conceitos de programação e o contexto em que o ensino e aprendizagem ocorre no âmbito do Ensino Superior. Ainda neste capítulo apresentamos à luz da literatura os conhecimentos, habilidades e fatores que auxiliam o processo de ensino e aprendizagem de programação, as dificuldades enfrentadas ao longo do processo e as metodologias e ferramentas existentes e que podem ser adotadas por professores e alunos.

No capítulo 'Enquadramento metodológico' apresentamos a abordagem metodológica desta tese, pautada na abordagem mista e na realização de um estudo de caso, considerado uma estratégia apropriada para nossa pesquisa e que, para tanto, nos pautamos. Abordamos ainda o procedimento metodológico que define todos os passos desenvolvidos no estudo, a saber: elaboração do projeto para o Comitê de Ética em Pesquisa, apresentação do caso estudado, a forma que foi realizada a recolha dos dados, o registro e tratamento dos dados, para sua posterior análise e interpretação.

No capítulo 'Estudo de caso' apresentamos a caracterização do caso, sua origem, objetivos e estrutura do curso. Nesse mesmo capítulo apresentamos a triangulação dos dados, à luz das percepções, vozes e palavras daqueles que participaram dessa pesquisa, organizadas de acordo com os parâmetros 'O que', 'Quem', 'Onde', 'Quando', 'Porque', 'Como' e 'Quanto', do *framework* 5W2H, e que nos ajuda a responder à questão de partida e alcançar os objetivos traçados.

No capítulo ‘Considerações finais’, apresentamos nossas conclusões e como os objetivos traçados foram contemplados, as implicações que nosso estudo traz para o ensino e aprendizagem de programação, as limitações do estudo e as sugestões para trabalhos futuros.

2. Ensino e Aprendizagem de Programação no Ensino Superior

Este capítulo apresenta a fundamentação teórica dessa tese, tendo como foco o processo de ensino e aprendizagem de programação no Ensino Superior. Organizamos o capítulo em duas partes: 'definição e contextualização' e 'caracterização'. Na seção de definição e contextualização são apresentados os conceitos do que é programação, sua importância para a formação dos alunos e como esta está inserida nos cursos de Ensino Superior. Esse estudo é importante para caracterizar o público-alvo da pesquisa e permitir que o leitor conheça a abrangência da programação no Ensino Superior, muitas vezes identificada apenas como da área de Informática.

Na seção de caracterização, apresentamos como ocorre o processo de ensino e aprendizagem de programação no Ensino Superior. Para o desenvolvimento desse tópico, realizamos uma pesquisa mais alargada no âmbito da temática, que complementa uma Revisão Sistemática de Literatura desenvolvida no âmbito do doutoramento (Morais et al., 2020, 2019), em que permite-se identificar as dificuldades enfrentadas pelos alunos. Com a pesquisa, identificamos as dificuldades enfrentadas no processo de ensino e aprendizagem de programação e as consequências dessas dificuldades, os conhecimentos, habilidades e competências necessárias para a aprendizagem de programação, os fatores que auxiliam ou afetam o processo e as metodologias e ferramentas que podem ser usadas como apoio.

Com essa pesquisa, verificamos a complexidade do processo de ensino e aprendizagem, sendo este alvo de estudo de diferentes pesquisadores, mas que ainda tem muito a ser explorado. A articulação das duas seções nos dirige para os demais capítulos da tese, nos norteando para a obtenção dos resultados.

2.1. Definição e Contextualização

Esta seção tem por objetivo apresentar conceitos acerca da programação e contextualizá-la ao Ensino Superior. Para tanto, está dividida em quatro partes: 'conceitos', 'importância', 'cursos' e 'designação das Unidades Curriculares'.

2.1.1. Conceitos

Nesta seção são apresentadas as definições e a contextualização da programação no Ensino Superior. Iniciamos a definição recorrendo ao trabalho doutoral de Tavares (2018), que define o processo de aprendizagem de programação como uma tarefa considerada complexa e que coloca desafios importantes aos docentes de Informática. Nos cursos superiores, especialmente os da área de Informática, as disciplinas de programação precisam ser entendidas como as que formarão um aluno mais preparado para os desafios restantes do curso, uma vez que são basilares para o entendimento

dos conceitos e fundamentações necessárias para outras disciplinas mais avançadas (Lahtinen et al., 2005, como citado em Medeiros, 2019). A disciplina de programação, segundo Rocha et al. (2010), é "requisito fundamental nos cursos de computação e um dos grandes motivos de evasão escolar nos cursos de computação" (p. 3). A arte da programação inclui o conhecimento de ferramentas e linguagens de programação, habilidades de resolução de problemas e estratégias eficazes para o desenvolvimento e implementação de programas (Using et al., 2010).

De acordo com Jenkins (2001), programação é um assunto que está no cerne da computação e a compreensão de como os programas são escritos é uma parte fundamental do desenvolvimento de qualquer estudante de computação. No Ensino Superior, a programação é tradicionalmente ensinada por meio de uma série de aulas teóricas, as quais cobrem os conceitos básicos de programação, tais como variáveis, *loops*, condicionais, e assim por diante. Geralmente esses conceitos são ilustrados usando a sintaxe de uma linguagem específica e mais detalhes do idioma são adicionados gradualmente à medida que os alunos se tornam mais proficientes (Jenkins, 2001).

De maneira geral, a programação é vista como uma forma de resolver problemas, indo desde a compreensão do problema à sua resolução (Ambrósio et al., 2011). Alguns autores defendem que a programação pode ser decomposta em fases ou etapas, haja vista a sua complexidade.

Para Farrel (2010), o desenvolvimento de um programa pode ser compreendido em quatro partes: i) entender o problema; ii) especificar ou planejar a organização lógica; iii) codificar o programa; e iv) executar e testar a solução.

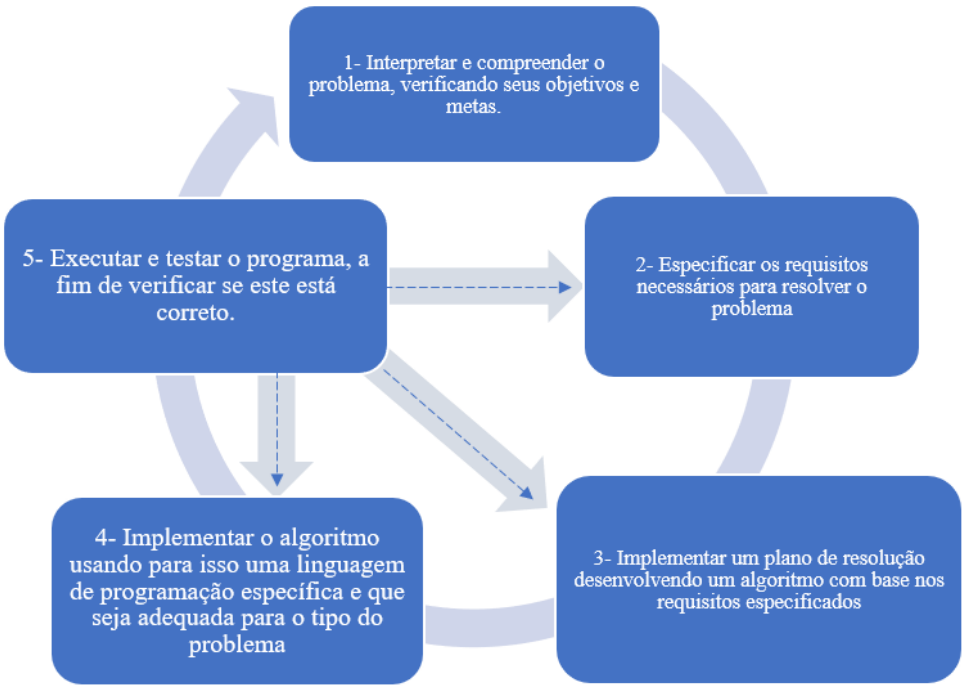
Ambrósio et al. (2011) também apresentam quatro fases para a resolução de problemas: i) entender o problema; ii) definir ou planejar uma solução para o problema; iii) implementar um plano de resolução; e iv) e verificar se está certo ou se é adequado o resultado conquistado.

Tavares (2018) organiza a tarefa de programação em cinco etapas: i) analisar o problema a ser resolvido; ii) criar uma especificação rigorosa e não ambígua do problema com os requisitos e restrições; iii) desenvolver um algoritmo que satisfaça os requisitos que foram especificados; iv) codificar o algoritmo criado usando uma linguagem de programação; e v) executar e testar o programa desenvolvido.

Para Lemos (2004), existem também cinco passos que devem ser realizados para a resolução de problemas: i) identificar os objetivos ou metas do problema; ii) selecionar planos de programação que alcancem seus objetivos; iii) selecionar o conjunto de ações que compõe cada plano; iv) decompor as ações em sub-planos (ações primitivas); v) codificar cada sub-plano em linguagem de programação.

Com base nos autores citados acima, desenvolvemos um esquema que permite representar o processo de resolução de problemas, por meio da programação, mostrado na Figura 1.

Figura 1 - *Etapas para resolução de problemas com a programação*



Nota: Autoria própria.

As etapas que envolvem a resolução de problemas com o desenvolvimento de programas são complementares e seguem uma sequência lógica, e são complexas por envolver sub tarefas ligadas a diferentes domínios de conhecimento e de processos cognitivos (Pea & Kurland, 1983). Entretanto, não necessariamente termina de forma linear, uma vez que o programador pode perceber, ao longo do desenvolvimento do software que as falhas ou inconsistências podem ocorrer em quaisquer uma das etapas anteriores. Dessa forma, além de ser um processo iterativo, a resolução de problemas com a programação permite que quem o esteja desenvolvendo volte às etapas anteriores sem precisar seguir a ordem. Por exemplo, ao executar e testar o programa, o que ocorre na etapa 5, o programador pode perceber que o plano de resolução apresenta falhas e, com isso, retornar à etapa 3, depois prosseguir com a 4 até chegar à 5 novamente.

Em nosso trabalho, percebemos que aprender a programar não é especificamente aprender uma linguagem de programação, mas aplicar diferentes conhecimentos, seguindo com rigor etapas necessárias para se obter a solução de um problema. De acordo com Medeiros (2019), programar "nada mais é que dar instruções para que a máquina realize alguma atividade que lhe foi indicada a

ser executada" (p. 28), e para isso, é necessário conhecer e escolher a linguagem de programação adequada para a solução. Para Tavares (2018), "a Programação tem por fim resolver problemas por computador, ou seja, visa preparar o computador para transformar dados em resultados que correspondam à solução do problema que se quer resolver" (p. 1), e complementa que programar é "traçar estratégias, a fim de resolver problemas com recurso ao computador, independentemente da linguagem utilizada" (p. 9).

Aguilar (2008) aponta que um programa pode ser entendido como um conjunto de instruções ou ordens dadas à máquina, com o fim de executar determinada tarefa. Já para A. J. Gomes et al. (2008),

> a programação é muito mais do que a escrita de um conjunto de linhas de código numa dada linguagem, é uma arte e uma ciência. Arte porque existem muitas maneiras diferentes de codificar instruções, com alguma criatividade. É também uma ciência, porque é constituída por um conjunto de regras orientadoras, porque é necessário o uso de lógica e porque existem alguns métodos rigorosos de programação que asseguram a eficiência, economia e utilidade dos programas gerados. (p. 93)

Isto aponta-nos que programar vai muito além de aprender uma linguagem de programação e suas regras de sintaxe e semântica. A programação independe da linguagem e ferramentas escolhidas para tanto, mas necessita de habilidades e conhecimentos que possam ajudar o programador a compreender o problema e a desenvolver a sua solução, usando para isto uma linguagem de programação.

2.1.2. Importância

A aprendizagem de programação de computadores é fundamental em cursos da área de Informática, uma vez que "este conhecimento objetiva ensinar como utilizar o computador para solucionar problemas do mundo real, através da construção de programas e sistemas computadorizados" (Ribeiro et al., 2019, p. 803), ao passo que as disciplinas relacionadas à programação são um dos alicerces dos cursos de computação e áreas afins (Esteves et al., 2019; Franzen et al., 2018; Stephan et al., 2020). De acordo com A. J. Gomes (2010), "o ensino da programação tem como propósito conseguir que os alunos desenvolvam as suas capacidades, adquirindo os conhecimentos e competências necessárias para conceber programas e sistemas computacionais capazes de resolver problemas reais" (p. 1).

Com o estudo que realizamos, pudemos verificar que o ensino de programação faz parte de diferentes currículos de cursos superiores. As disciplinas introdutórias de programação são a base para

o desenvolvimento lógico e algorítmico de alunos desses cursos, sendo essenciais para a construção da fundamentação necessária para o entendimento de tópicos mais avançados (França & Tedesco, 2015) de forma que seu ensino tem como objetivo que os alunos adquiram os conhecimentos básicos e competências necessárias para conceber programas capazes de resolver problemas reais (Medeiros, 2019).

2.1.3. Cursos

Como já apresentado anteriormente, as Unidades Curriculares de programação estão presentes em diversos currículos do Ensino Superior. Várias disciplinas presentes nos cursos da área de Tecnologia da Informação (TI) apresentam a programação de computadores como foco central, além de permanecerem ao longo do curso, de forma recorrente (A. Santos et al., 2015).

Na área de computação observa-se a intensificação de esforços em disciplinas que envolvem aspectos teóricos e práticos de programação, devido à importância destas disciplinas no currículo de cursos como Ciência da Computação, Sistemas de Informação, Engenharia da Computação e afins, tanto no contexto nacional [brasileiro] quanto no cenário internacional (Marcolino & Barbosa, 2015).

Para o contexto brasileiro, o ensino e aprendizado dos conceitos relacionados à programação de computadores é reservado, principalmente, aos estudantes que optam por cursos técnicos e de graduação na área da Computação e Informática (Berssanette, 2020). Em Araujo et al. (2019), são apresentados os cursos da área de Computação, catalogados pela Sociedade Brasileira de Computação (SBC), os quais possuem a disciplina de programação em seu currículo. São eles: Ciência da Computação, Engenharia da Computação, Engenharia de Software, Licenciatura em Computação, Sistemas de Informação, e os Cursos Superiores em Tecnologia, que englobam: Análise e Desenvolvimento de Sistemas, Banco de Dados, Gestão da Tecnologia da Informação, Gestão de Telecomunicações, Jogos Digitais, Redes de Telecomunicações, Segurança da Informação, Sistemas de Telecomunicações, Sistemas Embarcados, Sistemas para Internet e Telemática.

Já no contexto internacional, o documento *Computing Curricula* 2020 da *Association for Computing Machinery* (ACM) e *IEEE Computer Society* (IEEE-CS) apresenta os seguintes cursos como sendo da área de computação: Engenharia da Computação, Ciência da Computação, Segurança Cibernética, Sistemas de Informação, Tecnologia da Informação, Engenharia de Software e Ciência dos Dados (ACM & IEEE, 2020).

Ainda de acordo com o documento desenvolvido pela ACM e IEEE (ACM & IEEE, 2020), antes dos anos 1990, os cursos eram divididos em três áreas: hardware, software e *business*, com os cursos de Engenharia Elétrica e Engenharia da Computação na área de hardware, Ciência da Computação na

área de software e Sistemas de Informação na área de negócios. Após os anos 1990, manteve-se as áreas de hardware e software, e houve mudança de negócios para necessidades organizacionais. O curso de Engenharia Elétrica manteve-se na área de hardware, mas o de Engenharia da Computação passou a contemplar as áreas de hardware e software simultaneamente. A área de software, além de Ciência da Computação passou a ter, portanto, os cursos de Engenharia da Computação e Engenharia de Software, e a área de necessidades organizacionais os cursos de Tecnologia da Informação e Sistemas de Informação.

No âmbito de Portugal, analisando os cursos superiores das Universidades Universidade do Minho, Universidade de Coimbra, Universidade Aberta, Universidade de Lisboa e Instituto Politécnico de Bragança, encontramos os seguintes cursos que possuem Unidades Curriculares (UC) de programação em sua matriz curricular: Ciências da Computação, Engenharia Eletrônica e de Computadores, Engenharia Informática, Engenharia e Ciência de Dados, Engenharia Mecânica, Engenharia e Gestão Industrial, Matemática e Aplicações, Tecnologias de Informação, Engenharia Geoespacial, Informática de Gestão, Informática e Comunicações, Tecnologias da Comunicação.

De acordo com Pedrosa (2017):

> várias engenharias incluem como parte da sua formação a programação de computadores. Contudo, é na Engenharia Informática ou na sua congénere mais especializada, a Engenharia de Software, que a programação de computadores surge como competência central, com diversas camadas de complexidade. (p. 8)

Embora as UC de programação estejam predominantemente em cursos correlatos à área de Informática, como as citadas nesta seção, vê-se cada vez mais a introdução de programação, mesmo que de forma extracurricular [por meio da Extensão Universitária, por exemplo], nos mais diversos cursos superiores, com a finalidade de proporcionar aos estudantes conhecimentos iniciais de programação, que servirão para a sua formação, e com intuito de promover a integração com linguagens de programação e motivar a codificação e resolução de problemas práticos, de acordo com o contexto do curso superior.

2.1.4. Designação das Unidades Curriculares

No Ensino Superior, as UC de programação geralmente aparecem durante todo o curso, e de forma recorrente. As UC de introdução à programação fazem parte de diferentes currículos de graduação, e são denominadas como *Computer Science* 1 (CS1) *e Computer Science* 2 (CS2) (ACM & IEEE, 2020). Estas disciplinas são focadas em programação, a partir das quais os alunos aprendem os conceitos inerentes à Ciência da Computação, por meio de tarefas explícitas com o intuito de aprender

uma linguagem de programação (Medeiros, 2019). Araujo et al. (2019) apresentam para cada curso superior da área de Informática as disciplinas que fazem parte delas. Aqui listamos, em ordem alfabética, as diferentes nomenclaturas de UC de programação apresentadas nos Referenciais de Formação:

- Abstração e Estrutura de Dados
- Algoritmos
- Complexidade de Algoritmos
- Estrutura de Dados
- Fundamentos de Linguagens de Programação
- Implementação de Sistemas Embarcados
- Paradigmas e Padrões de Programação
- Paradigmas de Programação
- Programação
- Programação de Aplicações Web
- Programação de Aplicativos para Dispositivos Móveis
- Programação de Computadores
- Programação de Software Básico
- Programação em Linguagem de Montagem
- Programação em Linguagem Script
- Programação em Lógica
- Programação Funcional
- Programação Imperativa
- Programação Orientada a Objetos
- Programação para Aplicativos Móveis
- Programação Paralela e Distribuída
- Projeto de Algoritmos
- Técnicas de Programação

Analisando matrizes curriculares de cursos da área de computação em Universidades brasileiras, podemos verificar algumas nomenclaturas variantes das apresentadas por Araujo et al. (2019), mas que possuem como cerne as apresentadas nos Referenciais de Formação. Alguns exemplos dessas variantes são:

- Construção e Análise de Algoritmos

- Laboratório de Algoritmos e Estrutura de Dados (I e II)
- Linguagens de Programação
- Programação Avançada
- Programação Estruturada

No contexto internacional, ACM & IEEE (2020) apresentam as UC de cursos superiores da área de Computação. As disciplinas relacionadas com programação que estão presentes no documento *Computing Curricula 2020* são:

- CS1 e CS2 (disciplinas introdutórias)
- Estrutura de Dados, Algoritmos e Complexidade
- Fundamentos de Programação
- Linguagens de Programação

No âmbito de Portugal, analisando os Planos de Estudo de cursos superiores de Informática da Universidade do Minho, Universidade de Coimbra, Universidade Aberta, Universidade de Lisboa e Instituto Politécnico de Bragança, encontramos as seguintes designações para as UC de programação:

- Algoritmos e Estrutura de Dados
- Desenvolvimento de Aplicações
- Estratégias Algorítmicas
- Introdução à Programação
- Introdução à Programação e Resolução de Problemas
- Laboratório de Algoritmia I
- Laboratório de Algoritmia II
- Laboratório de Programação
- Linguagens de Programação
- Princípios de Programação Procedimental
- Programação Funcional
- Programação Imperativa
- Programação Orientada por Objetos

Embora as designações entre diferentes países e Universidades sejam diferentes, os Planos de Estudo das UC de programação são semelhantes, no sentido em que trazem os mesmos assuntos e métodos para o ensino de programação. Os Planos se preocupam especialmente com a semântica e sintaxe da linguagem adotada, bem como com detalhes tais como o ensino de variáveis, de estruturas de condições e repetições, das disciplinas iniciais (como Introdução à Programação, Laboratório de

Programação e Construção e Análise da Algoritmos), tendo como ênfase o paradigma de Programação Estruturada, usando para tanto a Linguagem C. Quando tratam de Programação Orientada a Objetos, é explícito que a preocupação é desenvolver a prática de programação com ênfase no paradigma de Programação Orientada a Objetos, ensinando, com uma linguagem de programação como Java ou C# os conceitos de classes, heranças e polimorfismo, por exemplo. Dessa forma, independente da UC nota-se que o conteúdo ministrado está muito ligado ao paradigma e à linguagem de programação adotada pelo curso ou pelo docente, e as metodologias aplicadas para o ensino são pautadas especialmente em aulas expositivas e práticas em laboratório.

2.2. Caracterização

Esta seção caracteriza o processo de ensino e aprendizagem de programação, apresentando o que nos falam os autores sobre as 'dificuldades enfrentadas pelo aluno durante a aprendizagem de programação', as 'condições para aprendizagem de programação' apresentando conhecimentos, habilidades e fatores importantes para o processo de aprendizagem, e as 'metodologias e ferramentas' adotadas por professores e alunos para o ensino e aprendizagem de programação.

2.2.1. Dificuldades Enfrentadas pelo Aluno Durante a Aprendizagem de Programação

Ao longo do seu percurso acadêmico, diferentes são as dificuldades e os níveis de dificuldades enfrentados pelos alunos no processo de aprendizagem de programação. Dependendo dos conhecimentos, habilidades e competências de cada indivíduo, as dificuldades apresentam-se de formas diferentes: alguns alunos sentem mais dificuldades, enquanto outros podem aprender a programar mais facilmente, quase que naturalmente. Neste tópico, são apresentadas as mais diversas dificuldades encontradas na literatura que podem ser enfrentadas pelos alunos ao longo do processo de aprendizado de programação.

Quando se trata de estudar programação de computadores, verifica-se que os processos de ensino e de aprendizagem de computação não são de conhecimento geral da população, já que estes conteúdos não são estudados no Ensino Básico e Fundamental. Quando os alunos ingressam no Ensino Superior, na área de Informática, eles se deparam com a Lógica de Programação e chega-se a um momento crítico, gerando um alto índice de desistências. A lógica de programação, bem como o estudo de linguagens de programação, exige um esforço real e o nível de dificuldade empregado é alto (Silveira et al., 2018). Fundamentos de programação não é um assunto fácil de ser ensinado - muitos alunos têm dificuldade em entender os conceitos abstratos de programação e têm uma visão errada sobre a atividade de programação (de Souza et al., 2011). Essas disciplinas são consideradas difíceis por muitos alunos iniciantes, uma vez que exigem habilidades de abstração lógico-matemáticas, as

quais nem todos desenvolvem em seu cotidiano (A. L. A. Raabe & Silva, 2005). Nesta seção, são apresentadas as principais dificuldades enfrentadas pelos alunos durante o processo de aprendizagem de programação que estão reveladas na literatura.

De acordo com Ambrósio et al. (2011), uma das dificuldades enfrentadas pelos estudantes está relacionada à transição do acadêmica (passagem do Ensino Médio para o Ensino Superior) e ao processo de adaptação ao Ensino Superior. Geralmente, alunos de cursos de Informática entram muito cedo na Universidade, e fatores como falta de experiência na gestão de tempo, relação com professores, novos processos didáticos e o próprio ambiente universitário (M. A. C. N. Gomes et al., 2019) podem ser determinantes para seu processo de aprendizagem.

A. J. Gomes & Mendes (2007) citam que as atitudes demonstradas pelos discentes podem dificultar o seu processo de aprendizagem de programação. Exemplos de atitudes que podem desencadear as dificuldades são: os alunos não estabelecerem analogias corretas com problemas já resolvidos anteriormente; não refletirem sobre o problema e, dessa forma, tentar resolvê-lo sem pensar de forma mais cuidadosa; falta de persistência, muitas vezes desistindo de resolver o problema se não encontram rapidamente uma solução.

Alunos que possuem uma base matemática fraca também podem sofrer dificuldades durante o processo de aprendizagem de programação (Chaves et al., 2013a; Franzen et al., 2018); A. J. Gomes & Mendes, 2007), isso por que a falta das habilidades necessárias em matemática leva à desmotivação dos alunos por não conseguirem fazer a abstração necessária para resolver os problemas (Medeiros et al., 2020).

Outra dificuldade enfrentada pelos alunos é a capacidade de raciocinar logicamente (Franzen et al., 2018; A. J. Gomes & Mendes, 2007; Silva et al., 2015). Conforme Neto & Schuvartz (2007), acadêmicos iniciantes, ao se depararem com a disciplina, sentem-se incapazes de programar devido ao conjunto de habilidades que a programação exige como capacidade o raciocínio lógico. Rapkiewicz et al. (2006), comentam que muitos dos alunos não conseguem desenvolver o raciocínio lógico necessário para o posterior desenvolvimento de programas, sendo este um dos principais motivos da desistência dos ingressantes em programação (Medeiros, 2019). Esta dificuldade é agravada pelo fato de que os docentes encontram dificuldades em trabalhar o raciocínio lógico de seus alunos, uma vez que nem sempre dispõem dos recursos didáticos necessários, e os estudantes, por sua vez, nem sempre se sentem motivados a buscar esses conhecimentos (Galdino et al., 2015).

A codificação de uma solução também é um fator que dificulta o processo de aprendizagem de programação (Silva et al., 2015; Jenkins, 2001; Silva & Trentin, 2016), especialmente pela dificuldade

do aluno migrar de um pseudocódigo para uma linguagem de programação específica, ou ainda quando os alunos partem para a codificação sem antes entender o problema a ser trabalhado (Medeiros, 2019). Além disso, de acordo com A. J. Gomes (2010), o paradigma de programação adotado e a linguagem de programação usada têm um considerável impacto no processo de aprendizagem e, consequentemente, no desempenho da tarefa. Entretanto não há consenso em relação ao melhor paradigma ou melhor linguagem de programação que devem ser adotados para o processo de ensino e aprendizagem de programação (Tavares, 2018). Atrelada à dificuldade de codificar, A. J. Gomes (2010) cita que a compilação de um programa também é uma dificuldade enfrentada pelos alunos, bem como o tratamento de erros código (Silva et al., 2015; (R. Santos & Menezes, 2019; A. J. Gomes, 2010).

Denning & Martell (2015) evocam que o grande desafio do ensino e aprendizagem de programação consiste em compreender o processo de programação e a prática do programador para proporcionar uma transferência educativa eficaz de conhecimentos e aptidões. Desta forma, é imprescindível que professor e aluno compreendam que, para se ter uma aprendizagem efetiva, é necessário que sejam desenvolvidas habilidades que vão desde a interpretação do problema ao desenvolvimento de sua solução.

Em sua pesquisa, Moreira et al. (2018) apontam que as dificuldades ligadas à programação identificadas há décadas ainda persistem e continuam preocupantes. Os resultados da pesquisa desenvolvida indicam que a maior dificuldade desses alunos está no desenvolvimento da lógica de programação.

Assim como a dificuldade em desenvolver a lógica de programação, a dificuldade em entender conceitos-chave da programação, tais como sintaxe, semântica, variáveis, tipos de dados, estrutura de controle e memória dinâmica agravam o processo de aprendizagem dos estudantes (Alves et al., 2019; Moreira et al., 2018; Silva & Trentin, 2016; A. J. Gomes, 2010). Aureliano et al. (2016) apontam que "o maior dos problemas vivenciados pelos iniciantes em programação não parece ser o entendimento dos conceitos básicos de uma LP [Linguagem de Programação], mas a combinação e a utilização adequada destes conceitos na construção de um determinado programa" (p. 2067).

De acordo com Silva et al. (2015), de maneira geral, as dificuldades enfrentadas pelos alunos são recorrentes, não refletindo algo incomum, pois os estudantes de computação apresentam os mesmos problemas, dentre eles a dificuldade em especificar uma solução (Jenkins, 2002; T. R. da Silva et al., 2015).

Medeiros (2019), aponta que a falta de *feedback* imediato dos professores aos exercícios dos alunos causa a desmotivação destes, e, desta forma, pode dificultar a aprendizagem de programação. Na pesquisa realizada por Santos & Menezes (2019), os autores relatam que sem ter o *feedback* dos professores em tempo hábil, os estudantes chegam a desistir de realizar as atividades ou buscam ajuda com os próprios colegas de curso.

Na revisão de literatura realizada, a dificuldade mais citada é a falta de habilidade de abstração (G. Alves et al., 2019; Bennedsen & Caspersen, 2019; Bittencourt et al., 2013; E. S. de Almeida et al., 2002; A. de J. Gomes, 2010; Koliver et al., 2004; Lahtinen et al., 2005; McGettrick et al., 2005; Medeiros et al., 2020; Nobre & de Menezes, 2002; A. L. A. Raabe & Silva, 2005; Silva & Trentin, 2016; T. R. da Silva et al., 2015). Para Alves et al. (2019), as disciplinas que envolvem o ensino de programação têm se mostrado um grande obstáculo para alguns alunos da área de Informática, e uma das dificuldades apresentadas pelos alunos é a capacidade de abstração. De acordo com Medeiros et al. (2020), "a falta das habilidades necessárias em matemática leva à desmotivação dos alunos por não conseguirem fazer a abstração necessária para resolver os problemas" (p. 6). Koliver et al. (2004) explicam que, muitas vezes, o professor inicia a disciplina comum a perspectiva equivocada a respeito das habilidades de seus alunos que, segundo os autores, geralmente chegam na sala de aula sem habilidade de abstração. De Almeida et al. (2002) afirmam que o problema é devido a uma forte carga de conceitos abstratos que permeiam todo o conhecimento envolvido na atividade de programação. Bittencourt et al. (2013) acrescentam que uma das dificuldades de aprender programação reside no fato de ser necessário aprender, em um curto espaço de tempo, conceitos diversos e desenvolver além da habilidade de abstração, o pensamento computacional, sendo que a falta destes podem causar problemas durante o processo de aprendizagem de programação.

A falta de tempo para se dedicar aos estudos também é apontada como uma dificuldade enfrentada pelos alunos durante a aprendizagem de programação (Giraffa & Mora, 2013; Medeiros, 2019; Moreira et al., 2018). De acordo com a pesquisa realizada por Giraffa & Mora (2013), um dos principais motivos que levam à desistência dos alunos em curso de programação é a falta de tempo, sendo as principais causas para a falta de tempo: "fatores como jornada dupla (trabalho, conjugado com estudos), dificuldades para organizar seu tempo, excesso de disciplinas e excesso de conteúdos" (p. 7).

Assim como possuir uma base matemática fraca, a dificuldade em interpretação textual também é um problema enfrentado pelos alunos para aprender a programar, de forma que diversos acadêmicos iniciantes nas disciplinas de programação consideram como uma barreira o fato das

disciplinas contemplarem conteúdos envolvendo interpretação de texto (Franzen et al., 2018; Medeiros, 2019). De acordo com Medeiros et al. (2020), alguns autores argumentam que "os conhecimentos em português e interpretação de texto são considerados fundamentais e basilares para os alunos entenderem as formulações dos problemas e partirem para a abstração do problema" (p. 4).

A dificuldade de interpretar ou compreender os problemas propostos durante o processo de aprendizagem de programação é também apresentada na literatura (Moreira et al., 2018; Nobre & de Menezes, 2002; R. Santos & Menezes, 2019; T. R. da Silva et al., 2015; Tavares, 2018). Para Zanini & Raabe (2012), diante de um problema a ser resolvido, o aluno pode ter diversas estratégias, mas três etapas são necessárias: "(i) compreensão do problema (abstração dos dados); (ii) elaboração de uma sequência lógica de instruções (solução proposta); (iii) avaliação e depuração da solução proposta)" (p.2) e os autores complementam afirmando que "parte da dificuldade encontrada pelos alunos, na compreensão dos problemas, pode estar atrelada à forma como são apresentados" (p. 2). De acordo com Proulx (2000), a dificuldade de compreender um problema pode ser devido à falta de familiaridade do aluno com o assunto ou devido à incapacidade de interpretar o enunciado do problema.

Para além das dificuldades já citadas, dois fatores podem interferir no processo de aprendizagem de programação: o método de ensino adotado pelo professor (A. Gomes & Mendes, 2007; A. L. A. Raabe & Silva, 2005) e o método de estudo adotado pelos alunos (A. Gomes & Mendes, 2007; R. Santos & Menezes, 2019). Os professores encontram dificuldades em trabalhar o raciocínio lógico de seus alunos, uma vez que nem sempre dispõem dos recursos didáticos necessários, e os estudantes, por sua vez, nem sempre se sentem motivados a buscar esses conhecimentos (Galdino et al., 2015). Já os alunos apontam que as informações obtidas nos recursos de apoio nem sempre são adequados por não oferecem um conteúdo contextualizado com a dificuldade e por nem sempre possibilitar uma pesquisa rápida (Santos & Menezes, 2019).

Conforme os resultados obtidos com este estudo, categorizamos as dificuldades da seguinte forma:

a) Adaptação: diz respeito ao processo de mudança do Ensino Médio para o Superior enfrentado pelo aluno.
b) Background: indica questões inerentes à formação do aluno que são anteriores ao seu acesso ao Ensino Superior.
c) Complexidade da programação: dificuldades enfrentadas devido à natureza complexa da programação.

d) Conhecimento específico da programação: dificuldades em assuntos específicos da programação.
e) Conhecimentos gerais: dificuldades em assuntos gerais, não apenas da programação, mas de quaisquer outras áreas do conhecimento.
f) Estrutura do curso: dificuldades enfrentadas devido à estrutura do curso, que são externos ao aluno.
g) Fator pessoal: fatores inerentes ao aluno, como questões pessoais, familiares e socioeconômicas.
h) Habilidade: dificuldades enfrentadas pela falta ou pouca habilidade.
i) Metodologia do discente: problemas enfrentados no processo de ensino e aprendizagem devido à metodologia de estudo adotada pelo discente.
j) Metodologia do docente: problemas enfrentados no processo de ensino e aprendizagem devido à metodologia de ensino adotada pelo docente.
k) Recursos de apoio: questões relacionadas aos recursos adotados por professores e alunos no processo de ensino e aprendizagem de programação.

A Tabela 1 apresenta as dificuldades identificadas neste estudo e na RSL (Morais et al., 2020), bem como os respectivos autores que as citam e suas categorias.

Tabela 1 - *Dificuldades enfrentadas pelos alunos durante o processo de aprendizagem de programação*

Categoria	Dificuldade	Autores FT*	Autores RSL**
Adaptação	Problemas na adaptação ao Ensino Superior	(Ambrósio et al., 2011)	
Adaptação	Dificuldades na Transição acadêmica	(Ambrósio et al., 2011)	
Background	Currículo		(A. Gomes & Mendes, 2015, 2010; Mendes et al., 2012; Piteira et al., 2017)
Background	Base matemática fraca	(Franzen et al., 2018; Gomes &	(Bosse & Gerosa, 2017; T. H. Castro et

		Mendes, 2007; Medeiros et al., 2020)	al., 2008; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015, 2010; Rezende & Bispo, 2018; Shadiev et al., 2013)
Background	Falta de experiência		(Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2010)
Background	Dificuldade em interpretar/entender um problema proposto (interpretação textual)	(Medeiros, 2019; Moreira et al., 2018; Nobre & de Menezes, 2002; Santos & Menezes, 2019; da Silva et al., 2015; Tavares, 2018)	(Aparicio & Costa, 2018; Bosse & Gerosa, 2017; T. H. Castro et al., 2008; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015, 2010; Mendes et al., 2012; Ortiz-Ortiz et al., 2018; Rezende & Bispo, 2018; Souleiman, 2017)
Complexidade da programação	Natureza da programação complexa	(Gomes & Mendes, 2007)	
Conhecimento específico da programação	Não compreender o processo de programação	(Denning & Martell, 2015)	
Conhecimento específico da programação	Não entender os conceitos-chave	(Alves et al., 2019; Gomes, 2010; Medeiros, 2019; Moreira et al., 2018; Silva & Trentin, 2016)	(Aparicio & Costa, 2018; Eltegani & Butgereit, 2015; Piteira et al., 2017; Rezende & Bispo, 2018; A. Santos et al., 2011)

Conhecimento específico da programação	Dificuldade no tratamento de erros do código	(Gomes, 2010; Santos & Menezes, 2019; Silva et al., 2015)	
Conhecimento específico da programação	Dificuldade para compilar um programa	(Gomes, 2010)	
Conhecimento específico da programação	Dificuldade no desenvolvimento da lógica de programação	(Moreira et al., 2018)	
Conhecimento específico da programação	Dificuldade em encontrar erros durante a programação		(Souleiman, 2017)
Conhecimentos específico da programação	Dificuldade me especificar uma solução	(Jenkins, 2002; Santos & Menezes, 2019; Silva & Trentin, 2016; Silva et al., 2015)	
Estrutura do curso	Turmas heterogêneas		(A. Gomes & Mendes, 2015; Mendes et al., 2012; A. Santos et al., 2010, 2013; Souleiman, 2017)
Estrutura do curso	Turmas grandes		(Mendes et al., 2012; A. Santos et al., 2010, 2013; Souleiman, 2017)
Fator pessoal	Atitudes demonstradas pelo	(Gomes & Mendes, 2007)	(Aparicio & Costa, 2018; Bosse & Gerosa,

	aluno que não contribuem com o aprendizado		2017; T. H. Castro et al., 2008; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015, 2010; Piteira et al., 2017)
Fator pessoal	Autoconfiança baixa		(Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015)
Fator pessoal	Desmotivação		(A. Gomes & Mendes, 2015; Mendes et al., 2012; Ortiz-Ortiz et al., 2018; Piteira et al., 2017; A. Santos et al., 2013; Skalka & Drlik, 2018)
Fator pessoal	Estresse		(Skalka & Drlik, 2018)
Fator Pessoal	Falta de tempo para se dedicar à disciplina	(Giraffa & Mora, 2013; Medeiros, 2019; Moreira et al., 2018)	(Rezende & Bispo, 2018; A. Santos et al., 2011)
Habilidade	Dificuldade em raciocinar logicamente	(Franzen et al., 2018; Galdino et al., 2015; Gomes & Mendes, 2007; Neto & Schuvartz, 2007; Rapkiewicz et al., 2006; Silva et al., 2015)	
Habilidade	Dificuldade em codificar uma solução	(Gomes, 2010; Jenkins, 2002; Silva & Trentin,	

		2016; Silva et al., 2015; Tavares, 2018)
Habilidade	Falta de habilidade de abstração	(G. Alves et al., 2019; Bennedsen & Caspersen, 2019; Bittencourt et al., 2013; E. S. de Almeida et al., 2002; A. de J. Gomes, 2010; Koliver et al., 2004; Lahtinen et al., 2005; McGettrick et al., 2005; Nobre & de Menezes, 2002; A. L. A. Raabe & Silva, 2005; SILVA & TRENTIN, 2016; T. R. da Silva et al., 2015)
Habilidade	Falta de habilidade de pensamento computacional	(Bittencourt et al., 2013)
Metodologia do discente	Método de estudo falho adotado pelo discente	(Gomes & Mendes, 2007; Santos & Menezes, 2019)
Metodologia do docente	Falta de ajuda (*feedback*) por parte	(Medeiros, 2019; Santos &

	do docente	Menezes, 2019)	
Metodologia do docente	Dissociação entre os exercícios propostos e o mundo real		(Al-Imamy et al., 2006; Mendes et al., 2012; Rezende & Bispo, 2018)
Metodologia do docente	Problemas com a Linguagem de programação adotada pelo professor		(Aparicio & Costa, 2018; Bosse & Gerosa, 2017; A. Gomes & Mendes, 2015; Piteira et al., 2017)
Metodologia do docente	Problemas com o método de ensino adotado pelo docente	(Gomes & Mendes, 2007; Raabe & Silva, 2005)	
Recursos de apoio	Falta de acesso a recursos didáticos	(França & Amaral, 2012; Santos & Menezes, 2019)	
Recursos de apoio	Não conhecer ferramentas de desenvolvimento de software	(Gomes, 2010)	

Nota: *Fundamentação teórica. **Revisão Sistemática de Literatura. Autoria própria.

Disciplinas introdutórias de programação sofrem devido a altas taxas de reprovações e desistências. Esta é uma situação que afeta principalmente os novatos, pois estas disciplinas geralmente são ofertadas no início dos cursos. De acordo com Mendes et al. (2012), diferentes autores sugeriram possíveis causas para as dificuldades dos alunos, como a natureza da programação, o histórico dos alunos e as atitudes de estudo bem como as estratégias pedagógicas comumente usadas em disciplinas de algoritmos e programação.

Estas dificuldades podem causar desde reprovações nas disciplinas de algoritmos e programação até desistências do curso. As consequências das dificuldades estão categorizadas da seguinte forma:

a) Fator pessoal: fatores inerentes ao aluno, como questões pessoais, familiares e socioeconômicas.

b) Percurso acadêmico: consequências que afetam a trajetória do aluno dentro do curso.

A Tabela 2 apresenta as consequências das dificuldades enfrentadas, de acordo com sua categorização, e seus respectivos autores.

Tabela 2 - *Consequências das dificuldades enfrentadas pelos alunos durante o processo de aprendizagem de programação*

Categoria	**Consequências das dificuldades**	**Autores FT***	**Autores RSL****
Fator pessoal	Desmotivação	(G. Alves et al., 2019; Cambruzzi & Souza, 2015; I. Silva et al., 2009).	(A. Gomes & Mendes, 2015; A. Santos et al., 2013; Skalka & Drlik, 2018)
Fator pessoal	Frustração		(Ortiz-Ortiz et al., 2018; A. Santos et al., 2013; Skalka & Drlik, 2018)
Percurso acadêmico	Alto índice de evasão	(G. Alves et al., 2019; Aureliano et al., 2016; Chaves et al., 2013a; Galdino et al., 2015; A. de J. Gomes, 2010; Medeiros, 2019; T. R. da Silva et al., 2015).	(Bosse & Gerosa, 2017; A. Gomes & Mendes, 2015; Mendes et al., 2012; T. A. N. de Oliveira & Rebouças, 2018; Ortiz-Ortiz et al., 2018; A. Santos et al., 2011, 2013; Skalka & Drlik, 2018; Souleiman, 2017)
Percurso acadêmico	Alto índice de reprovação	(Alves et al., 2019; Aureliano et al., 2016; Cambruzzi & Souza, 2015; Chaves et al., 2013; Gomes, 2010; Medeiros, 2019; Silva et al., 2015; Silva & Borges, 2016;	(Bosse & Gerosa, 2017; Eltegani & Butgereit, 2015; Gomes & Mendes, 2010; Mendes et al., 2012; Oliveira & Rebouças, 2018; Ortiz-Ortiz et al., 2018; A.

		Tavares, 2018)	Santos et al., 2010, 2011, 2013; Skalka & Drlik, 2018; Souleiman, 2017)
Percurso acadêmico	Baixo rendimento nas disciplinas subsequentes do curso	(Cardoso & Antonello, 2015)	
Percurso acadêmico	Mudança de curso	(Cardoso & Antonello, 2015)	

Nota: *Fundamentação teórica. **Revisão Sistemática de Literatura. Autoria própria.

Assim, com base na literatura, pudemos elencar neste tópico as principais dificuldades enfrentadas pelos alunos ao longo do processo de aprendizagem de programação. Além disso, foi possível estabelecer as consequências geradas por estas dificuldades. Embora nem todos os alunos passem por dificuldade, percebemos que as dificuldades são recorrentes, o que leva a comunidade acadêmica a desenvolver métodos e ferramentas que possam auxiliar o processo de ensino e aprendizagem de programação, na tentativa de diminuir ou extinguir as dificuldades e as consequências delas, na busca de proporcionar aos atores do processo maior êxito em seu percurso acadêmico.

2.2.2. Condições para Aprendizagem de Programação

Uma vez que pudemos identificar dificuldades enfrentadas pelos alunos ao longo do processo de aprendizagem de programação, aqui apresentamos que condições são necessárias para que os alunos possam aprender a programar. Além disso, apresentamos que fatores influenciam para que existam dificuldades para esse aprendizado, bem como que conhecimentos, habilidades ou aptidões podem influenciar neste processo.

2.2.2.1. Fatores que Podem Influenciar na Aprendizagem. O ensino de programação é um desafio complexo, pois depende de uma diversidade de fatores, incluindo: experiência e competências que o professor tem em programação, a abordagem pedagógica e o tipo de instrumentos utilizados (Kumar & Khurana, 2012). Também é difícil para o professor proporcionar um ambiente de aprendizagem que beneficie todos os alunos, devido às diferentes atitudes que cada aluno tem perante a aprendizagem de programação. Nesta seção, apresentamos fatores encontrados na literatura que podem influenciar no processo de aprendizagem da programação.

As competências de algoritmos e lógica de programação, apesar de poderem ser praticadas e desenvolvidas na infância e adolescência, acabam recebendo atenção apenas quando o aluno ingressa no Ensino Superior. Este contato súbito e tardio com as disciplinas de lógica e algoritmo pode potencializar dificuldades na absorção do assunto, sendo importante encontrar abordagens de ensino que contornem ou amenizem tais dificuldades, como já apresentado anteriormente.

Para A. Raabe et al. (2015), “o primeiro contato dos estudantes com os conceitos de programação pode ser determinante na forma como perceberão os desafios e enfrentarão as dificuldades inerentes à aprendizagem de lógica” (p.1), sendo isto um fator que pode influenciar no processo de adaptação e aprendizagem de programação.

Piva Jr & Freitas (2010) citam que "a cada novo ano, a cada nova turma, o nível dos alunos que advém do Ensino Médio é mais e mais preocupante" (p. 1). Segundo eles, a maioria desses alunos entra no Ensino Euperior sem uma base adequada em disciplinas como Português e Matemática, o que implica em sérias dificuldades na interpretação de textos e na resolução de equações matemáticas. Assim, um dos motivos para evasão e desinteresse na área são problemas que os alunos enfrentaram ao longo de sua formação básica (Paula et al., 2009), e que podem desencadear deficiências na base matemática e dificuldades de interpretação de problemas, o que pode fazer com que muitos alunos não apresentem as competências necessárias para a resolução de problemas, o que prejudica sua aprendizagem de programação.

Além disso, as pessoas que mostram interesse na área de computação e programação geralmente são heterogêneas: são pessoas que diferem em características como sexo, idade, nível de escolaridade, maneira de aprender e aptidão para a resolução de problemas. Isto torna difícil a criação de um material único que atenda a todas as diferentes características (A. S. Oliveira et al., 2015).

Aureliano et al. (2016) afirmam que é extremamente difícil para um professor atender todas as demandas dos alunos ocorridas durante a realização das atividades e esse problema agrava-se mais quando o aluno se encontra fora dos limites de sala de aula.

Por exemplo, um fato comumente conhecido entre professores de programação é que, diferentemente dos alunos iniciantes na primeira linguagem de programação, aqueles que já possuem experiência prévia em alguma linguagem de programação sentem menos dificuldades para aprender a segunda linguagem. Há, portanto, um forte indício de que o conhecimento prévio sobre resolução de problemas computacionais, ou seja, a habilidade de construir planos de programação, seja um pré-requisito para o aprendizado de linguagens de programação.

Entretanto, além dos problemas enfrentados pelos alunos em sua formação escolar, nas áreas das Ciências Exatas constata-se que um dos maiores problemas existentes nos cursos superiores envolve a falta de formação pedagógica do seu corpo docente. A maioria dos professores que atua nesta área possui experiência profissional e pós-graduação, habilitando-os ao exercício da docência. Entretanto, muitos não possuem nenhum tipo de formação pedagógica, que os habilitem a atuar adequadamente em sala de aula. Na maioria das vezes, suas atividades em sala de aula baseiam-se em estilos de seus ex-professores, ou seja, o professor aplica com seus alunos o estilo de aula de um professor com o qual se identificou durante sua graduação ou pós-graduação (Pimenta & Anastasiou, 2002). Grillo (2008) coloca que

> é frequente o professor orientar-se pela memória afetiva e procurar reproduzir ou evitar desempenhos conhecidos por ele enquanto aluno, ou, já como professor, imitar algum colega que o influenciou. Nesse sentido, visualiza-se o professor como um especialista (...), na ideia de que para ensinar basta ter domínio do conteúdo. Como especialista, o professor julga possuir um conhecimento pronto, acabado, existente no exterior do sujeito que aprende. O ensino, então, tem só uma modalidade: dar aula, transmitindo-se ao aluno esse conhecimento. (p. 62)

Aliado a isso, os estudantes de programação adotam metodologias de estudo que são ineficientes. Eles decoram soluções para os problemas ao invés de entendê-las (A. Gomes & Mendes, 2007), estudam de maneira passiva ou superficial (Biggs & Tang, 2011; Clark et al., 2011) e fazem poucos exercícios (A. Gomes & Mendes, 2007). De acordo com Perkins & Martin (1986), estudantes de programação possuem um tipo de conhecimento considerado frágil, uma vez que, embora possuam o conhecimento, não conseguem aplica-los em novas situações ou problemas encontrados.

A resistência perante a condução do aprendizado lógico se forma antes do começo das atividades, fato que pode ocorrer por sua característica lógico-matemática, dado o enraizado estigma relacionado aos conteúdos não abordados no nível médio. Dessa forma, o professor passa a necessitar de uma estratégia pedagógica adequada que contribua para o aprendizado, o que acaba deixando claro

que a absorção do conteúdo passado em sala de aula não depende apenas do nível de abstração do aluno.

As atividades propostas pelos professores para que os estudantes pratiquem programação, muitas vezes, são realizadas através de estratégias de tentativa e erro, que causam frustração na busca pela solução (Ettles et al., 2018). Esse processo, chamado de tentativa e erro, se mantém até que a solução correta seja alcançada (Almeida et al., 2017). Entretanto, as tentativas frustradas podem se tornar frequentes, fazendo com que o aluno sinta desmotivação para continuar praticando a atividade, o que pode levá-lo a desistir de entregar as atividades ou entregá-las incompletas. Impactos na motivação dos iniciantes são observados, dentre outras razões, porque muitos não possuem conhecimentos prévios na área, o que influencia o nível de complexidade que alguns julgam estar associado ao processo de aprender a codificar (Berssanette, 2016). A motivação dos alunos é fator indispensável para o seu sucesso em qualquer processo de aprendizagem, e quando estão motivados apresentam entusiasmo para realizar atividades e costumam orgulhar-se do próprio desempenho (Guimarães & Boruchovitch, 2004).

Outros fatores que podem influenciar e afetar o processo de ensino e aprendizagem de programação envolvem os problemas de escalabilidade, como salas com muitos alunos, poucos professores e monitores, e os problemas de infraestrutura, como laboratórios com poucos computadores ou, em alguns casos, sem acesso à internet ou sem infraestrutura adequada. Logo, o contexto pedagógico em que os alunos aprendem influencia o seu envolvimento e determinação para alcançar resultados de aprendizagem.

Categorizamos os fatores que podem influenciar [positivamente] ou afetar [negativamente] no processo de ensino e aprendizagem de programação da seguinte forma:

a) Adaptação: diz respeito ao processo de mudança do Ensino Médio para o superior enfrentado pelo aluno.
b) Apoio do docente: diz respeito ao apoio que o docente fornece ao aluno durante o processo de ensino.
c) Background: indica questões inerentes à formação do aluno que são anteriores ao seu acesso ao Ensino Superior.
d) Estrutura do curso: dificuldades enfrentadas devido à estrutura do curso, que são externos ao aluno.
e) Fator pessoal: fatores inerentes ao aluno, como questões pessoais, familiares e socioeconômicas.

f) Formação docente: diz respeito à formação profissional dos professores.

g) problemas enfrentados no processo de ensino e aprendizagem devido à metodologia de estudo adotada pelo discente.

h) Metodologia do docente: problemas enfrentados no processo de ensino e aprendizagem devido à metodologia de ensino adotada pelo docente.

i) Recursos de apoio: questões relacionadas aos recursos adotados por professores e alunos no processo de ensino e aprendizagem de programação.

A Tabela 3 apresenta de forma sintética os fatores que podem influenciar e afetar o processo de ensino e aprendizagem de programação, de acordo com a categorização e seus respectivos autores.

Tabela 3 - *Fatores que podem influenciar e afetar o processo de ensino e aprendizagem de programação*

Categoria	**Fatores**	**Autores FT***	**Autores RSL****
Adaptação	Primeiro contato com a programação	Raabe et al., 2015	
Apoio do docente	Apoio eficaz ao aluno		(Bosse & Gerosa, 2017; Mendes et al., 2012; Nawahdah et al., 2015; R. M. S. Pereira, 2017; A. Santos et al., 2010, 2013; Souleiman, 2017)
Apoio do docente	Feedback		(Bosse & Gerosa, 2017; T. Castro et al., 2011; Chuchulashvili et al., 2016; A. Gomes & Mendes, 2015; Nawahdah et al., 2015; Ortiz-Ortiz et al., 2018; Paredes et al., 2019; R. M. S. Pereira, 2017; Shadiev et al., 2013; Skalka & Drlik, 2018; Souleiman, 2017)

Background	Conhecimento prévio em programação	Aureliano et al., 2016	(Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2010)
Estrutura do curso	Adaptação a necessidades específicas dos alunos		(Mendes et al., 2012; A. Santos et al., 2010, 2013)
Estrutura do curso	Infraestrutura do curso	(Berssanette, 2016	
Fator pessoal	Compromisso		(Eltegani & Butgereit, 2015)
Fator pessoal	Motivação	(Guimarães & Boruchovitch, 2004	(Bosse & Gerosa, 2017; Chuchulashvili et al., 2016; Costa, 2019; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015, 2010; Hwang et al., 2012; Mendes et al., 2012; Nawahdah et al., 2015; Ortiz-Ortiz et al., 2018; R. M. S. Pereira, 2017; A. Santos et al., 2013; Souleiman, 2017)
Fator pessoal	Organização no estudo		(Chuchulashvili et al., 2016; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015)
Fator pessoal	Vocação		(A. Gomes & Mendes, 2015)
Formação docente	Experiência e competências que o professor tem em	(Kumar & Khurana, 2012)	

	programação		
Formação docente	Formação docente	(Pimenta & Anastasiou, 2002; Grillo, 2008)	
Metodologia do discente	Decorar uma solução ao invés de entendê-la	(Gomes & Mendes, 2007)	
Metodologia do discente	Estudar de maneira passiva ou superficial	(Biggs & Tang, 2011; Clark et al., 2011)	
Metodologia do discente	Resolver poucos exercícios	Gomes & Mendes, 2007	
Metodologia do docente	Abordagem pedagógica	(Kumar & Khurana, 2012)	
Metodologia do docente	Adaptação ao ritmo do aluno		(Eltegani & Butgereit, 2015; Paredes et al., 2019; A. Santos et al., 2013; Souleiman, 2017)
Metodologia do docente	Melhora nas habilidades dos alunos		(Aparicio & Costa, 2018; Eltegani & Butgereit, 2015; Ortiz-Ortiz et al., 2018; A. Santos et al., 2010, 2013)
Recursos de apoio	Ferramentas utilizadas	(Kumar & Khurana, 2012)	

Nota: *Fundamentação teórica. **Revisão Sistemática de Literatura. Autoria própria.

2.2.2.2. Competências, Conhecimentos e Habilidades que Podem Auxiliar no Processo de Aprendizagem.

Este tópico tem como objetivo apresentar que competências, conhecimentos e habilidades são definidos na literatura como basilares para o processo de aprendizagem em programação. CC2020 (ACM & IEEE, 2020) articula uma noção de competência como um objetivo educacional prático que refina a estrutura Knowledge-Skill-Disposition (Conhecimento – Habilidade – Disposição) (K-S-D). Enquanto as dimensões do conhecimento da computação foram amplamente exploradas nos vários currículos de computação, o que se entende por habilidade e disposição teve significativamente menos foco. Estendendo a estrutura K-S-D, os Relatórios CC2020 especificam a competência como composta das dimensões K-S-D observadas no desempenho de uma tarefa, T (Task).

Uma especificação de competência enumera conhecimentos, habilidades e disposições que são observáveis na realização de uma tarefa, que prescreve propósito dentro de um contexto de trabalho (ACM & IEEE, 2020). Assim, tomaremos como referência para este tópico os conceitos de competências, conhecimentos e habilidades definidos pelo CC2020, para conduzir a apresentação dos resultados obtidos da revisão de literatura:

a) *Conhecimento:* dimensão 'saber o quê' da competência como uma compreensão factual. Esta dimensão reflete, por exemplo, o assunto enumerado que os professores catalogam como tópicos em seus programas. Um elemento de conhecimento designa um conceito central essencial para uma competência.

b) *Habilidade*: introduz a capacidade de aplicar o conhecimento para realizar ativamente uma tarefa. Portanto, uma habilidade expressa um elemento de conhecimento conforme atuado com proficiência para definir a dimensão *know-how* da competência.

c) *Competência:* por sua vez, é a junção do conhecimento com a habilidade, adicionada com a disposição de um indivíduo para desenvolver uma determinada tarefa.

O aprendizado dos conceitos inerentes à programação exige do aluno o domínio de habilidades específicas relacionadas à abstração e ao raciocínio lógico, aptidões que são pouco exploradas no Ensino Médio (A. Santos et al., 2015), de forma que os discentes chegam ao Ensino Superior muitas vezes sem tê-las. Para Dijkstra (1989), aprender a programar é uma atividade complexa, chegando a ser desmotivadora. Compreender detalhadamente os conceitos de algoritmos e programação de computadores requer um conjunto grande de habilidades dos discentes que engloba a compreensão e abstração de um problema, articulação e modelagem da solução, a elaboração de códigos em uma

linguagem de programação ou pseudocódigo capaz de resolver o problema, e a possibilidade de identificar e corrigir erros nos códigos produzidos.

De acordo com A. J. Gomes (2010):

> Para aprender a programar não basta saber a sintaxe de determinada linguagem. Apesar de haver características específicas de determinada linguagem de programação que tenham de ser aprendidas, numa primeira fase a linguagem de programação deve ser apenas vista como um meio utilizado para resolver um problema de programação. (p. 29)

A aprendizagem de programação exige, portanto, o uso de complexas habilidades cognitivas, como raciocínio, resolução de problema, planejamento e criatividade, de maneira que o programador seja capaz de desenvolver representações de um problema sob a forma de estruturas lógicas, que são traduzidas em códigos, utilizando-se de uma linguagem formal (Mohorovičić & Strčić, 2011, como citado em Oliveira et al., 2015).

Em seu trabalho, Ambrósio et al. (2011) enumeram algumas competências necessárias para aprender programação: compreensão leitora; raciocínio crítico e pensamento sistêmico; metacomponentes cognitivas de identificação, planejamento e resolução de problemas; a criatividade e curiosidade intelectual; as habilidades matemáticas e o raciocínio condicional; o pensamento procedimental e o raciocínio temporal; o raciocínio analítico e o raciocínio quantitativo; o raciocínio analógico, silogístico e combinatório. Além disso, apresentam que, de acordo com alguns estudos, ter conhecimento prévio em matemática e em ciências apresenta correlação com o desempenho dos alunos no âmbito da aprendizagem de programação. Entretanto, embora ter conhecimento prévio em computação não seja um fator determinante, pode contribuir para melhores resultados em cursos introdutórios de programação. Em relação às habilidades, Ambrósio et al. (2011) citam como habilidade cognitiva a "capacidade de sair de uma análise mais holística dos problemas do quotidiano e fazer uma leitura mais analítica dos mesmos, realizando inclusive um planejamento sequencial" (p. 191). Outra habilidade cognitiva citada pelos mesmos autores é a capacidade de abstração, de forma que "espera-se que os alunos passem do mundo concreto para uma outra realidade mais semântica e simbólica, formulando ou representando um problema de forma mais abstrata e menos aos pormenores e elementos singulares concretos" (p. 191).

Ainda em sua pesquisa, Ambrósio et al. (2011) entrevistaram professores a fim de explicitar as competências necessárias para aprender a programar. Em suas respostas, os professores levam a crer que a habilidade matemática é importante para o sucesso da programação, de forma que ao conseguir

desenvolver uma lógica matemática, o aluno adquire estruturas ou habilidades cognitivas que podem facilitar ou promover a aprendizagem da programação (Ambrósio et al., 2011).

De acordo com Araujo et al. (2019) e ACM & IEEE (2020), o ensino de programação é associado ao de algoritmos, isto porque, para aprender a programar, o aluno precisa dominar a forma e organização lógica que possam ser executadas pelo computador, ou seja, pensar algoritmicamente. Segundo Silva et al. (2015), há três tipos de habilidades que o aluno deve ter nas disciplinas introdutórias de programação: cognitiva, emocional e social, as quais estão relacionadas com o conceito de atitude. A habilidade cognitiva diz respeito a: resolver problemas, estabelecer conclusões lógicas, planejar e tomar decisões. As habilidades sociais referem-se a: lidar com regras, cooperar e elaborar. E as emocionais são: autoconfiança, autoestima e autoavaliação. Além destas, são citadas competências tais como criatividade, estruturação do pensamento, responsabilidade, curiosidade e trabalho em equipe.

De acordo com Villalobos et al. (2009), habilidades de programação são habilidades que os alunos devem desenvolver para aplicar seus conhecimentos com eficácia e prontidão para resolver problemas. O desenvolvimento de tais habilidades requer reforço iterativo para ser integrado de forma incremental às capacidades dos alunos, no entanto, elas podem ser desenvolvidas em qualquer ordem. Assim, os autores apresentam como habilidades para desenvolver um programa, dado um problema: entender o problema, abstração, decomposição, modelagem, análise das abstrações, desenvolver soluções para o problema, escrever as soluções usando uma sintaxe, construir as soluções usando uma tecnologia e avaliar as soluções desenvolvidas, gerando assim soluções concretas (Villalobos et al., 2009).

Em sua tese de doutorado, A. J. Gomes (2010) elenca, como habilidades necessárias para a compreensão e aprendizagem adequada dos conteúdos existentes nas UCs de programação, a interpretação, o raciocínio lógico, a abstração e a resolução de problemas. E. O. da Silva & Falcão (2020) enumeram, como habilidades para aprender programação, o raciocínio lógico, a abstração, noções matemáticas e a resolução de problemas. Além destas, uma habilidade que vem ganhando atenção em pesquisas que buscam amenizar as dificuldades de aprendizagem de programação é o Pensamento Computacional (E. O. da Silva & Falcão, 2020; Wing, 2014).

O desenvolvimento de programas de computador envolve tarefas que exigem a habilidade de resolver problemas escrevendo uma sequência de passos (Franzen et al., 2018). Em relação à aprendizagem de uma linguagem de programação, Medeiros (2019) pontua que os alunos necessitam: entender os conceitos e paradigmas de linguagens de programação, entender a semântica formal e

compreender a teoria dos tipos, polimorfismo, verificação e inferência de tipos. De forma geral, Medeiros (2019) cita que, dentre as habilidades definidas pela Sociedade Brasileira de Computação (SBC) para cursos superiores na área de Computação, três são específicas para disciplinas de programação: identificar problemas que têm uma resolução algorítmica; resolver problemas usando um ambiente de programação; e compreender e explicar as dimensões quantitativas de um problema. Dessa forma, é necessário o exercício constante do raciocínio lógico e da capacidade de abstração para resolver problemas – interpretar, analisar requisitos e apresentar soluções algorítmicas. Estas, por sua vez, exigem a compreensão de conceitos como estruturas de controle, recursão e ponteiros, por exemplo (G. Santos et al., 2020). Logo, saber programar, entender estruturas computacionais e raciocinar logicamente são competências importantes para os alunos desses cursos (Araujo et al., 2019). De acordo com Medeiros et al. (2020), as UCs de programação geralmente exigem

> habilidades de resolução de problemas, conceitos básicos de programação, sintaxe e semântica de linguagens de programação e o uso destas linguagens para formular soluções. As disciplinas são introdutórias nos cursos e basilares para o entendimento dos conceitos e fundamentações necessárias para outras disciplinas mais avançadas. (p. 186)

Categorizamos os conhecimentos habilidades e competências apresentadas pelos autores de acordo com as seguintes categorias:

a) Background: indica questões inerentes à formação do aluno que são anteriores ao seu acesso ao Ensino Superior.
b) Conhecimento específico da programação: dificuldades em assuntos específicos da programação.
c) Conhecimentos gerais: dificuldades em assuntos gerais, não apenas da programação, mas de quaisquer outras áreas do conhecimento.
d) Fator pessoal: fatores inerentes ao aluno, como questões pessoais, familiares e socioeconômicas.
e) Habilidade: capacidade de aplicar o conhecimento para realizar ativamente uma tarefa.

A Tabela 4 apresenta a categorização e conhecimentos, habilidades e competências necessários para aprender programação citadas ao longo deste tópico com seus respectivos autores.

Tabela 4 - *Conhecimentos, habilidades e competências para aprender a programar encontrados na literatura*

Categoria	**Conhecimentos, habilidades e competências**	**Autores FT***	**Autores RSL****
Background	Compreensão leitora	(Ambrósio et al., 2011)	
Background	Conhecimento em ciências	(Ambrósio et al., 2011)	
Background	Conhecimento em matemática	(Ambrósio et al., 2011; E. O. da Silva & Falcão, 2020)	
Background	Conhecimento prévio em computação	(Ambrósio et al., 2011)	
Background	Interpretação	(Gomes, 2010; G. Santos et al., 2020)	
Conhecimento específico da programação	Algoritmia	(Franzen et al., 2018; G. Santos et al., 2020)	
Conhecimento específico da programação	Análise de requisitos	(Santos et al., 2020)	
Conhecimento específico da programação	Compreensão da sintaxe e estrutura do programa		(Aparicio & Costa, 2018; Chang et al., 2000; Costa, 2019; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015; Ortiz-Ortiz et al., 2018; A. Santos et al., 2010, 2013; Skalka & Drlik, 2018;

			Souleiman, 2017)
Conhecimento específico da programação	Conhecer ferramentas (IDEs)		(T. A. N. de Oliveira & Rebouças, 2018; Souleiman, 2017)
Conhecimento específico da programação	Construir soluções usando tecnologia	(Villalobos et al., 2009)	
Conhecimento específico da programação	Elaboração de códigos em uma linguagem de programação ou pseudocódigo	(Araujo et al., 2019; Dijkstra, 1989; Medeiros et al., 2020; Santos et al., 2020; Villalobos et al., 2009)	
Conhecimento específico da programação	Estrutura de controle	(Santos et al., 2020)	
Conhecimento específico da programação	Identificar e corrigir erros nos códigos	(Dijkstra, 1989)	
Conhecimento específico da programação	Linguagem de programação	(Dijkstra, 1989; Medeiros, 2019; Medeiros et al., 2020; Villalobos et al., 2009)	
Conhecimento específico da programação	Modelagem	(Dijkstra, 1989; Villalobos et al., 2009)	
Conhecimento específico da programação	Paradigmas de linguagem programação	(Medeiros, 2019)	
Conhecimento específico da programação	Pensamento computacional	(Silva & Falcão, 2020; Wing, 2014)	(Al-Imamy et al., 2006; Aparicio & Costa, 2018; T. Castro et al., 2011;

			R. M. S. Pereira, 2017; Rezende & Bispo, 2018)
Conhecimento específico da programação	Pensar algoritmicamente	(ACM & IEEE, 2020; Araujo et al., 2019)	
Conhecimento específico da programação	Polimorfismo	(Medeiros, 2019)	
Conhecimento específico da programação	Ponteiros	(Santos et al., 2020)	
Conhecimento específico da programação	Recursividade	(Santos et al., 2020)	
Conhecimento específico da programação	Teoria dos tipos	(Medeiros, 2019)	
Conhecimento específico da programação	Semântica formal	(Medeiros, 2019; Medeiros et al., 2020)	
Conhecimentos gerais	Avaliação da solução	(Villalobos et al., 2009)	
Conhecimentos gerais	Pensamento sistêmico	(Ambrósio et al., 2011)	
Fator pessoal	Autoavaliação	(Silva et al., 2015)	
Fator pessoal	Autoconfiança	(Silva et al., 2015)	
Fator pessoal	Autoestima	(Silva et al., 2015)	
Fator pessoal	Autonomia		(Al-Imamy et al., 2006;

			Chuchulashvili et al., 2016)
Fator pessoal	Cooperar	(T. R. da Silva et al., 2015)	
Fator pessoal	Curiosidade intelectual	(Ambrósio et al., 2011; T. R. da Silva et al., 2015)	
Fator pessoal	Lidar com regras	(Silva et al., 2015)	
Fator pessoal	Planejamento	(Ambrósio et al., 2011; A. S. Oliveira et al., 2015; T. R. da Silva et al., 2015)	(Al-Imamy et al., 2006; Skalka & Drlik, 2018)
Fator pessoal	Proatividade		(Chuchulashvili et al., 2016; Mendes et al., 2012)
Fator pessoal	Responsabilidade	(Silva et al., 2015)	
Fator pessoal	Trabalhar em equipe	(Silva et al., 2015)	
Habilidade	Abstração	(Ambrósio et al., 2011; Dijkstra, 1989; Gomes, 2010; Santos et al., 2015; Santos et al., 2020; E. O. da Silva & Falcão, 2020; da Silva et al., 2015; Villalobos et al., 2009)	
Habilidade	Compreender as dimensões quantitativas de um problema	(Medeiros, 2019)	
Habilidade	Criatividade	(Ambrósio et al., 2011; Oliveira et al., 2015; Silva et al., 2015)	
Habilidade	Decompor o problema	(Villalobos et al., 2009)	

Habilidade	Estabelecer conclusões lógicas	(Silva et al., 2015)	
Habilidade	Estruturação do pensamento	(Silva et al., 2015)	
Habilidade	Habilidades matemáticas	(Ambrósio et al., 2011)	
Habilidade	Identificação de problemas	(Ambrósio et al., 2011; A. de J. Gomes, 2010; MEDEIROS, 2019; Villalobos et al., 2009) (Dijkstra, 1989)	
Habilidade	Raciocínio lógico	(Ambrósio et al., 2011; Araujo et al., 2019; Gomes, 2010; Oliveira et al., 2015; Santos et al., 2015; Santos et al., 2020)	(Al-Imamy et al., 2006; Bosse & Gerosa, 2017; T. Castro et al., 2011; T. H. Castro et al., 2008; Costa, 2019; T. A. N. de Oliveira & Rebouças, 2018; Ortiz-Ortiz et al., 2018; Rezende & Bispo, 2018; Skalka & Drlik, 2018)
Habilidade	Resolução de problema	(Ambrósio et al., 2011; Dijkstra, 1989; Gomes, 2010; Medeiros, 2019; Medeiros et al., 2020; Oliveira et al., 2015; da Silva & Falcão, 2020; da Silva et al., 2015; Villalobos et al., 2009)	(Aparicio & Costa, 2018; Bosse & Gerosa, 2017; T. Castro et al., 2011; T. H. Castro et al., 2008; Chang et al., 2000; Chuchulashvili et

		al., 2016; Costa, 2019; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015; Nawahdah et al., 2015; T. A. N. de Oliveira & Rebouças, 2018; Ortiz-Ortiz et al., 2018; Rezende & Bispo, 2018; A. Santos et al., 2010, 2011, 2013; Skalka & Drlik, 2018; Souleiman, 2017)
Habilidade	Tomar decisões	(Silva et al., 2015)

Nota: *Fundamentação teórica. **Revisão Sistemática de Literatura. Autoria própria.

A Tabela 4 nos ajuda a perceber de forma sintética as habilidades, competências e conhecimentos necessários para a aprendizagem de programação que estão apresentados na literatura. Conhecer esses aspectos nos ajuda a compreender o que é necessário explorar durante o processo de ensino para que os alunos possam ter melhor entendimento e consigam ter maior facilidade em aprender a programar. Não queremos dizer que o aluno precisar possuir todos os conhecimentos, habilidades e competências citados nesse tópico, e nem que uma isolada é capaz de proporcionar ao aluno aprender a programar, uma vez que cada aluno possui seu perfil, seu nível de conhecimento e seu ritmo de aprendizagem. Entretanto, percebemos que ao aliar essas competências, conhecimentos e habilidades, o aluno poderá ter maior índice de êxito em seu processo de aprendizagem, e o professor, por sua vez, poderá ter maior facilidade em ensinar aos alunos ao explorar e enfatizar essas características neles.

2.2.3. Metodologias e Ferramentas

Diante dos conhecimentos, habilidades e competências necessários para se aprender a programar, e tendo em vista as dificuldades apontadas por professores e alunos durante o processo de ensino e aprendizagem de programação, muitos são os esforços e pesquisas desenvolvidas para

criação de metodologias, estratégias e ferramentas que possibilitem a facilitação deste processo. Neste tópico inventariaremos o que a literatura aborda sobre estes três aspectos. Para melhor apresentação, será dividido em duas subseções: 'metodologias para o ensino e aprendizagem de programação' e 'ferramentas utilizadas no processo de ensino e aprendizagem de programação'.

2.2.3.1. Metodologias para o Ensino e Aprendizagem de Programação. Devido à sua natureza complexa, o processo de ensino e aprendizagem de programação exige muito esforço por parte dos docentes em busca de estratégias e metodologias diversificadas que auxiliem na efetividade da aprendizagem, bem como uma grande dedicação por parte do estudante para aquisição dos conhecimentos e da habilidade prática necessária (Coutinho et al., 2018).

Em busca de promover maior envolvimento dos estudantes, educadores buscam experimentar diferentes metodologias, técnicas e ferramentas para apoiar o ensino-aprendizagem de programação. No contexto brasileiro, o Parecer no Conselho Nacional de Educação/Câmara de Educação Superior (CNE/CES nº 136/2012), orienta que nos cursos de graduação em computação,

> A metodologia de ensino deve ser centrada no aluno como sujeito da aprendizagem e apoiada no professor como facilitador do processo de ensino e aprendizagem. O professor deve fortalecer o trabalho extraclasse como forma de o aluno aprender a resolver problemas, aprender a aprender, tornar-se independente e criativo. O professor deve mostrar, ainda, as aplicações dos conteúdos teóricos, ser um mediador, estimular a competição, a comunicação, provocar a realização de trabalho em equipe, motivar os alunos para os estudos e orientar o raciocínio e desenvolver as capacidades de comunicação e de negociação. O projeto pedagógico deve prever o emprego de metodologias de ensino e aprendizagem que promovam a explicitação das relações entre os conteúdos abordados e competências previstas para o egresso. A metodologia de ensino deve desenvolver uma visão sistêmica para resolução de problemas. (p. 4)

Neste sentido, alternativas encontradas para tentar descomplicar o aprendizado de programação pauta-se na criação de alguns métodos de ensino e ferramentas específicas (Medeiros, 2019). Entretanto, para que estas alternativas sejam eficazes é necessário "entender de forma mais detalhada e completa quais são os problemas no ensino e aprendizagem de introdução à programação para aprimorar e criar novos métodos e ferramentas mais eficazes" (Medeiros, 2019, p. 32). O ensino de programação, na maioria dos casos, se dá pelo desenvolvimento de atividades práticas com problemas a serem resolvidos pelo aluno (Iepsen et al., 2013). Ensinar dessa forma estimula para que,

antes de partir para a codificação em uma linguagem de programação, o estudante compreenda o problema e defina uma estrutura lógica da solução.

Em sua pesquisa, T. R. da Silva et al. (2015) apresentam algumas metodologias de ensino, oriundas de uma revisão sistemática de literatura, e que são usadas no contexto do ensino de programação, tais como Aprendizagem Significativa, Pedagogia do Construtivismo, Taxonomia de Bloom, Taxonomia revisada de Bloom, Abordagem Construcionista, Abordagem Instrucionista e Ciclo de Kolb (p. 188).

Para além das abordagens apresentadas em Silva et al. (2015), outras metodologias são encontradas na literatura, sendo que nas pesquisas realizadas as que mais se destacam são as metodologias ativas. Paillard & Moreira (2017) citam que as metodologias de ensino ativas, como a utilização de problemas, têm sido utilizadas com o objetivo de estimular o envolvimento, colocar o discente como sujeito fundamental no processo de ensino e aprendizagem, de forma que entre as abordagens é possível destacar pesquisas sobre a influência de diferentes paradigmas ou linguagens de programação. A aprendizagem ativa visa fazer com que os estudantes se tornem ativamente envolvidos no seu próprio aprendizado, buscando ampliar suas próprias descobertas (G. Alves et al., 2019). Em sua pesquisa, Alves et al. (2019) utilizaram uma técnica conhecida como *Coding Dojo* (CD) para auxiliar a implementação de práticas de aprendizagem ativa.

Em sua pesquisa, J. Silva et al. (2018) propõem um modelo conceitual que facilita o acompanhamento do professor, oferecendo ainda uma personalização do ensino, de acordo com o perfil de aprendizado do aluno, tendo como base o domínio cognitivo da Taxonomia de Bloom, com elementos de gamificação. Figueiredo (2015) também utiliza a gamificação como metodologia para ensinar e aprender programação, com ênfase na cooperação entre os alunos. Netto et al. (2017), por sua vez, propõem a utilização da gamificação para estimular o raciocínio lógico dos alunos. Já Silva et al. (2018) associam a gamificação a narrativas contextualizadas ao assunto abordado no ensino de programação, de forma que, como resultado, obtiveram uma maior motivação dos alunos para aprender a programar, ativada pela curiosidade atrelada ao enredo do *storytteling*.

A pesquisa desenvolvida por Farias & Nunes (2019) trata-se de uma revisão sistemática de literatura acerca da aprendizagem ativa no ensino de programação. Segundo os autores,

> o uso de metodologias ativas de ensino no contexto da aprendizagem de programação ainda surge como obscuro em muitas realidades, principalmente por ainda ocorrer conflitos na postura epistemológica do professor e do estudante, bem como pela adoção de métodos conservadores de avaliação centrados na resolução exaustiva de exercícios por parte do

> aprendiz que são frágeis quando se observa o desenvolvimento holístico das habilidades relacionadas aos pilares do pensamento computacional premissa ao aprendizado de programação. (p. 378)

Como resultado da pesquisa, Farias & Nunes (2019) identificaram as seguintes metodologias ativas: aprendizagem baseada em desafios, aprendizagem baseada em questionários, aprendizagem baseada em projeto e problema, aprendizagem móvel, aprendizagem híbrida e aprendizagem baseada em jogos.

Em sua pesquisa, Silveira et al. (2018) apontam a sala de aula invertida (*Flipped Classroom*) como uma metodologia ativa que pode ser usada no processo de ensino e aprendizagem de programação. Outra metodologia citada pelos autores é a aprendizagem baseada em problemas (*Problem Based Learning* - PBL), em que os alunos podem ser desafiados a resolver problemas, usando para isso seus conhecimentos na área de lógica de programação. Silveira et al. (2018) acreditam que "a aplicação de metodologias ativas de aprendizagem, que permitam aprimorar o trabalho docente, possa auxiliar o professor em suas atividades e beneficiar os alunos, aumentando a interação e a possibilidade de aprendizagem" (p. 28). Além disso, essas metodologias colocam os alunos como sujeitos ativos, sendo baseadas na teoria construtivista de Piaget.

Em busca de melhores resultados e aliadas a metodologias de ensino e aprendizado de programação, diversas são as estratégias que são propostas na literatura e utilizadas na prática. Se ater a estratégias é importante por que preocupar-se apenas com transmissão de conhecimento não é suficiente para o ensino de programação (Law et al., 2010). De acordo com Oliveira et al. (2015), estudos relatam que é preciso identificar um estilo correto de ensino e aprendizagem de programação, procurar por abordagens de Tecnologia da Informação que aproximem aluno e professor, para motivar os alunos para adquirir as habilidades de programação com rapidez e mais facilidade, gerando assim o interesse deles pelo conteúdo (Law et al., 2010).

Em seu estudo, Jenkins (2001) aponta que a melhor maneira de aprender a programar será diferente para pessoas diferentes, mas, qualquer que seja a forma adotada de estudar, deve certamente envolver estas três estratégias:

- Prática: a programação é uma habilidade e a única maneira de adquirir e desenvolver tal habilidade é praticando.
- Uso de material de referência: mesmo programadores experientes trabalharão com uma referência de linguagem ao seu lado.

- Recorrer a aulas de programação: uma maneira eficaz de adquirir uma habilidade é por um aprendizado.

Pedrosa (2017), em seu trabalho doutoral, recorre a alguns pesquisadores para apresentar diferentes metodologias para o contexto de ensino e aprendizagem de programação. A autora cita Mohorovičić & Strčić (2011) ao indicar que é importante selecionar um conjunto de estratégias de ensino adequadas que proporcionem aos alunos um ambiente de aprendizagem mais eficiente, e recomenda a concepção e implementação de vários métodos de ensino inovadores para melhorarem o ensino e a aprendizagem da programação. Ao citar Sarpong, Arthur & Amoako (2013), Pedrosa (2017) aborda estratégias de ensino que dão aos alunos melhores oportunidades para interagir com seus colegas e com os professores, tais como: tutoria em pares, programação em pares/grupo e resolução de problemas. Além disso, Pedrosa (2017) apresenta em sua tese de doutoramento a abordagem *SimProgramming*.

> A abordagem *SimProgramming* baseia-se em quatro fundamentos conceptuais [...]1) Ambiente de aprendizagem de simulação empresarial (Business-like), 2) Autorregulação das aprendizagens; 3) Corregulação da aprendizagem, e 4) Avaliação formativa. Com base nestes fundamentos, o processo da atividade de aprendizagem desenvolve-se ao longo de quatro fases e os alunos têm tarefas específicas em cada uma dessas fases. (p. 80)

De acordo com a autora, a *SimProgramming* apresenta-se como uma abordagem promissora para a aprendizagem do aluno e no desenvolvimento de competências de programação e interpessoais, além de ajudar a melhorar as estratégias de autorregulação e de corregulação de aprendizagem dos alunos.

Dentro do contexto de aprendizagem colaborativa, existem iniciativas como o trabalho de Adán-Coello et al. (2008), que aborda a dificuldade do professor em dividir os alunos em grupos de trabalho. Segundo os autores, para que se possa garantir a colaboração entre os membros de um grupo, a formação dos grupos não deve ser aleatória e, dessa forma, os autores propõem e desenvolvem ferramentas para a avaliação dos programas criados pelos alunos e para organização dos grupos baseado nos estilos de aprendizagem de cada aluno.

Uma outra estratégia para o ensino e aprendizagem de programação é a criação de Comunidade de Prática (CdP), proposta por Chen et al. (2012), na qual os estudantes são divididos em grupos para o desenvolvimento de atividades, cada um com papeis bem definidos, tais como gerente de projetos, líder de equipe, aprendiz e programador, e que simulam um ambiente corporativo. Esteves et al. (2008) propuseram uma CdP no mundo virtual *Second Life* em que os estudantes eram

estimulados a desenvolver projetos em duplas dentro do ambiente *Second Life*, que podiam interagir e trocar experiências para atingir o objetivo proposto. Verificou-se que os estudantes gostaram da prática, sobretudo pelo fato de o mundo virtual permitir que visualizassem de forma concreta o produto de sua codificação.

Partindo do contexto da CdP, e tendo em vista que no contexto atual deve-se haver uma forte ligação entre a teoria e a prática, a atuação docente precisa estar sintonizada com o mundo do trabalho, aliando os estudos acadêmicos com as tecnologias utilizadas atualmente. Assim, a adoção de atividades interdisciplinares é uma estratégia que permite mostrar aos alunos que os assuntos da UC estão relacionados a outros temas da área. Logo, é preciso apresentar aos alunos em que contexto os conteúdos da UC poderão ser aplicados na vida profissional e como esta UC se relaciona com as demais do currículo do curso (Silveira et al., 2018).

As pesquisas desenvolvidas por von Wangenheim et al. (2014) e Arantes & Ribeiro (2017) estão focadas na melhoria do processo de aprendizagem do aluno quando sugerem estratégias para o ensino de programação de forma interdisciplinar, com o uso da ferramenta Scratch. A proposta destes autores é contribuir para a formação de um pensamento computacional. Na literatura, é possível encontrar muitas linguagens de programação desenvolvidas cuja principal finalidade é facilitar o acesso de iniciantes ao conhecimento de programação. É o caso, por exemplo das linguagens: Logo (Papert, 2008), Pascal (Pacitti et al., 1983), Portugol (Manso et al., 2009), e as linguagens de programação visual (Hansen et al., 2000; Hundhausen et al., 2002). Estudos foram feitos para verificar se, efetivamente, o uso da animação facilita a aprendizagem de algoritmos. A animação de algoritmos pode ajudar os alunos a entender os programas, pode facilitar a análise de programas existentes e pode mesmo facilitar o desenvolvimento de novos programas.

De acordo com Tavares (2018), "é muito importante dar aos alunos a oportunidade de praticar e resolver exercícios de programação por si mesmos desde o primeiro contato com as disciplinas de programação" (p. 4), entretanto é essencial o acompanhamento por parte do docente, fornecendo aos seus alunos o apoio e *feedback* imediato. Segundo a autora,

> O *feedback* imediato é importante para incentivar o aluno a progredir dando-lhe uma avaliação do trabalho em curso, indicando se o resultado que o seu programa está a produzir está certo ou errado e, se possível, explicando as causas do erro ou pelo menos descrevendo as situações em que o programa falha. (p. 4)

Outra estratégia utilizada no processo de ensino e aprendizagem de programação é o uso da robótica. Pesquisadores afirmam que a Robótica Educativa é um importante instrumento para o

desenvolvimento de habilidades que necessitam de pensamento lógico e abstrato (Pio et al., 2006). De acordo com estes autores, a robótica permite o aprimoramento do pensamento lógico, melhora na compreensão de conceitos computacionais abstratos e um aumento da motivação para realização das atividades propostas.

Soluções lúdicas, como a utilização de linguagem de programação visual, contribuem para facilitar a aprendizagem dos alunos à medida que os auxiliam com resoluções de problemas. Assim como a robótica, o uso de jogos é uma estratégia que motiva e promove o engajamento dos alunos na aprendizagem de programação. De acordo com Marques et al. (2011), os jogos, além de serem cada vez mais acessíveis, tanto para jogar quanto para desenvolver, muitas vezes são também utilizados para atrair alunos para os cursos de Ciência da Computação ou áreas afins.

Em sua pesquisa, G. Santos et al. (2020) realizaram um mapeamento sistemático da literatura sobre metodologias de ensino e aprendizagem de programação. Em seus resultados, os autores apresentam as seguintes estratégias: Computação plugada/desplugada, tendo como base a teoria do Construcionismo; uso da robótica, fazendo uso de dispositivos para a construção de conhecimentos; uso de jogos, tanto para a aprendizagem com jogos quanto para a aprendizagem por meio do desenvolvimento de jogos; gamificação, utilizando elementos de design de jogos para motivar e aumentar a a atenção do aluno; e a valorização de aspectos comportamentais, para realização do planejamento das atividades a serem desenvolvidas na disciplina. No trabalho de O'Kelly & Gibson (2006), é apresentada a PBL aplicada ao ensino de programação. De acordo com os autores, para funcionar efetivamente, a PBL necessita de um ambiente com um bom conjunto de problemas, para tornar o contexto de ensino e aprendizagem rico e dinâmico.

Tradicionalmente, as aulas de programação são dadas em uma sala de aula, em que o professor apresenta os conceitos de sintaxe, lógica, e análise de código, por meio de leitura e discussão, sendo que a estratégia de ensino adotada pelo docente é determinante no processo da aprendizagem de programação (Bosse & Gerosa, 2017; Ortiz-Ortiz et al., 2018; Rezende & Bispo, 2018; Shadiev et al., 2013). De acordo com Hwang et al. (2012), ensinar programação apenas (ou em maior carga) de forma expositiva muitas vezes limita a eficácia da aprendizagem, uma vez que a oportunidade de o aluno praticar a programação se torna limitada, e os professores acabam não tendo certeza se todos os alunos estão adaptados a este contexto de aprendizagem. Outra estratégia adotada no meio acadêmico consiste na exposição de exemplos para orientação de resolução de problemas básicos, cujas soluções são apresentadas por meio de algoritmos (Ortiz-Ortiz et al., 2018). Embora os

alunos percebam os algoritmos no momento em que são apresentados pelos professores, na maioria das vezes são incapazes de deduzir as soluções por si só (Ortiz-Ortiz et al., 2018).

A capacidade de escrever um programa se torna um dos principais requisitos dos alunos em todos os níveis educacionais, mas muitos sistemas educacionais não são capazes de satisfazer suas necessidades a uma taxa suficiente. De acordo com Skalka & Drlik (2018), esse problema pode ser resolvido aumentando o interesse dos alunos em programas de estudo em TI ou diminuindo também o número de alunos que saem do estudo cedo devido ao fracasso da aprendizagem em alguns cursos.

Gomes & Mendes (2015) afirmam que as estratégias em sala de aula adotadas pelos professores de programação, geralmente, seguem uma abordagem bem semelhante, de forma a apresentar e explicar os detalhes sintáticos das linguagens de programação, complementando que "a apresentação de cada conceito é geralmente seguida da explicação de pequenos exemplos mostrando certos aspectos da sintaxe que levam a resultados corretos, incorretos ou inesperados" (p. 3, tradução nossa). Ao entrevistar alguns professores, os autores perceberam as mais diversas estratégias de ensino de programação adotadas, com a finalidade de promover melhorias na aprendizagem dos alunos, como, por exemplo:

- separar as aulas por assuntos, e nessas aulas, apresentar exercícios apenas relacionados ao assunto visto;
- trabalhar em pequenos grupos, e aumentar o tempo disponível aos alunos, rentabilizando o tempo de sala de aula, o que permitiu conhecer melhor as dificuldades e preferências de cada aluno;
- apresentar, nas aulas práticas, exercícios resolvidos, para depois introduzir a temática daquele assunto;
- fornecer muitos exemplos e exercícios aos alunos, de forma que eles realizem a prática de programação, embora alguns professores defendam que o número de exercícios dados em sala de aula vai depender do assunto abordado e do ritmo dos alunos.

Além disso, um dos aspectos percebidos pelos autores, em relação às informações dadas pelos professores é de que a aproximação do professor é fundamental para motivar os estudantes e incentivá-los a continuar trabalhando até atingir um bom nível de programação.

A estratégia proposta por Hwang et al. (2012) consiste na aprendizagem cooperativa. De acordo com Chiu (2008) (como citado em Hwang et al., 2012) "o aprendizado cooperativo oferece muitos benefícios potenciais além do aprendizado de programação, ou seja, os alunos capitalizam os recursos e habilidades uns dos outros" (p. 5, tradução nossa), o que permite maior interação entre os

alunos, de forma que possam avaliar as ideias uns dos outros, além de um aluno monitorar o trabalho do outro.

Desta forma, a cooperação motiva os alunos a participarem ativamente da aprendizagem de programação, sendo a motivação considerada um fator importante no processo de aprendizagem de programação (Bosse & Gerosa, 2017; A. Gomes & Mendes, 2015, 2010; Hwang et al., 2012; Mendes et al., 2012; A. Santos et al., 2013; Souleiman, 2017). Além disso, durante a aprendizagem cooperativa, os alunos dividem as tarefas, trabalham em conjunto, apoiam-se, aprendem uns com os outros e podem compartilhar experiências para que possam alcançar os objetivos de aprendizagem (Hwang et al., 2012). De forma a verificar a potencialidade, de forma prática, da aprendizagem cooperativa de programação, Hwang et al (2012) propuseram um sistema *web* de assistência à aprendizagem de programação por cooperação denominado WPASC (*Web-Based Programming Assisted System For Cooperation*), em que foram desenvolvidas atividades de aprendizagem para facilitar a aprendizagem cooperativa de programação.

Frente às dificuldades de aprendizagem de programação, Santos et al. (2013) propuseram uma taxonomia de exercícios para apoiar no processo de aprendizagem de programação. De acordo com os autores, taxonomias com objetivos educacionais são utilizadas para descrever os resultados da aprendizagem bem como da avaliação, para que possa refletir um estágio de aprendizagem do aluno, de forma que, na literatura, autores dividem os objetivos educacionais em três domínios: afetivo, cognitivo e psicomotor, porém "a pesquisa existente sobre o uso de taxonomias de aprendizagem em Ciência da Computação concentra-se no domínio cognitivo" (p. 2, tradução nossa). A taxonomia desenvolvida mapeia cada exercício de programação em uma ou mais dimensões, levando em consideração aspetos diferentes que consideramos relevantes para a classificação de exercícios. Algumas das dimensões utilizadas para o desenvolvimento da taxonomia proposta foram os níveis de Taxonomia de Bloom, o conhecimento matemático necessário para desenvolver cada exercício, a complexidade das estruturas de controle envolvidas na solução (quando o caso), a complexidade do exercício e a complexidade da solução algorítmica. De acordo com os autores, a taxonomia proposta permite "uma classificação adequada de exercícios introdutórios de programação. Isso deve facilitar a seleção de um exercício apropriado a ser proposto a um aluno em particular em um momento específico" (A. Santos et al., 2013, p. 3, tradução nossa). Assim como Santos et al. (2013), Pereira et al. (2017) e Chuchulashvili et al. (2016) versam sobre a necessidade de adotar estratégias de aprendizagem que se adaptem ao ritmo de aprendizagem do aluno, em Mendes et al. (2012) é citado que as metodologias devem ser voltadas a necessidades específicas de cada aluno.

A Taxonomia proposta por Santos et al. (2013) é dividida em três dimensões: tópicos, complexidade e níveis. Os tópicos representam os conteúdos apresentados, a complexidade diz respeito ao grau de dificuldade de cada exercício, e o nível diz respeito aos níveis da Taxonomia de Bloom, ou ainda ao nível de cada exercício (se inicial, intermediário, ou avançado, por exemplo).

Para avaliação e validação da proposta, os autores experimentaram a taxonomia com 44 alunos em uma disciplina de programação de um curso superior de Ciência da Computação. O experimento foi dividido em cinco etapas e, ao final, a maioria dos estudantes conseguiram resolver os exercícios propostos como esperado. Ao longo do experimento, e como resultado, verificou-se que a taxonomia "também pode ajudar a promover a motivação dos alunos, criando um equilíbrio adequado entre os sucessos ('estímulo motivacional') e falhas ('erros positivos') essenciais para uma aprendizagem consolidada" (A. Santos et al., 2013, p. 7, tradução nossa).

Tendo em vista as dificuldades dos alunos em aprender a programar, o que leva à necessidade de desenvolvimento de novas técnicas e metodologias, Aparicio & Costa (2018) propuseram uma solução baseada em robôs virtuais como uma solução para capacitar alunos com uma ferramenta digital que proporciona aos alunos visualizar, de forma imediata, o resultado de seu esforço na programação. De acordo com os autores, a utilização de robôs parece continuar sendo uma solução para apoiar o processo de aprendizagem da programação de computadores. Um robô virtual oferece um ambiente controlado, tendo o poder de focar o estudo na programação em vez de na complexidade mecânica e eletrônica. No entanto, eles têm algumas limitações, uma vez que não possuem a tangibilidade dos robôs reais (Aparicio & Costa, 2018). Ao longo do artigo, os autores apresentam uma visão geral do robô proposto, como foi implementado, bem como os conceitos de programação suportados por ele para proporcionar a aprendizagem de programação.

O trabalho realizado por Ortiz-Ortiz et al. (2018) apresenta a plataforma BASICS, a qual adota a estratégia didática de aprendizagem híbrida (*blended learning*), de maneira que os estudantes "recebem aulas em sala de aula e praticam em laboratório, reforçando os conhecimentos com 'BASICS' no seu próprio ritmo, a qualquer hora, em qualquer lugar e com qualquer dispositivo conectado à internet que possua um navegador" (Ortiz-Ortiz et al., 2018, p. 1, tradução nossa). Segundo os autores, o principal objetivo da plataforma é, de fato, melhorar os resultados da aprendizagem de programação, de maneira a atingir uma redução na taxa de reprovação nas disciplinas e a taxa de evasão dos alunos ao longo do curso. Para isso, a plataforma proposta promove o aprendizado autorregulado, desenvolve o pensamento lógico, incorpora desafios baseados em gamificação, fornece *feedback* imediato e permite comunicação síncrona e assíncrona entre aluno e

professor, permitindo um ambiente de trabalho colaborativo. Ao longo do artigo, os autores apresentam uma visão geral da plataforma, como esta foi desenvolvida e como ela pode ser utilizada por professores e alunos. Para sua validação, os autores realizaram um experimento com dois grupos de alunos do primeiro ano de programação, um grupo com 17 e outro com 18 alunos, sendo um grupo de controle e um de experimento que, por sua vez, utilizou adicionalmente a plataforma como recurso de aprendizagem. Como resultado, a plataforma mostrou-se como boa solução no processo de aprendizagem de programação, de forma que os alunos do grupo de experimento obtiveram melhores resultados, em relação ao grupo de controle, sentiram-se motivados e encorajados a utilizar a plataforma e a aprender programação. Além disso, a reprovação na disciplina foi reduzida no grupo de experimento. Embora a plataforma tenha tido uma boa aceitação e avaliação no contexto em que foi validado, de acordo com os autores,

> Na educação mista, mudanças constantes devem ser feitas, todas as atividades devem ser avaliadas a fim de melhorar o que está sendo bem-feito, e o que não funciona deve ser alterado ou removido. O professor deve fazer ajustes na aula de acordo com as características dos alunos, e o BASICS não é exceção. Os professores devem, portanto, avaliar a atividade dos alunos ao usar o sistema, a fim de corrigir, atualizar, excluir, testar e projetar reagentes que irão melhorar o ensino e a aprendizagem. (Ortiz-Ortiz et al., 2018, p. 5, tradução nossa)

A pesquisa desenvolvida por Shadiev et al. (2013) apresenta o método de aprendizagem 'aprender explicando' (*learning by explaining*), que facilita ao aluno explicar um determinado conteúdo para si mesmo ou para outras pessoas. De acordo com os autores, esse método pode levar à construção de conhecimento adicional para e pelo indivíduo explicador, isso porque, para que um estudante consiga explicar um conteúdo, ele precisa estar familiarizado com este, além de pensar em como explicar aos colegas de forma que estes possam entender o conteúdo. Outro conceito abordado por Shadiev et al. (2013) é o 'ensino recíproco' (*reciprocal teaching*), que consiste em um procedimento instrucional que permite que aluno e professor se revezem durante a explicação de um conteúdo. Para isto, os autores elencam quatro estratégias importantes para que o ensino recíproco ocorra:

- fazer um resumo do conteúdo de aprendizagem;
- compor perguntas sobre o conteúdo que está sendo abordado ou sobre partes do conteúdo que o aluno ainda não conseguiu entender;
- esclarecer dúvidas ou partes complexas do assunto; e
- prever o próximo conteúdo a ser visto.

A aprendizagem por explicação bem como o ensino recíproco facilitam a interação entre alunos e colegas bem como entre alunos e professores. De acordo com Shadiev et al. (2013),

> As estratégias de ensino recíproco facilitam a interação dos alunos com os colegas explicando, questionando e esclarecendo sobre as dificuldades vivenciadas por eles, códigos de programa, métodos aplicados etc. Portanto, as estratégias de ensino recíproco permitem aos alunos trabalhar em cooperação em tarefas cognitivas complexas e melhorar suas habilidades de resolução de problemas e desempenho de aprendizagem. (p. 1, tradução nossa)

Para apresentar a proposta, os autores desenvolveram um experimento com alunos que estudaram programação e alunos iniciantes em programação, propondo duas estratégias de ensino: uma unidirecional e outra recíproca, com o intuito de comparar as duas estratégias. Para apoiar os alunos durante o experimento, os autores propuseram um sistema de aprendizagem baseada na *web* chamado VPen. Participaram do experimento 46 alunos, os quais foram divididos em dois grupos: um de controle e outro de experimento. O experimento foi dividido em duas fases e teve duração de três meses. Durante o experimento, os alunos utilizaram a ferramenta VPen e as quatro estratégias de aprendizagem recíproca apresentadas anteriormente. Para verificar os efeitos das duas abordagens no processo de ensino e de aprendizagem dos alunos envolvidos, os pesquisadores desenvolveram critérios de avaliação baseados na Taxonomia de Bloom, por meio de questionários iguais para ambos os grupos. Também foi desenvolvido um estudo para se saber a percepção e opinião dos alunos em relação à ferramenta proposta.

Como resultados da pesquisa, a maioria dos estudantes relataram que as estratégias unidirecionais e recíprocas foram úteis para o processo de aprendizado, assim como destacaram a facilidade e utilidade da ferramenta VPen. Percebeu-se na análise estatística desenvolvida pelos autores que houve significativa diferença entre os grupos experimental e de controle, em relação ao desenvolvimento de programas e ao nível de cognição dos alunos em relação à programação nas duas fases do experimento. Com base nos resultados deste estudo, a aplicação de estratégia de ensino recíproco facilita o aprendizado de programação, particularmente de programadores iniciantes, uma vez que esta permite maior interação entre os alunos.

O trabalho desenvolvido por Eltegani & Butgereit (2015) apresenta uma investigação sobre os atributos que influenciam no envolvimento dos alunos no processo de aprendizagem de programação. Ao longo do artigo, os autores evocam outras pesquisas que relatam as principais dificuldades apresentadas pelos alunos durante a aprendizagem de programação; apresentam algumas estratégias de ensino e aprendizagem, e citam algumas ferramentas de apoio a esses processos. De acordo com

os autores, para melhorar a forma de ensinar aos alunos, é necessário compreender como os alunos aprendem, sendo que um dos maiores desafios da educação é manter os alunos motivados e interessados durante o tempo que for necessário para aprender, o que motiva o professor a buscar meios que possibilitem maior envolvimento dos alunos. Para tanto, é sugerido o uso das Tecnologias de Informação e Comunicação (TIC) como apoio à educação devido às inúmeras soluções que estas podem oferecer no âmbito acadêmico. Outra solução apresentada é a aprendizagem híbrida, também citada em (Ortiz-Ortiz et al., 2018). De acordo com Eltegani & Butgereit (2015), a aprendizagem híbrida é conceituada como "um formato de sala de aula que consiste em uma mistura de atividades equilibradas entre interação online e face a face, bem como sequenciamento cuidadoso de atividades acadêmicas" (p. 103, tradução nossa) e os autores apresentam dois fatores que tornam essa estratégia de ensino necessária: "necessidade de atividades que economizam tempo devido à agenda sobrecarregada dos alunos. E a disponibilidade/acessibilidade de todos os tipos de tecnologias de ponta que podem ser utilizadas na educação" (p. 103, tradução nossa). Diante da pesquisa realizada, os autores elencam os principais resultados obtidos. Primeiramente, apresentam os fatores individuais que contribuem para o envolvimento dos alunos no processo de aprendizagem de programação, tais como conquista, motivação e objetivos a serem alcançados; depois apresentam as práticas pedagógicas que podem ser adotadas para contribuir com o envolvimento dos alunos, como estratégias de aprendizagem ativa, bem como aprendizagem por colaboração, aumentando a interação entre os alunos, assim como apresentado por Shadiev et al. (2013). Além disso, são apresentados atributos necessários para auxiliar o processo de aprendizagem de programação, tais como estudo intensivo, suporte e feedback contínuo por parte do professor, prática intensiva, disponibilidade de recursos tecnológicos para o apoio ao docente e aluno, e a promoção de autoconfiança no aluno, para que este sinta-se motivado e capaz de aprender a programar.

Embora os autores tenham apresentado atributos, estratégias e ferramentas, não foram realizados experimentos ou estudos para validar, em um ambiente real de aprendizagem, a eficácia destes para verificar o contributo em relação ao envolvimento dos alunos na aprendizagem de programação.

A estratégia proposta por Nawahdah et al. (2015) consiste na aprendizagem por programação em pares (*pair programming*). De acordo com os autores, em 1999 surgiu o conceito da eXtreme Programming (XP) e uma das práticas mais polêmicas introduzidas pela XP foi a programação em pares. Segundo Nawahdah et al. (2015), na programação em pares dois programadores compartilham o mesmo computador, sentados lado a lado, olhando para a mesma tela, usando recursos de

hardware como teclado e mouse para manipular o computador e desenvolver o código. Os programadores, na programação em pares, trabalham juntos, revezando-se para digitar, discutindo o código para criá-lo, revisá-lo e depurá-lo. A técnica de programação em pares pode ser aplicada na indústria de software e em laboratórios de pesquisa, sendo amplamente aplicada como técnica de ensino de programação (Nawahdah et al., 2015), contudo, o sucesso da programação em pares depende do envolvimento de ambos os membros que formam o par, de forma que trabalhem igualmente para se obter os melhores resultados. A principal questão de pesquisa desenvolvida era verificar se a programação em pares possui o potencial de melhorar o desempenho, em termos de notas e êxito dos alunos, e permitir que eles produzissem código de software de qualidade, dentro das restrições locais. Para avaliar a técnica, os autores desenvolveram um experimento em turmas de programação avançada, em dois semestres, realizando o experimento em duas seções para cada semestre: uma delas baseada em programação em pares e a outra era uma seção de controle, onde o curso era ministrado seguindo o percurso tradicional.

Durante o experimento, os alunos eram convidados a trabalhar em pares apenas nas aulas práticas de laboratório, sob a supervisão de um professor e de um assistente, que os instruíam a mudar de função entre seus pares e os incentivavam a discutir o problema antes de começar a resolvê-lo, evitando pedir apoio ao professor e ao assistente. Já nas seções de controle, os alunos não recebiam instruções específicas, mas eram incentivados a buscar soluções de forma autônoma. Para a formação dos pares, "a literatura mostrou que dois métodos principais de formação de pares se destacaram por terem mais méritos que outros; seleção aleatória e os alunos selecionando seus próprios parceiros" (Nawahdah et al., 2015, p. 18, tradução nossa). Para o experimento no primeiro semestre, os alunos escolheram seu próprio par. Já no segundo semestre, os pares foram escolhidos pelo professor e pelo assistente. Para a coleta de dados, os autores desenvolveram questionários, avaliaram os trabalhos desenvolvidos pelos alunos e observaram as aulas em laboratórios. Verificou-se que nas seções de programação em pares, os alunos se interessavam mais a ir às aulas práticas, confiavam mais no código produzido, participavam das aulas mais preparados, e se interessavam mais em ter ideias diferentes e para desenvolver os programas. Como resultado, o estudo revelou que a programação em pares melhorou a taxa de conclusão do curso, o aproveitamento do curso pelos alunos, e a qualidade do código que eles produziram até o final do semestre.

Outra estratégia para auxiliar no processo de aprendizagem de algoritmos e programação é o uso de linguagem de programação visual. Em seu trabalho, Rezende & Bispo (2018) avaliam a contribuição da linguagem de programação visual Scratch em comparação à aprendizagem de

algoritmos com pseudocódigo, tendo como público-alvo alunos de um curso de Licenciatura em Física. De acordo com os autores, com a linguagem de programação visual Scratch, "o usuário organiza os blocos de acordo com a sequência lógica que ele acredita ser o ideal para escrever a sua estória [...] sem se preocupar com detalhes da sintaxe de uma linguagem de programação" (p. 492). Ainda, de acordo com os autores,

> Adotar essa linguagem para auxiliar no processo de ensino-aprendizagem da lógica de programação, e analisar o resultado proveniente dessa utilização, pode trazer uma avaliação mais ampla sobre as abordagens tradicionais para o ensino de programação quando comparadas com o uso de tecnologias didáticas diferenciadas, nesse caso, a linguagem de programação visual. (p. 492)

Para o desenvolvimento da pesquisa e obtenção dos resultados, os autores realizaram um procedimento técnico de pesquisa-ação, com objetivo experimental, por meio de abordagens qualitativas e quantitativas para o levantamento dos dados, usando o questionário como instrumento de recolha. Dessa forma, Rezende & Bispo (2018) atribuíram conceitos e tarefas a serem desenvolvidos em oito diferentes etapas (ideia inicial, reconhecimento, planejamento das atividades, implementação, monitoramento, evolução do efeito da ação, alteração das atividades e saída de execução do plano), com a finalidade de obter os resultados da pesquisa. Como resultado, os autores verificaram que a linguagem de programação visual Scratch contribui positivamente no processo de ensino e aprendizagem de programação, especialmente no tocante a conteúdos menos complexos, e complementam dizendo que "a adequação dos recursos às etapas de ensino, de forma conveniente para o processo de ensino e aprendizagem, propicia um melhor aproveitamento dos mesmos" (Rezende & Bispo, 2018, p. 497).

A pesquisa desenvolvida por Piteira et al. (2017) propõe um *framework* conceitual gamificado para cursos de aprendizagem de programação on-line. Para a realização da pesquisa, os autores identificaram as características e elementos de gamificação utilizados no contexto educacional. Segundo os autores, a aplicação da gamificação no contexto educacional pode contribuir para aumentar a motivação dos alunos na aprendizagem, de forma que a gamificação oferece ciclos de retorno curtos e permite que os alunos avaliem suas habilidades e criem um ambiente em que o esforço de aprendizado seja recompensado. Além disso, "esta abordagem está a ganhar notoriedade pela sociedade e, é por muitos vista como um alto impulsionador da motivação extrínseca dos estudantes reduzindo os efeitos negativos das suas faltas de motivação intrínseca" (Piteira et al., 2017, p. 1, tradução nossa). Como alternativa, os alunos podem ver o fracasso como uma oportunidade de

aprendizado e são particularmente úteis para estudantes que tendem a desistir facilmente quando os resultados não forem os esperados. No *framework* proposto, os autores apresentam as principais dimensões adaptadas à inserção da gamificação em cursos de programação a distância e on-line, sendo elas: público-alvo, objetivos gerais, objetivos e tópicos específicos, conteúdo, princípios do desenho educacional, mecânica de jogo, absorção cognitiva, fluxo e personalidade.

Piteira et al. (2017) afirmam que "a aplicação da gamificação no contexto educacional pode contribuir para aumentar o envolvimento do aluno na aprendizagem" (p. 2, tradução nossa). O uso de elementos de jogos na educação tem recebido maior atenção nos últimos anos, tanto por pesquisadores quanto por educadores (Piteira et al., 2017). Embora o *framework* proposto tenha como alvo cursos de programação, ele possui uma estrutura teórica que pode ser adaptada para diferentes cursos on-line suportados por uma plataforma de aprendizagem. O *framework* proposto por Piteira et al. (2017) foi integrado ao sistema de gerenciamento de aprendizagem (*Learning Management System* - LMS) Moodle, sendo validado em um experimento com 108 estudantes no Ensino Superior. Os resultados apresentaram a viabilidade em se ter um sistema gamificado para a aprendizagem de programação.

A investigação desenvolvida por Pereira et al. (2017) recorre a vários autores para apresentar como a gamificação pode melhorar a aprendizagem de programação. Caponetto, Earp e Ott (2014), (como citado em Pereira et al., 2017) afirmam em sua pesquisa que a maioria dos estudos sobre gamificação "apoiam-se em obras anteriormente criadas por outros investigadores, com o objetivo de confirmar a capacidade que a gamificação tem em aumentar a motivação e o empenho dos estudantes" (p. 4). Além disso, Caponetto, Earp e Ott (2014)), (como citado em Pereira et al., 2017) complementam dizendo que a gamificação ajuda a completar objetivos transversais, tais como fomentar a participação e colaboração entre estudantes, aprendizagem autodidata, realização dos trabalhos extrassala de aula, desenvolver a criatividade dos alunos, sendo estas algumas das habilidades citadas por autores da área de aprendizagem de programação.

Ho, Chean, Chai & Tan (2019) apresentam um modelo de experiência no aprendizado de programação. Diante do crescimento do número de cursos on-line abertos e massivos (*Massive Online Open Courses* - MOOC) para o ensino de programação, em que diversos alunos de todo o mundo podem participar de um mesmo curso e necessitam de um apoio ao processo de aprendizagem de programação, os autores preocuparam-se em investigar questões tais como: como ajudar os alunos a compreenderem os erros de programação e se o modelo proposto ajuda os alunos no processo de aprendizagem de programação. A preocupação central era o provimento de avaliação automática, por

meio de feedback, em que o aluno pudesse acompanhar o progresso de sua aprendizagem, ao passo que percebia os defeitos na sua programação. De acordo com Ho, Chean & Tan (2019), "avaliação automatizada pode reduzir o esforço dos educadores, além de beneficiar os alunos com feedback imediato" (p. 926, tradução nossa). A proposta elaborada pelos autores está relacionada principalmente à análise estática do programa, isto é, ao seu código fonte, de forma que, para o desenvolvimento do modelo de experiência de aprendizagem de programação (*Programming Learning Experience Model* – PLEM), os autores se basearam em quatro tipos de perguntas de avaliação de aprendizagem de programação (*Programming Learning Assesment* – PLA), as quais favorecem a possibilidade de feedback automático: verdadeiro ou falso, múltipla escolha, preenchimento de espaços em branco e tempo de conclusão da resposta. Para avaliar como melhorar a avaliação automática para melhorar a aprendizagem de programação, foram desenvolvidos quatro itens de medição: escore de medição, velocidade da avaliação, eficiência e utilidade do feedback. Embora os autores já tenham disponibilizado o modelo em um curso introdutório de programação, o estudo não apresenta a validação do mesmo em um ambiente real de aprendizagem.

O trabalho apresentado por Castro et al. (2011) aborda a análise e representação de padrões na aprendizagem de programação em grupo. De acordo com os autores, o uso de recursos de *grupoware* baseados na internet possibilitam a inserção de boas práticas no processo de aprendizagem de programação, isso por que esse recurso permite a gravação das interações e ações de um aluno ou grupo de alunos durante a resolução de um exercício, ou do desenvolvimento de um código, além disso "o registro da atividade é um requisito essencial para manter o controle de sua progressão, tornando possível para o professor ou qualquer outro colega dar feedback apropriado durante os exercícios" (p. 2, tradução nossa). Esses recursos permitem que sejam extraídas informações que auxiliem os investigadores a analisarem os padrões do processo de aprendizagem de programação em grupo. Para o desenvolvimento da pesquisa, os autores realizaram um estudo de caso com 110 alunos (entre alunos de Ciência da Computação e de Engenharia da Computação), durante um semestre, dividido em sete fases. Com o estudo, foi possível identificar padrões de interações entre os alunos e, posteriormente, identificar como e quando deve haver intervenção do professor para melhorar o processo de aprendizagem dos alunos.

A pesquisa desenvolvida por Mendes et al. (2012) propõe a reformulação pedagógica do ensino de programação introdutória. A ideia central consiste em aumentar o nível do comprometimento do aluno com o curso, tendo em vista as altas taxas de desistência e reprovação nas disciplinas de programação. O artigo apresenta uma visão detalhada das mudanças realizadas, a avaliação da

proposta e os resultados obtidos. O estudo foi realizado tanto com alunos novatos, quanto com alunos repetentes (ou seja, que já conheciam a metodologia anterior, e experimentaram a metodologia proposta). A principal mudança se deu em relação ao melhor aproveitamento do tempo da aula explanativa, aumento das aulas práticas e diminuição do número de alunos nas aulas práticas. Com a aplicação e validação do método, os autores verificaram que houve um aumento da motivação dos alunos, bem como de suas notas na disciplina, da média geral da turma (em comparação a anos anteriores) e aumento do número de aprovação no curso, diminuindo, portanto, o índice de reprovação e abandono.

Santos et al. (2010) apresentam em seu trabalho ferramentas autorais de apoio ao ensino e aprendizagem de programação, focando nos benefícios que cada aplicação traz para tanto. Dentre as ferramentas desenvolvidas estão a VIP, desenvolvida em 1980, que permite o desenvolvimento de pseudocódigo e a simulação do mesmo; SICAS, desenvolvida em 1990, baseada na teoria construtivista, e que apresenta um ambiente que permite aos alunos desenvolver suas capacidades na base da prática, permitindo que eles projetem, observem, analisem e simulem visualmente algoritmos; OOP-Anim, desenvolvida em 2004, que ajuda os alunos a compreender os conceitos da abordagem orientada a objetos por meio da criação e visualização dos programas; SICAS-COL, desenvolvida em 2005, que dá suporte a atividades de aprendizagem colaborativa; ProGuide, desenvolvida em 2005, baseada em um modelo de sistema de tutoria, auxilia alunos a diminuir as dificuldades encontradas nos estágios iniciais de programação; H-SICAS, criada em 2008, é uma evolução da ferramenta SICAS adaptada para dispositivos móveis; SICAS NG, que é atrelada ao LMS Moodle, tendo também como objetivo o desenvolvimento de pseudocódigo e sua simulação por meio de fluxogramas, além disso, por estar disponível em um ambiente on-line, permite a colaboração síncrona do professor no processo de aprendizagem de seus alunos.

O trabalho apresentado por Chuchulashvili et al. (2016) propõe a realização de micro atividades baseadas em *WebQuests* como contributo para a aprendizagem de programação. De acordo com os autores, "os *WebQuests* surgiram como um método de integrar o uso da internet de uma forma positiva e controlada no processo de aprendizagem" (p. 1505) e essa estratégia pode aumentar a motivação e a autonomia dos alunos na resolução de exercícios de programação que possam incrementar o envolvimento do aluno na aprendizagem e, consequentemente, aumentar o tempo e esforço que os alunos despendem na busca pela solução dos problemas. A proposta baseia-se numa plataforma *web* que apresenta aos alunos problemas de forma contextualizada e interessante, sugerindo a divisão por tarefas curtas, mensuráveis e de resolução com um grau crescente de

dificuldade, dando feedback imediato ao aluno, possibilidade de mais do que uma tentativa de resolução, sugestões de melhoria, explicações acrescidas sobre outras soluções, classificação obtida e ligação aos passos seguintes de aprendizagem (Chuchulashvili et al., 2016). Para validar e avaliar a proposta, os autores desenvolveram um experimento com 40 alunos divididos em dois grupos, um de controle (que participou da aula tradicionalmente) e um de experimento (que utilizou o *WebQuest*). Como resultado, observou-se que não houve diferença no desempenho quando comparados os dois grupos, em relação à aprendizagem de programação. No entanto, os alunos que experimentaram o uso do *WebQuest* analisaram a proposta como interessante, intuitiva e viável para a aprendizagem de programação. Além disso, os autores verificaram um aumento na autonomia e capacidade de raciocínio dos alunos que utilizaram o *WebQuest* durante a pesquisa, propondo, portanto, como uma ferramenta que pode ser utilizada como apoio à aprendizagem de programação.

Costa (2019) apresenta a utilização de *microworlds* como solução para superar as dificuldades dos alunos diante da aprendizagem de algoritmos e programação. Para tanto, utiliza o software Alice1, que é considerado um *microworld* para o ensino de programação, tendo por objetivo compreender os efeitos desse software na aprendizagem de programação, quando associado a alguns métodos pedagógicos, especificamente no conhecimento de programação e raciocínio lógico. Para validar a proposta, a autora desenvolveu um experimento com dois estudos: um grupo experimental, que aprendeu os conceitos de programação com o software Alice e pseudocódigo; e um grupo de controle, que aprendeu os mesmos conceitos sem o uso da ferramenta, de forma que, em ambos os casos, a metodologia de ensino utilizada pelo professor foi a expositiva. Os resultados obtidos no experimento sugerem que não se obteve diferenças em relação à aprendizagem de ambos os grupos.

Al-Imamy et al. (2006) propuseram uma ferramenta baseada na *web* para o ensino de programação. A motivação central para o desenvolvimento da ferramenta é a quantidade excessiva de tempo de aula gasto no ensino da sintaxe de linguagem de programação, "que limita, e muitas vezes impede, a capacidade de atingir os outros dois objetivos fundamentais de programação de desenvolver habilidades de design de programa e criatividade na solução do problema" (p. 280, tradução nossa). Para validar a proposta, os autores experimentaram a ferramenta com um grupo de 20 estudantes e, como resultado, verificou-se que a ferramenta foi eficaz para acelerar o ensino e a aprendizagem da sintaxe da linguagem. Além disso, os autores puderam observar que a ferramenta pode ser usada na criação de modelos que reforçam os conceitos e construções importantes; a ferramenta é flexível e permite alternar de um idioma para outro; ajuda a preencher a lacuna entre estudantes de várias

1 Alice Project. http://www.alice.org

origens e trazê-los à tona a um nível igual em um estágio inicial do curso; ajuda a aumentar o entusiasmo e a confiança do aluno em escrever programas corretos; e pode ser usada como uma ferramenta de autoaprendizado baseada na *web* que reduzirá a necessidade de ensino e assistência de aprendizagem.

Souleiman (2017) desenvolveu sua investigação com o objetivo de verificar como orquestrar atividades de aprendizagem para promover, incentivar e facilitar a aquisição de habilidades que faltam aos estudantes para a aprendizagem de programação. Para tanto, observou os seguintes cenários: planejamento, regulação, adaptação e avaliação, por meio de uma revisão de literatura. Como contributo, é proposto um modelo de avaliação da solução do aluno descrita no pseudocódigo usando um método pedagógico baseado em análise estática, a fim de verificar se a solução proposta pelo aluno implementa todas as ações previstas e detectar, a partir daí, o entendimento e suas dificuldades para resolver o problema. O objetivo desta avaliação é acompanhar o aluno na aquisição e melhoria das suas habilidades, implementando um assistente baseado em seleção de feedback ou tarefa (atividade), dependendo das dificuldades detectadas. Consequentemente, as atividades propostas são modeladas de maneira que uma adaptação possa ser proposta de acordo com esta observação.

Skalka & Drlik (2018) elencaram em seu trabalho alguns tipos de domínios de conhecimento e habilidade necessárias para a aprendizagem de algoritmos e programação, os quais podem ser categorizados desde o nível de iniciante até especialista. De acordo com os autores, esses níveis são: domínio do problema, domínio da linguagem de programação, domínio de transformação do problema para a linguagem de programação, domínio para entender o código fonte, domínio de entendimento profundo, e domínio de desenvolvimento de software. Tendo em vista a complexidade de se aprender a programar e suas consequências, tais como desistência e retenção, os autores propuseram o sistema PRISCILLA (*PRogressIve System for interaCtive programmIng Learning and Learning Analytics*), que busca suprir os níveis de domínio acima apresentados, logo, por trás do sistema, estão os conceitos de micro aprendizagem, de avaliação automatizada, gamificação, aprendizado em grupo, personalização, *learning analytics* e solução de problemas reais. Essas características estão divididas em diferentes módulos do sistema. Ao longo do artigo, os autores apresentam a arquitetura do sistema e como cada característica é implementada, no entanto não realizam a validação do mesmo em um ambiente de aprendizagem.

O trabalho realizado por Gomes & Mendes (2010) apresenta quatro estudos a partir dos quais desenvolveu-se uma proposta pedagógica para superar algumas dificuldades existentes no processo de ensino e aprendizagem de programação. Para tanto, considerou-se um conjunto de atividades que

devem ser realizadas, sugeriu-se a eliminação da tradicional divisão das aulas entre aulas expositivas e práticas e a diminuição do número de alunos por turmas, para cerca de, no máximo, 20 alunos, para garantir que as aulas tenham uma combinação de teoria e prática e um bom nível de interatividade. Para Gomes & Mendes (2010), "é importante ter o aluno como principal ator do processo de aprendizagem e não como um mero agente passivo. Para conseguir isso, é importante uma boa motivação do aluno e uma atitude adequada do professor" (p. 5, tradução nossa). Além disso, os autores propõem que os professores utilizem estratégias que possam motivar o processo de aprendizagem dos alunos, tais como: propor exercícios relevantes usando exemplos do dia a dia; permitir que os alunos escolham o tipo de tarefa a ser realizada; motivar os alunos a se envolverem nas tarefas de programação, mesmo aquelas mais simples; e propor que os alunos colaborem entre si. Assim, ao proporem turmas menores, os autores acreditam que o professor terá mais facilidade em perceber os problemas de aprendizagem dos alunos, fazendo intervenções diretas, e diminuindo, portanto, taxas de abandono, podendo auxiliar o aluno a obter êxito.

Embora sejam diferentes as metodologias descritas na literatura, não existe um consenso a respeito de qual é a melhor ou mais eficaz. Entretanto, compreendemos que o a metodologia deve ser escolhida de acordo com o contexto, a estrutura do curso, o perfil da turma, e o conteúdo que está sendo ensinado. Além disso, percebemos que é importante, para além da metodologia, existir ferramentas que deem suporte à sua condução. Dessa forma, a seção a seguir aborda algumas ferramentas utilizadas no processo de ensino e aprendizagem de programação que encontramos na nossa revisão de literatura. O objetivo do estudo é desenvolver um inventário das ferramentas, para que professores e alunos possam ter acesso e utilizá-las conforme a necessidade.

2.2.3.2. Ferramentas Utilizadas no Processo de Ensino e Aprendizagem de Programação. Nesta seção, apresentamos as ferramentas encontradas ao longo da revisão de literatura desenvolvida para este capítulo. As ferramentas estão classificadas, de acordo com suas características, nas seguintes categorias:

a) Ambiente com Gamificação, de desenvolvimento de jogos ou ambiente de jogo: possui elementos de jogos, permite que o utilizador desenvolva jogos ou quando trata-se de um jogo, respectivamente.
b) Ambiente de Programação com Fluxogramas: utiliza fluxogramas para o ensino e aprendizagem de programação.
c) Ambiente ou linguagem de Programação Visual: fornece elementos gráficos ou icônicos que podem ser manipulados pelos usuários de forma interativa.

d) Ambiente de Recomendação de Atividades: fornece a recomendação de atividades a serem desenvolvidas, de acordo com o perfil do utilizador.

e) Ambiente de Resolução de Problemas: oferece aos utilizadores diferentes problemas a serem solucionados.

f) Ferramenta de Correção Automática ou *feedback*: fornece a correção automática ou *feedback* automático ou personalizado aos utilizadores.

g) Ferramenta para Programação: ferramentas para o desenvolvimento de programas em uma linguagem específica.

h) Juízes On-line: repositório de problemas com correção e *feedback* automático.

i) Micromundos: permite a exploração e simulação de acontecimentos da vida real.

j) Programação em Blocos: fornece a possibilidade de programar em 'blocos de encaixe', em uma alternativa às linhas de código.

k) Robótica: permite o ensino e aprendizado com a programação de robô.

l) Sistema Tutor Inteligente: fornece suporte automático ao processo de aprendizagem.

Cada ferramenta pode estar classificada em uma ou mais categorias, e cada categoria pode abranger uma ou mais ferramentas. As categorias foram criadas baseadas no trabalho realizado por Medeiros (2019). Na Tabela 5, apresentamos o inventário destas ferramentas, com o nome, a categorização e a referência bibliográfica em que é apresentada.

Tabela 5 - *Inventário de ferramentas para o ensino e aprendizagem de programação*

Categoria	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**Autores FT***	**Autores RSL***
Ferramentas														
Agent Cubs	X		X						X	X			(*AgentCubes*, [s.d.])	
LabPy					X	X	X						(Sirotheau et al., 2018)	
PRISCILLA	X				X	X	X							(Skalka & Drlik, 2018)
Program Your Robot	X		X							X	X		(Kazimoglu et al., 2012)	
URI Online Judge	X				X			X		X			(Bez et al., 2014)	
Hour of Code			X		X					X			(Wilson, 2014)	
Huxley					X	X							(Paes et al., 2013)	
Kodu Game Lab	X		X										(*Kodu Game Lab*, [s.d.])	

MOJO				X	X		X					(Chaves et al., 2013b)	
ProAlg				X	X	X						(Franzen et al., 2018)	
Process Legend	X		X						X			(Nishida et al., 2017)	
Robocode	X									X		(Hartness, 2004)	
Squeak Etoys	X		X					X				(Ingalls et al., 1997)	
TSTView				X	X	X						(Gaudencio et al., 2013)	
VisualLogic		X	X		X							(*Visual Logic*, [s.d.])	
Alice			X					X				(Cooper et al., 2000)	
Ask-Elle					X						X	(Gerdes et al., 2017)	
Code Chef	X			X								(*CodeChef*, [s.d.])	
CodeBlocks						X			X			(*CodeBlocks*, [s.d.])	
Construct 2	X		X									(*Construct 3,* [s.d.])	
Flowgorithm		X	X									(Cook, 2015)	
FurBot-Web				X					X			(Lopes & Kopsch, 2018)	
Game Maker	X		X									(*GameMake*, [s.d.])	
GameProgJF	X		X									(Stephan et al., 2020)	
iVProg			X						X			(Kamiya & Brandão, 2009)	
JavaTool			X		X							(Sirotheau et al., 2012)	
LightBot				X								(*LightBot*, [s.d.])	
Mblock			X							X		(*mBlock*, [s.d.])	
MIT APP Inventor			X	X								(Magnuson, 2010)	
Monster Coding									X			(*Monster Coding*, [s.d.])	
ProGuide						X					X		(A. Santos et al., 2010)
ProgTest				X	X							(de Souza et al., 2011)	
PyGame	X		X									(*PyGame*, [s.d.])	

RoboMind					X					X		(*RoboMind*, [s.d.])	
run.Codes						X						(*run.codes*, [s.d.])	
Scratch			X						X			(Resnick et al., 2009)	(Rezende & Bispo, 2018)
SICAS-NG		X					X						(A. Santos et al., 2010)
SOAP				X	X							(Oliveira & Oliveira, 2014)	
Takkou			X									(Barbosa et al., 2011)	
AAPW							X					(M. Gomes et al., 2014)	
Agile	X											(Vasconcellos et al., 2019)	
Analogus				X								(Santos Júnior et al., 2009)	
BASICS							X						(Ortiz-Ortiz et al., 2018)
Blockly									X			(Fraser, 2015)	
BlueJ							X					(Kölling et al., 2003)	
Bridge											X		(Chang et al., 2000)
CAPRA											X		(Chang et al., 2000)
Code Combat	X											(Wood et al., 2015)	
Code Monkey	X											(Khouri et al., 2020)	
Code.org	X											(Kalelioğlu, 2015)	
CodeMage							X					(Whittall et al., 2017)	
Dekstra					X							(Nicácio & Costa, 2018)	
EduJudge								X				(Verdú et al., 2012)	
Escracho			X									(Jesus, 2010)	
Etoys			X									(Kay et al., 1997)	
Feeper						X						(F. P. Alves & Jaques, 2014)	
H-Sicas							X						(A. Santos et al., 2010)
Jcolibri							X					(Díaz-Agudo et al.,	

						2007)	
Lego Mindstorms NXT					X	(Anderson & Gavan, 2012)	
Lisp TUTOR					X		(Chang et al., 2000)
Logirunner	X					(Casarotto et al., 2018)	
OOP-Anim			X				(A. Santos et al., 2010)
Pict		X				(Glinert & Tanimoto, 1984)	
Portugol Studio			X			(Noschang et al., 2014)	
ProGame	X					(Sales & Dantas, 2010)	
Proust					X		(Chang et al., 2000)
Raptor		X				(Carlisle et al., 2005)	
SICAS			X				(A. Santos et al., 2010)
SICAS-Col			X				(A. Santos et al., 2010)
Snap				X		(Harvey et al., 2013)	
StarLogo TNG		X		X		(Begel & Klopfer, 2007)	
Super Mario Logic	X					(Panegalli et al., 2019)	
Tri-Logic	X					(Natal et al., 2018)	
TutorICC			X			(Píccolo et al., 2010)	
VIP			X				(A. Santos et al., 2010)
ViP		X				(Barbosa et al., 2019)	
VisualG			X			(de Souza, 2009)	
VPL				X		(Chan et al., 2003)	
VPEN			X				(Shadiev et al., 2013)
WH-IDE			X			(Frantz & Pontes, 2014)	

WPASC	X	(Hwang et al., 2012)

Nota: *Fundamentação teórica. **Revisão Sistemática de Literatura. Autoria própria.

Com o estudo realizado, encontramos 81 diferentes ferramentas de apoio ao ensino e aprendizagem de programação. As ferramentas apresentadas na Tabela 5 estão em ordem decrescente de acordo com número de categorias em que cada uma está atribuída. A atribuição das categorias foi realizada de acordo com as descrições das ferramentas apresentadas em cada estudo. Percebemos que a maior ocorrência é na categoria 'ambiente ou linguagem de Programação Visual' (c), que trata de ferramentas como o Scratch e CodeBlocks, por exemplo. O 'ambiente de recomendação de atividades' (d), por sua vez, tem apenas duas ferramentas com suas características. As ferramentas aqui apresentadas estão disponibilizadas em um inventário[2] de nossa autoria que pode ser expandido de forma colaborativa por professores, alunos e entusiastas da área, visando trazer novas possibilidades para o apoio ao processo de ensino e aprendizagem de programação.

2.3. Síntese Conclusiva do Capítulo

Neste capítulo apresentamos conceitos importantes apresentados em duas diferentes seções que nos dão melhor entendimento e suporte para o desenvolvimento das etapas seguintes para a condução desta tese. Primeiramente, na seção 'definição e contextualização' vimos que o processo de aprendizagem de programação é tarefa considerada complexa e que coloca desafios importantes aos docentes de Informática, que são diferentes as percepções dos pesquisadores sobre o que é programar, mas que convergem para o processo de interpretação e compreensão do problema – especificação dos requisitos para resolvê-lo – implementação de um plano de resolução – implementação do algoritmo em uma linguagem de programação – execução do programa e testagem, passos esse que são cíclicos e iterativos, até que se atinja o resultado esperado.

Também é complexo por estar em diferentes áreas do conhecimento, por ter diferentes designações e níveis de formação, embora seja predominante em cursos de Informática e áreas afins.

Na seção 'caracterização' percebemos com mais profundidade as dificuldades enfrentadas pelos discentes para o processo de aprendizagem de programação, que podem ser devido ao currículo do estudante (base matemática, dificuldade em interpretação de textos, dificuldades em lógica), a questões socioeconômicas (falta de recursos para estudar, ter que conciliar estudo com trabalho), ou até mesmo à estrutura do curso superior (turmas muito grandes, formação docente inadequada, turmas heterogêneas). Notamos que as dificuldades podem gerar consequências tais como retenção,

[2] O inventário está disponível em: https://padlet.com/ferramentasparaprogramar/inventario_de_ferramentas

desistência, frustração e desmotivação dos alunos, que podem afetar o seu processo de aprendizagem e seu percurso acadêmico.

Ainda na caracterização, pôde-se conhecer ainda as condições que podem auxiliar no processo de aprendizagem de programação, e que vêm desde a forma como ocorre o primeiro contato do aluno com a programação, ou seja, a abordagem pedagógica, a linguagem de programação adotada pelo professor e o apoio dado pelo docente, até a forma como o aluno organiza-se para estudar e se este está motivado para aprender. Por outro lado, um aluno que estuda de maneira superficial ou resolve poucos exercícios certamente terá sua aprendizagem afetada. Além destes fatores, destacam-se como habilidades importantes para aprender a programar a autoconfiança, autoestima, proatividade e planejamento; como conhecimento destaca-se a abstração, algoritmia, conhecimento em matemática e conhecer linguagens de programação (embora não seja necessário esse conhecimento em um primeiro momento; e como competência é interessante que o aluno seja criativo, cooperativo, saiba trabalhar em equipes e consiga ligar com regras.

Com o objetivo de dirimir as dificuldades e auxiliar na melhoria ou aquisição das condições para a aprendizagem, apresentamos diferentes metodologias e ferramentas que vêm sendo desenvolvida ao longo dos últimos anos. Vários são os esforços de pesquisadores na busca de compreender o processo de aprendizagem, melhorando assim formas de se ensinar e aprender a programar. Pode-se destacar que independente da metodologia ou ferramenta adotada, a aprendizagem de programação destaca-se nestas três estratégias: prática – uso de bom material de apoio – recorrer a aulas de programação que sejam confiáveis. Aliada a estas estratégias, encontramos na literatura a programação por pares, a sala de aula invertida, o uso de jogos e linguagem de programação visual. Entretanto, não existem evidências da melhor metodologia ou ferramenta a ser adotada, uma vez que isto dependerá do contexto em que está se ensinando a programação.

Dessa forma, este capítulo pautou-se na busca para responder os objetivos desta tese, apresentados no capítulo introdutório, e ainda para nos dar respaldo e orientação no desenvolvimento das demais atividades desenvolvidas no âmbito do doutoramento, a saber: os procedimentos metodológicos, os objetos de recolha e a forma de conduzir as análises e interpretações dos dados obtidos durante o estudo de caso desenvolvido, que estão detalhados no capítulo 3 desta tese.

3. Enquadramento Metodológico

Neste capítulo apresentaremos o enquadramento metodológico definido a partir da questão de partida e dos objetivos centrais desta tese. Referimos que a fundamentação teórica desenvolvida, bem como todo o percurso acadêmico traçado ao longo do doutoramento, foram fundamentais para o desenvolvimento de uma reflexão crítica da temática, estruturação e redação deste texto.

O capítulo está estruturado em duas seções: abordagem metodológica e procedimento metodológico. Essas seções são fundamentais para a contextualização da presente pesquisa, tornando-a clara e objetiva, ancorada numa abordagem mista e imersa num estudo de caso inerente ao processo de ensino e aprendizagem de programação no Ensino Superior, tendo como caso um Curso de Computação [3]em uma Universidade integrada no contexto brasileiro, contexto este para o qual este trabalho foi desenvolvido.

Para a concretização do estudo de caso elaboramos um projeto que de seguida foi apreciado em comitê de ética em pesquisa para então ser iniciado. Além disso, definimos os participantes do estudo de caso, elaboramos os instrumentos de recolha de dados, definimos a forma de registro e tratamento de dados, bem como a forma da análise e interpretação dos desses.

3.1. Abordagem Metodológica

A modalidade de uma pesquisa científica deve estar alinhada com a questão de partida e os objetivos, para garantir um melhor êxito do trabalho e a possibilidade de sua aplicação em outros contextos. A abordagem mista que enquadra a nossa pesquisa possibilitou um melhor direcionamento para as atividades realizadas ao longo do desenvolvimento desta tese, além de impor credibilidade e coerência a recolha e análise dos dados (Richardson, 2017).

Entendemos que a pesquisa realizada sobre o processo de ensino e aprendizagem de programação tendo como caso um curso de Computação proporciona uma compreensão deste processo que é complexo e que abrange professores e alunos dos mais diversos cursos de Ensino Superior da área de Informática.

O estudo de caso foi considerado uma estratégia apropriada para nossa pesquisa e, para tanto, nos pautamos em Yin (2015), que define estudo de caso como uma investigação empírica que explora um fenômeno contemporâneo dentro do seu contexto da vida real, principalmente quando os limites existentes entre o fenômeno e o contexto não estão bem definidos. Conforme André (2013), estudo de caso é, no âmbito da educação, uma estratégia metodológica que busca analisar determinada

[3] Por questões éticas, para garantir o sigilo dos dados, omitimos o nome do curso e a Universidade ao qual ele pertence. Sempre que nos referirmos ao caso, usaremos o termo Curso, grafado em maiúsculo.

realidade educativa de maneira mais profunda e particular, possibilitando uma compreensão holística do objeto estudado.

De acordo com Yin (2015), existem três condições para escolher o método de pesquisa apropriado para uma investigação, os quais são: a) o tipo da questão de pesquisa proposto; b) a extensão do controle que um pesquisador tem sobre os eventos comportamentais reais; c) o grau de enfoque sobre os eventos contemporâneos em oposição aos eventos totalmente históricos.

Respondendo a estas três condições, observamos respectivamente que: nossa pesquisa tem como tipo de questão 'como', ao perguntarmos 'como ocorre o processo de ensino e aprendizagem de programação no Ensino Superior?'; não exige controle dos eventos comportamentais, ou seja, não interferimos nos fenômenos nem nos comportamentos para obter os dados a serem analisados; e enfoca eventos contemporâneos e não históricos. Assim, de acordo com Yin (2015), ela enquadra-se nas condições de um estudo de caso.

Para a nossa pesquisa, desenvolvemos um estudo de caso único, tendo o propósito de investigar um caso de ensino e aprendizagem de programação no Ensino Superior, de forma a conhecer metodologias adotadas e ferramentas utilizadas, conhecimentos necessários, fatores e dificuldades enfrentadas pelos alunos durante a aprendizagem de programação.

O estudo de caso único auxilia na investigação em circunstâncias distintas, dentre elas quando a circunstância é representativa, isto é, trata-se de um projeto típico entre muitos outros projetos e o que é aprendido desse caso fornece informações sobre experiências para outros (Yin, 2015). Dessa forma, estudar o caso de processo de aprendizagem de programação no Curso escolhido pode oferecer novos conhecimentos e contributos para outros estudos nessa área.

Percebemos nosso estudo de caso único com a variante holística, uma vez que analisaremos um único caso, representado por seus alunos, professores, sua estrutura e Unidades Curriculares. Para obtermos dados que contemplem o processo de aprendizagem ao longo das UC de programação e não apenas de parte delas, fazem parte da pesquisa, além dos alunos, professores das diferentes UC, os documentos que se relacionam diretamente ao Curso, como por exemplo o Projeto Pedagógico do Curso (PPC), reformulado no ano de 2019 e as ementas das UC de programação.

Devido à natureza complexa do fenômeno analisado nesta tese e proporcionado pelas características inerentes ao estudo de caso, a obtenção dos dados foi dirigida por procedimentos quantitativos e qualitativos, nos dando um maior suporte e mais subsídios para aprofundar a investigação. Com isto, tendo em vista as diferentes estratégias desenvolvidas dentro do estudo de

caso e a diversidade de dados obtidos, a nossa pesquisa caracteriza-se por uma abordagem metodológica mista.

De acordo com Richardson (2017), a abordagem mista combina ou associa as abordagens qualitativa e quantitativa, de modo que a força geral de um estudo seja maior que a da pesquisa qualitativa ou quantitativa quando realizado de forma isolada, rejeitando a dicotomia entre o quantitativo e o qualitativo, uma vez que nosso intuito é construir o conhecimento frente à questão de partida e os objetivos traçados, de forma que possamos unir a realidade do mundo e nossa experiência cotidiana.

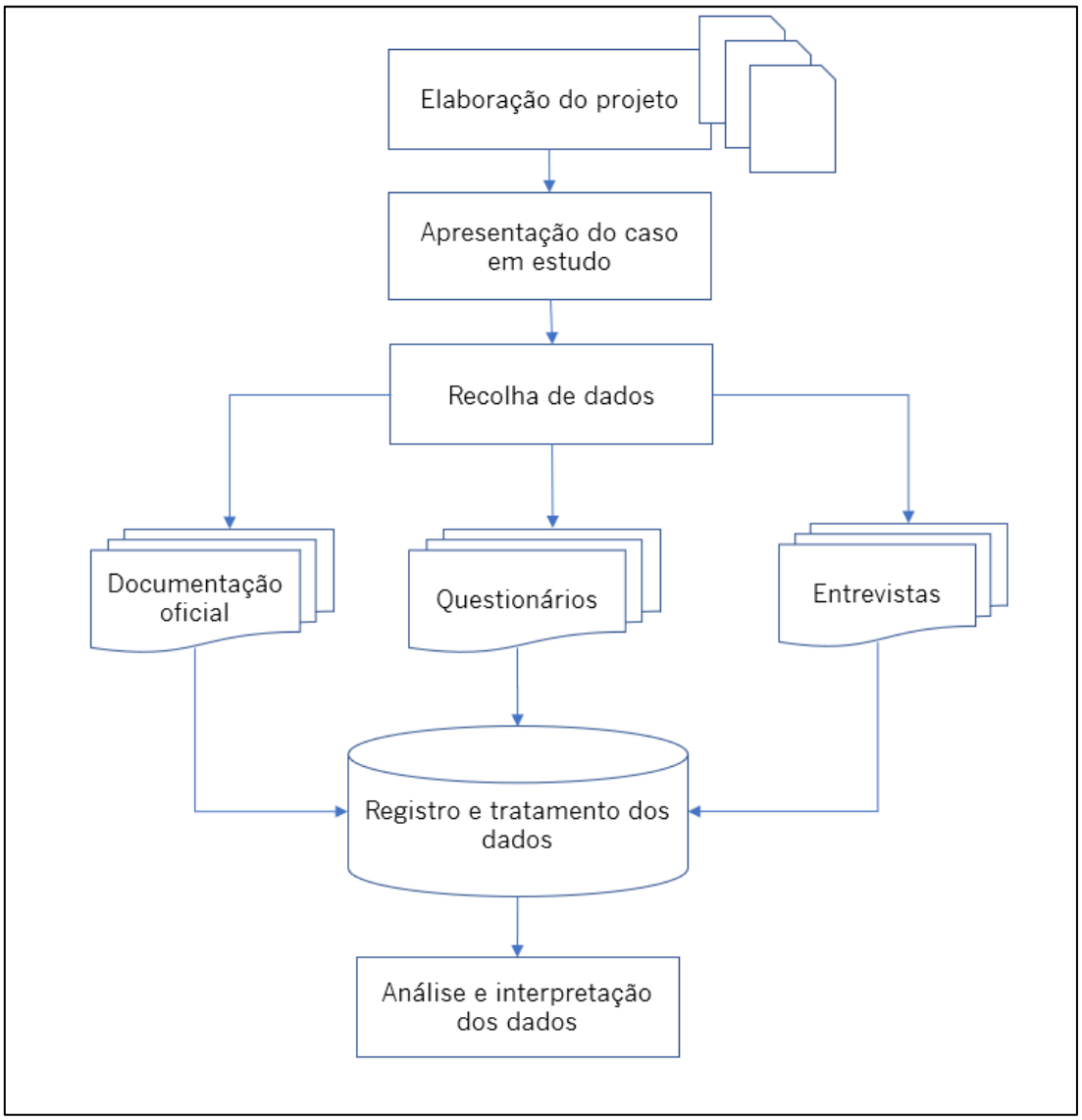
O plano de pesquisa desta investigação é a estratégia de triangulação de dados que, como definida por Stake (2005), é "considerada um processo de usar múltiplas percepções para esclarecer o significado, verificando a repetibilidade de uma observação ou interpretação" (p. 454, tradução nossa). A justificativa de usar este plano é que com ele podemos combinar pontos fortes das metodologias quantitativa e qualitativa para obter informações adicionais sobre o mesmo problema de pesquisa, a partir de diferentes percepções.

Para o nosso estudo, adotamos inicialmente uma perspectiva teórica que norteou a investigação para então iniciarmos o uso das diferentes abordagens que conduziram a triangulação dos resultados. A estratégia adotada foi necessária para que o rigor fosse garantido, de forma que pudéssemos encontrar resultados que contemplassem nossos objetivos e nossa questão de partida, especialmente diante da complexidade do contexto e do objeto de pesquisa.

3.2. Procedimento Metodológico

Nesta seção apresentamos as etapas para o desenvolvimento da nossa pesquisa em campo: elaboração do projeto, apresentação do caso em estudo, recolha de dados, registro e tratamento de dados e a análise e interpretação dos dados, apresentadas na Figura 2 e detalhadas nas subseções a seguir.

Figura 2 - *Etapas do percurso metodológico da investigação de campo*



Nota: Autoria própria.

3.2.1. Elaboração do Projeto

Para o desenvolvimento de pesquisas que envolvam seres humanos é necessária a prévia aprovação do procedimento metodológico a ser adotado pelo pesquisador por Comitê de Ética em Pesquisa (CEP). Dessa forma, seguimos todos os protocolos exigidos para a elaboração do projeto e submissão na Plataforma Brasil4.

A elaboração do projeto teve início no mês de maio do ano 2020 e a primeira versão foi submetida na Plataforma Brasil no mês de junho de 2020, contendo as seguintes informações, de acordo com as exigências da própria Plataforma: identificação, introdução, metodologia, cronograma, orçamento, resultados e divulgação dos resultados, guião dos instrumentos de recolha, carta de

4 http://conselho.saude.gov.br/plataforma-brasil-conep?view=default

anuência da unidade participante do estudo, Termo de Consentimento Livre e Esclarecido (TCLE) e os termos de autorização de imagem, áudio e vídeo.

Os instrumentos de recolha foram elaborados à luz da fundamentação teórica desta tese e validados para a realização de melhorias e modificações sugeridas durante a validação. Os questionários e entrevistas dos alunos foram validados por cinco alunos egressos do Curso e as entrevistas dos professores foram legitimadas pelos orientadores, sendo um da área de Educação e o outro da área de Informática.

Após a validação e finalização do projeto, esse foi submetido para avaliação na Plataforma Brasil e avaliado pelo Comitê de Ética em Pesquisa da Universidade do Estado do Rio Grande do Norte (CEP-UERN), que emitiu o parecer da primeira versão em julho de 2020. De acordo com o parecer, o projeto apresentou pendências estruturais, as quais foram corrigidas e uma nova versão foi gerada e submetida na Plataforma Brasil ainda em julho de 2020, tendo seu parecer consubstancial favorável para execução, emitido pelo CEP-UERN, em julho de 2020 (Apêndice A).

3.2.2. Apresentação do Caso em Estudo

De acordo com Stake (2012), "O caso é um sistema integrado. [...] Assim sendo, pessoas e programas são claramente casos em perspectiva" (p. 18). Merriam & Tisdell (2015) reforçam que para ser um caso, a entidade a ser estudada necessita ser escolhida com base em elementos que a tornem específica o suficiente para ser o caso do estudo.

Para a escolha do caso, consideramos um curso superior cuja matriz curricular possuísse UC de programação em diferentes níveis de competências e que as tenham de forma contínua ao longo do percurso acadêmico. A razão pela escolha do Curso deu-se por estar inserido no contexto brasileiro e pela possibilidade de realizar um estudo de campo em um curso que temos facilidade de acesso aos professores e alunos. Também levamos em consideração a disponibilidade e acesso a documentos oficiais do Curso para realizarmos a análise documental desta investigação.

O percurso acadêmico dos alunos do nosso caso é constituído por 43 UC obrigatórias e duas optativas, distribuídas ao longo de oito períodos letivos. Das 43 UC, cinco são de ensino de programação: Construção de Algoritmos, Estrutura de Dados, Programação Estruturada, Programação Orientada a Objetos e Programação Avançada, e pelo menos outras 14 têm como requisito a habilidade de desenvolvimento de software, embora o ensino e aprendizagem de programação não seja o foco. Compreender a estrutura do Curso e as UC que o formam ajudam a perceber o percurso acadêmico dos alunos e como acontece o processo de ensino e aprendizagem de programação, de

forma a verificarmos as diferentes metodologias adotadas, as habilidades e conhecimentos necessários e as dificuldades enfrentadas durante o processo.

3.2.2.1. Documentos. O estudo de caso não se sustenta apenas em uma fonte de evidência, uma vez que essa não seria suficiente para compreender o caso, de forma que "a análise documental torna-se, no contexto de estudos de caso, uma das técnicas relevantes que integra o processo de triangulação de dados, contribuindo assim para reforçar e dar sustentabilidade às demais fontes de evidências" (Oliveira, 2020, p. 206)**.** Para Yin (2015), os documentos apresentam detalhes específicos para corroborar com as outras fontes e auxiliam o pesquisador a fazer inferências e constatações. Além disso, documentos apresentam alguns pontos fortes, como o fato de serem estáveis (podem ser revistos), discretos (não foram criados em função do estudo do pesquisador) e exatos (possuem referências e detalhes).

Após a escolha do caso para esta pesquisa, partimos para a busca dos registros oficiais dele. Com a análise documental, buscamos perceber características do Curso, as metodologias adotadas, os conhecimentos e habilidades requeridos para o aluno aprender a programar, bem como o que se espera do profissional formado, especialmente as expectativas no âmbito da área de programação.

3.2.2.2. Participantes. Para a realização de um estudo de caso é necessário o desenvolvimento de ações de definição dos participantes e a realização da comunicação com o público-alvo para que estes conheçam o propósito da pesquisa e compreendam a necessidade de participar para contribuir com ela.

Nesse sentido, contactamos inicialmente o responsável pelo Curso, via e-mail, para apresentar o projeto aprovado pelo CEP-UERN e obter formalmente uma autorização para a realização da pesquisa no âmbito do Curso. Com o deferimento da autorização, foi realizado um convite via e-mail aos professores para participarem das entrevistas (Apêndice B).

Solicitamos os dados de contato dos alunos, nomeadamente e-mails, para que pudéssemos enviar o convite para participarem do preenchimento do questionário (Apêndice C) e das entrevistas (Apêndice D), instrumentos de recolha de dados utilizados no estudo. Além disso, os professores do Curso foram mobilizados via e-mail a fim de que ajudassem na disseminação dos questionários e estimulassem a participação de uma maior quantidade de alunos no processo de recolha de dados.

Em todos os convites realizados estavam anexos ao e-mail os TCLE, sendo um específico aos professores (Apêndice E) e outro aos alunos (Apêndice F), para que ambos tivessem conhecimento, indicando o aceite, antes de participar da recolha de dados

Para os convidados para a entrevista, além do TCLE, foram enviados também os Termos de Áudio (Apêndice G) e os Termos de Imagem (Apêndice H), que também deveriam ser lidos e assinados por cada um, autorizando, respectivamente, o uso de áudio e imagem para esta tese.

Para o nosso estudo, a definição dos participantes se deu de forma intencional, de acordo com alguns critérios. Em relação aos professores, a especificação se deu de acordo com a atuação no ensino de programação e pela aceitação do convite em participar da pesquisa. Dos 19 professores integrantes do Curso, sete atuam na docência de alguma UC de programação. Desses, apenas cinco foram contactados para participar da entrevista uma vez que os demais, na época da realização deste estudo, encontravam-se afastados do exercício de sala de aula para capacitação docente em programas de doutoramento.

Para a recolha de dados dos professores participantes, elaboramos um roteiro de entrevista contemplando perguntas que visaram caracterizar o perfil dos docentes, as metodologias adotadas no ensino e as suas percepções em relação ao processo de ensino de programação e das vivências dos alunos em relação à aprendizagem.

No caso dos alunos, o convite para participar do estudo foi com base no período letivo em que os alunos estavam matriculados e na aceitação em participar. Como a primeira UC de programação é a abordada no segundo período do Curso, o público-alvo compreendeu-se entre o segundo e o oitavo período do Curso, de forma que participaram da pesquisa apenas alunos que já estariam cursando uma UC de programação pela primeira vez, ou que já teriam cursado no mínimo uma UC de programação. Desta forma, contactamos cerca de 60 alunos, dos quais 32 aceitaram participar respondendo ao questionário e seis se disponibilizaram em participar como entrevistados. Para a recolha de dados dos alunos participantes, elaboramos um roteiro de questionário e um de entrevista, contemplando perguntas que visaram caracterizar o perfil dos alunos, suas visões acerca das habilidades e conhecimentos necessários para aprender a programar, suas opiniões acerca das metodologias do docente e de sua forma de estudar, assim como as dificuldades e desafios enfrentados no processo de aprendizagem.

Todos os participantes da pesquisa tiveram sua identidade anonimizada, para garantir a integridade e sigilo das informações, respeitando os preceitos da ética na pesquisa que envolve seres humanos. Dessa forma, os participantes dos questionários receberam o identificador Axx, em que A indica Aluno e xx a numeração de 01 até 32, de acordo com a ordem em que os questionários foram respondidos. A identificação dos alunos participantes das entrevistas é ALUx, em que ALU remete a Aluno e x é a numeração recebida de acordo com a ordem das entrevistas, que vai de 1 até 6. A

identificação dos professores, por sua vez, é PROFx, onde PROF indica Professor e xx a numeração que vai de 1 a 5 de acordo com a ordem das entrevistas.

3.2.3. Recolha de Dados

A recolha de dados é fundamental para o desenvolvimento de um estudo de caso. A partir dessa podemos ter acesso aos fatos e compreender melhor as teorias. Para o desenvolvimento do estudo de caso desta tese, tivemos que optar por instrumentos de recolha que nos permitisse coletar os dados de forma remota, face ao contexto pandêmico que vivenciamos desde meados de março de 2020, mas sem perder a acurácia que é importante para esta modalidade de estudo.

Com as aulas no contexto remoto, não foi possível desenvolver a observação participante previamente planejada, entretanto, conseguimos contactar alunos e professores que se disponibilizaram a contribuir remotamente para a investigação.

Assim, os instrumentos de recolha de dados utilizados, em conjunto, puderam fornecer diferentes tipos de dados, nos permitindo um maior aprofundamento nos conceitos adquiridos na fundamentação teórica.

3.2.3.1. Questionário. O questionário é um instrumento de coleta de dados que inclui diversas questões escritas, apresentadas aos participantes com o propósito de obter informações sobre conhecimentos, atitudes e aspectos socioeconômicos.

De acordo com Richardson (2017), um questionário tem como objetivo coletar informações que permitam classificar pessoas e suas circunstâncias, coletar informações relacionadas com o comportamento das pessoas e coletar dados sobre as atitudes ou opiniões de um grupo relacionadas com um assunto específico.

Para o nosso caso, o questionário tem como objetivo inquirir alunos e professores do Curso, para perceber as suas características, bem como seu relacionamento com as disciplinas de programação, para então compreendermos como se dá o processo de ensino e aprendizagem e verificarmos aspectos tais como as competências, habilidades e fatores necessários, as metodologias utilizadas pelos alunos e as dificuldades que eles enfrentam ao longo do cumprimento dessas disciplinas.

Assim como o contato, o formulário do questionário também foi desenvolvido no formato eletrônico, na plataforma Google, com a ferramenta Google Forms[5], facilitando o acesso e o envio das respostas por parte dos respondentes.

5 https://docs.google.com/forms/u/0/

Com a validação do questionário, verificou-se que o tempo médio que cada respondente leva para respondê-lo são 10 minutos e que todas as questões foram consideradas consistentes e contextualizadas para contribuir com a nossa pesquisa.

O questionário desenvolvido possui 12 seções, organizadas para melhor compreensão do inquirido e estruturado de forma coerente, de acordo com o conteúdo a ser respondido. A estrutura do questionário é apresentada na Tabela 6. O guião está disponibilizado no Apêndice I desta tese.

Tabela 6 - *Nome e breve descrição das seções que compõem o questionário*

Seção	**Nome da Seção**	**Breve descrição**
1	O processo de ensino e aprendizagem de programação no Ensino Superior	Faz a apresentação da investigação, o nome do projeto e da investigadora. Dá boas-vindas ao inquirido. Possui link que dá acesso ao TCLE. Dá oportunidade para que o inquirido possa dar o seu aceite ou não em participar da pesquisa.
2	Sobre você	Pergunta se o inquirido já cursou ou está cursando alguma disciplina de programação. Caso a resposta seja não, o questionário é finalizado, pois o respondente não faz parte do público-alvo.
3		Apresenta perguntas relacionadas a aspectos pessoais, tais como idade, disciplinas já cursadas e período em que está matriculado.
4 5 6 7	Sobre seu percurso acadêmico nas disciplinas de Algoritmos e Programação	Apresenta perguntas específicas sobre as disciplinas já cursadas, se o aluno já desistiu ou se já reprovou uma delas.
8	Sobre os fatores que influenciam no aprendizado de programação	Apresenta 8 perguntas, em escalas de Likert, para perceber como o aluno associa alguns fatores como influências ao processo de aprendizagem de programação.

9	Sobre as competências e conhecimentos necessários para aprender programação	Apresenta 11 perguntas, em escalas de Likert, para verificar como o aluno associa suas competências e habilidades para o seu processo de aprendizagem de programação
10	Sobre as dificuldades enfrentadas no processo de aprendizagem de programação	Apresenta 13 perguntas, em escalas de Likert, para perceber que dificuldades o aluno enfrenta durante o seu processo de aprendizagem de programação
11	Se possível, conte-nos um pouco da sua experiência nas disciplinas de algoritmos e programação!	Pergunta aberta para que o aluno possa aprofundar suas respostas, dando depoimento sobre seu percurso acadêmico.
12	Obrigada pela participação!	Seção de agradecimento ao respondente que responde o questionário em sua totalidade.

Nota*: Autoria própria.*

Para o desenvolvimento do questionário, atentamos aos campos de instruções, perguntas e respostas, deixando-os distintos. Em relação à ordem das perguntas, inicialmente são apresentadas as perguntas sobre o respondente, depois sobre seu percurso acadêmico, para então serem apresentadas perguntas mais específicas, como apresentadas na Tabela 6. O questionário foi aplicado e disponibilizado durante um mês, entre o dia 12 de agosto de 2020 e o dia 12 de setembro de 2020, durante o qual obtivemos 32 respostas.

3.2.3.2. Entrevista. O termo entrevista refere-se ao ato de 'perceber', realizado entre duas pessoas. Sendo uma das formas de desenvolver pesquisa qualitativa, a entrevista permite uma maior interação entre o pesquisador e o sujeito da pesquisa, desenvolvendo uma estreita relação entre as pessoas e caracterizando-se como um modo de comunicação no qual determinada informação é transmitida de uma pessoa A para B, podendo obter configurações do tipo unilateral ou bilateral. Na nossa investigação, a entrevista é bilateral, onde a comunicação é realizada em ambos os sentidos, e semiestruturada, possuindo estrutura flexível com questões abertas que definem a área a ser explorada.

A vantagem dessa técnica é sua flexibilidade e a possibilidade de fácil adaptação, de acordo com o contexto, tema e entrevistado. O uso da entrevista permitiu-nos estreitar os laços com o público-alvo de forma a compreender melhor como ocorre o processo de ensino e aprendizagem de programação. Os dados recolhidos foram fundamentais para a realização da análise do conteúdo que pôde esclarecer, aprofundar e complementar os conhecimentos adquiridos e amadurecidos com a fundamentação teórica.

Em relação às entrevistas realizadas com os alunos, ao todo foram seis entrevistados, os quais já haviam cursado disciplinas de programação. Para dar suporte à entrevista foi desenvolvido um guião composto de 11 perguntas abertas (Apêndice J). As entrevistas foram realizadas por meio da ferramenta Zoom, que possibilita gravação de áudio e imagem. Dos seis alunos participantes, apenas um não autorizou o uso da imagem e, neste caso, a câmera não foi ligada de forma que apenas o áudio foi gravado. Em média, as entrevistas duraram 35 minutos e todos os participantes responderam a todas as perguntas realizadas. As entrevistas foram realizadas entre os dias 3 e 12 de agosto de 2020. A Tabela 7 apresenta os dados das entrevistas realizadas com os alunos, tais como Id do aluno entrevistado, data da realização da entrevista e tempo de duração de cada entrevista.

Tabela 7 - *Dados das entrevistas realizadas com os alunos*

Id do entrevistado	Data da entrevista	Tempo de duração da entrevista
ALU1	03/08/2020	00:36:08
ALU2	04/08/2020	00:25:33
ALU3	07/08/2020	00:31:35
ALU4	10/08/2020	00:34:41
ALU5	11/08/2020	01:00:05
ALU6	12/08/2020	00:21:17

Nota*: Autoria própria.*

Em relação aos professores, ao todo foram cinco entrevistados, os quais ministram ou já ministraram alguma UC de programação no Curso. Para dar suporte à entrevista, foi desenvolvido um guião, composto de 11 perguntas abertas (Apêndice K). As entrevistas também foram realizadas por meio da ferramenta Zoom. Em média, as entrevistas duraram 60 minutos e todos os participantes responderam a todas as perguntas realizadas. As entrevistas foram realizadas entre os dias 1 do mês de fevereiro de 2021 e 4 de abril de 2021. Todos os professores autorizaram uso de som e imagem. A Tabela 8 apresenta os dados das entrevistas realizadas aos professores, tais como Id do professor entrevistado, data da realização da entrevista e tempo de duração de cada entrevista.

Tabela 8 - Dados das entrevistas realizadas com os professores

Id do entrevistado	Data da entrevista	Tempo de duração da entrevista
PROF1	01/02/2021	00:34:18
PROF2	08/02/2021	01:03:58
PROF3	10/02/2021	00:39:47
PROF4	02/03/2021	01:11:20
PROF5	04/04/2021	01:30:00

Nota*: Autoria própria.*

3.2.3.3. Análise Documental. De acordo com Oliveira (2020),

> A análise documental é uma técnica de exploração dos documentos encontrados que o pesquisador considera serem relevantes para a compreensão do seu estudo. Essa técnica possui uma dinâmica em cascata, uma vez que um documento pode levar o pesquisador a ter necessidade de conhecer outros documentos, podendo se estender durante todas as etapas da pesquisa. (p. 207)

Após verificar os documentos pertinentes para compor este trabalho, os documentos recolhidos foram o PPC do Curso (PPC, 2019), os PGCC das UC de programação e atos regulatórios. Primeiramente, a partir de um guião (Apêndice L), foi analisado o PPC do Curso, cuja última atualização foi realizada no ano de 2019. Com a análise, buscamos compreender com maior profundidade o Curso e perceber aspectos relevantes para o processo de ensino e aprendizagem de programação ali apresentados. Com a exploração dos PGCCs, esperou-se conhecer detalhes tais como estratégias adotadas pelos docentes para o ensino de programação e, por exemplo, os pré-requisitos ou conhecimentos necessários (ou importantes) para a aprendizagem daquela UC.

3.2.4. Registro e Tratamento dos Dados

Os dados obtidos por meio dos instrumentos de recolha foram registrados em ambiente local e virtual para garantir a integridade, segurança e disponibilidade. No caso dos dados coletados pelo questionário, estão registrados na ferramenta Google Drive, no próprio Google Forms, no disco rígido local e em e-mails pessoais. Para tanto, os dados foram baixados da plataforma Google Forms em formato compatível com a ferramenta SPSS, utilizada para a análise quantitativa.

Os dados obtidos nas entrevistas estão registrados em formato de MP3 e MP4 no disco rígido local e a transcrição das entrevistas estão registradas em formato de texto em disco rígido local, no Google Drive e em e-mails pessoais.

Os dados obtidos tanto nas entrevistas quanto na aplicação dos questionários precisaram ser tratados para posterior análise e interpretação. No caso das entrevistas, foi realizada a transcrição de todo o teor da entrevista, seguindo o guia de orientação para codificação e transcrição de entrevistas

(Apêndice M) baseado no protocolo desenvolvido por Oliveira (2020). A transcrição foi realizada de forma manual com auxílio da ferramenta Express Scribe Transcription Free versão 10.01 e transcrita para um arquivo no formato de texto.

O tratamento dos dados obtidos no questionário foi essencial para posterior análise no software SPSS. Para isso, foram retirados os dados incompletos (NAs) e as respostas da pergunta aberta, usadas posteriormente na análise do conteúdo. Foi ainda necessário ajustar no SPSS as categorias e as variáveis, para a realização de todas as análises.

3.2.5. Análise e Interpretação dos Dados

Após recolha e tratamento dos dados obtidos das diferentes fontes, a análise e interpretação dos dados foram executadas. As análises foram realizadas de forma a permitir a compreensão sobre o assunto e responder à questão de partida, por meio da contemplação dos objetivos traçados para esta tese.

Os dados obtidos nos questionários foram analisados com apoio da ferramenta IBM SPSS Statistic Data Editor versão 27. Já as análises qualitativas das entrevistas e dos documentos oficiais foram realizadas com o apoio da ferramenta N-Vivo 12 (Versão Release 1.5.1 (940)). Os resultados de ambas as análises estão apresentados no capítulo 4 desta tese.

Com os dados dos questionários, desenvolvemos estatísticas descritivas que nos deram subsídios suficientes para compararmos e complementarmos as informações obtidas nas análises qualitativas. Com a aplicação das entrevistas e da análise dos documentos oficiais do Curso obtivemos dados que nos permitiram realizar uma análise qualitativa. Neste sentido, optamos pela modalidade de análise do conteúdo uma vez que, de acordo com Bardin (1977):

> Pertencem, pois, ao domínio da análise do conteúdo, todas as iniciativas que, a partir de um conjunto de técnicas parciais e complementares, consistam na explicitação e sistematização do conteúdo das mensagens e da expressão deste conteúdo, com o contributo de índices passíveis ou não de quantificação, a partir de um conjunto de técnicas que embora parciais são complementares. (p. 42)

Ao longo das análises das respostas, fomos categorizando-as de acordo com as classificações pré-definidas, emergidas da fundamentação teórica. Após o desenvolvimento da categorização, partimos para a análise do conteúdo, tendo como fundamentação os conceitos apresentados por Bardin (1977), que apresenta a categorização, descrição e interpretação como etapas essenciais numa pesquisa qualitativa, e que busca sentido no conteúdo das mensagens por meio da interpretação dos indicadores selecionados na pesquisa (Bardin, 1977).

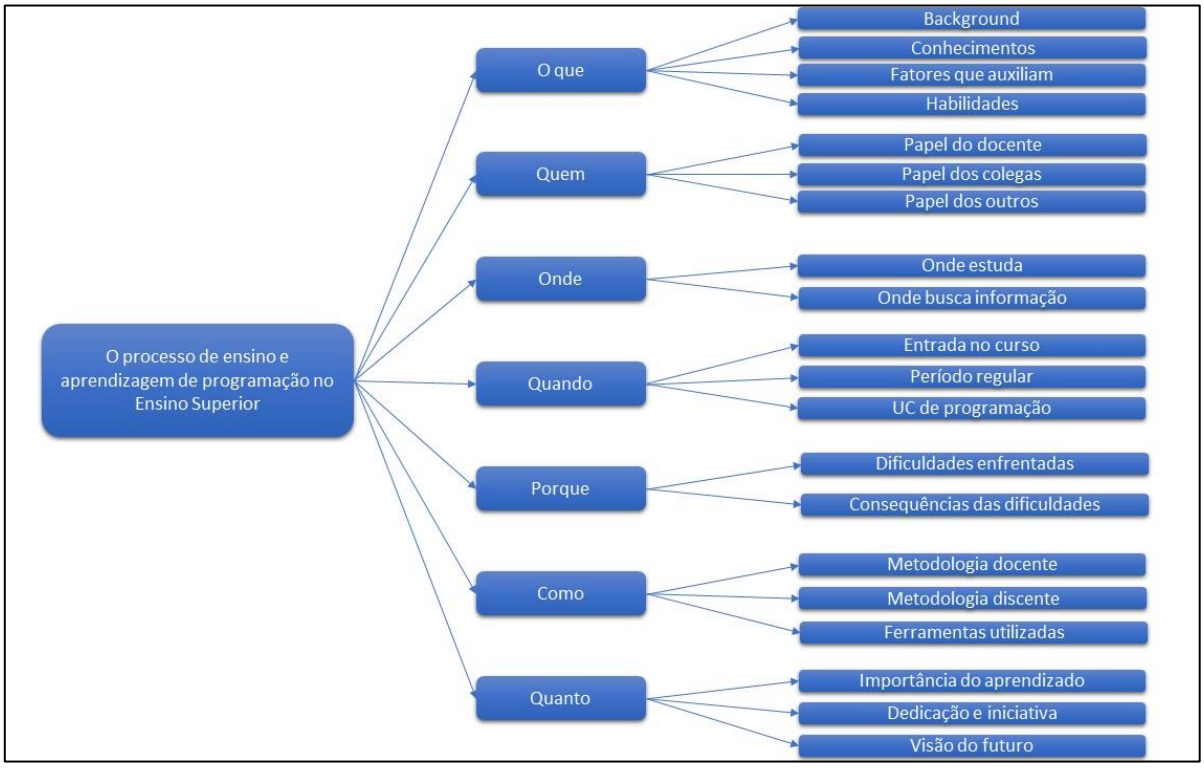
A partir do desenvolvimento de mapas conceituais, que refletiam os objetivos da nossa pesquisa, pudemos refinar e caracterizar os parâmetros que compõem o processo de aprendizagem. Para caracterizar e categorizar esses parâmetros, utilizamos o *framework* 5W2H (do inglês *what, who, where, when, why, how, how much*) como suporte para orientar nossa investigação. Para nossa tese, utilizaremos a tradução do 5W2H, de forma a ser 'O que', 'Quem', 'Onde', 'Quando', 'Porque', 'Como' e 'Quanto', como apresentamos a seguir.

3.2.6. **Framework 5W2H**

O *framework* 5W2H tem seu uso na área de negócios e estratégias, bem como estudos na área de gamificação (Klock et al., 2016). Também foi utilizada para desenvolvimento da linguagem Q7 (Leite et al., 2005) para o desenvolvimento orientado a aspectos e adaptada para o desenvolvimento de processo de reuso de requisitos de software (Morais, 2010).

Tendo em vista sua flexibilidade, o *framework* pode ser utilizando nos mais diversos contextos, inclusive no âmbito da educação, podendo ser adaptado de acordo com a especificidade de cada situação. Para nossa investigação, os sete parâmetros estão interligados e cada um possui diferentes categorias, as quais nos guiarão para a categorização das análises das unidades de registro obtidos com as fontes de dados. A Figura 3 apresenta os parâmetros e suas respectivas categorias.

Figura 3 - *Parâmetros e respectivas categorias de análise*



Nota: Autoria própria.

Para o nosso contexto, cada um dos parâmetros está definido da seguinte forma:

Parâmetro ‘O que’**:** define os fatores, as competências, os conhecimentos e as habilidades necessárias para a aprendizagem de programação, fatores estes importantes para que o aluno obtenha êxito. Além disso, apresenta as vivências anteriores dos alunos, seu *background* e como este pode influenciar no processo de aprendizagem de programação.

Parâmetro ‘Quem’**:** indica os participantes do processo de ensino e aprendizagem de programação no nível superior. Consideramos, portanto, analisar o papel dos professores, dos colegas e de outros, nas percepções de professores e alunos.

Parâmetro ‘Onde’**:** define o local em que ocorre o processo de aprendizagem investigado, com o objetivo de compreender como e onde os alunos buscam informações e como e onde os professores geralmente indicam ou recomendam que os alunos realizem essas buscas, de forma de ser um complemento para além do que é visto em sala de aula.

Parâmetro ‘Quando’: define quando ocorre o contato dos alunos e professores com a programação. Para o nosso escopo, definimos que a pesquisa é realizada com participantes que se encontram ainda no processo de aprendizagem, isto é, que estão ainda matriculados e que já viram pelo menos uma UC de programação. Dessa forma, é importante conhecer o período em que os estudantes se encontram matriculados na altura desta pesquisa e as UC por esses cursadas. Em relação aos professores, este parâmetro indica quando eles ministram programação, em que UC e há quanto tempo.

Parâmetro ‘Porque’: apresenta a justificativa de estarmos investigando o processo de aprendizagem de programação. Revela as dificuldades e os desafios enfrentados pelo aluno e turmas de alunos, e o que pode ser acarretado por estes desafios durante o percurso acadêmico.

Parâmetro ‘Como’: ajuda a conhecer a forma como é feita a interação entre os professores e alunos, as metodologias, estratégias e ferramentas utilizadas por professores para ensinar e utilizadas pelos alunos para aprender a programar.

Parâmetro ‘Quanto’: aponta o que os envolvidos no processo de aprendizagem pretendem atingir. Qual a importância da aprendizagem de programação, suas atitudes diante do processo de aprendizagem, bem como a visão do futuro.

3.3. Síntese Conclusiva do Capítulo

Esse capítulo apresentou a abordagem metodológica e o procedimento metodológico da pesquisa desenvolvida. Percebemos que, devido à natureza complexa do objeto de estudo, para responder à questão de partida e para alcançar os objetivos definidos, foi desenvolvido um estudo de caso com vista a analisar ‘como ocorre o processo de ensino e aprendizagem de programação no

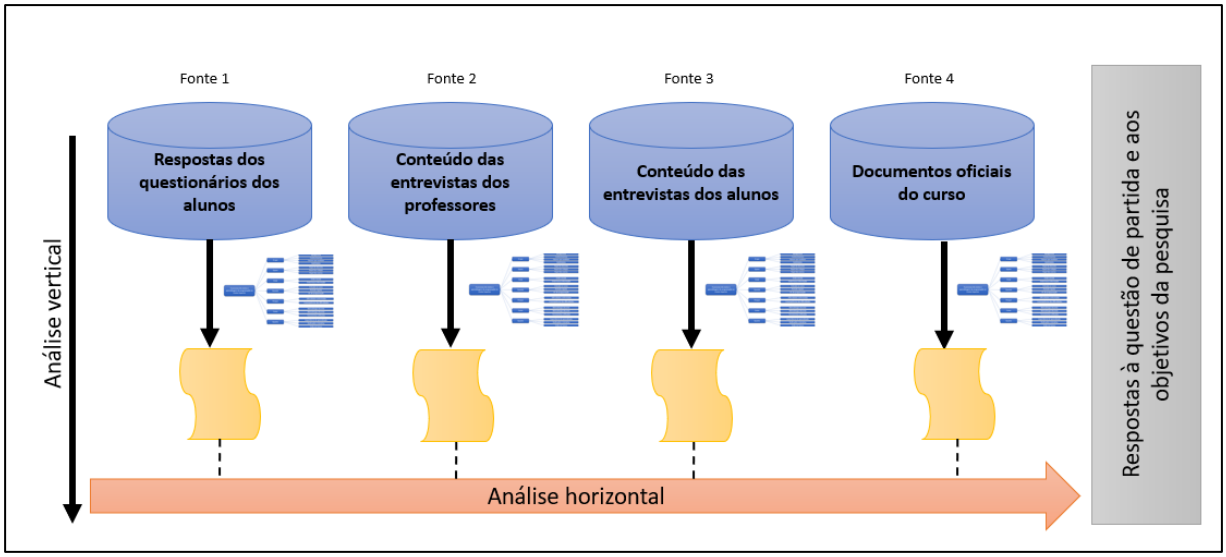
Ensino Superior'. Para tanto, usamos diferentes estratégias quantitativas e qualitativas para a obtenção dos dados a serem analisados com a triangulação, de acordo com os conceitos da abordagem metodológica mista.

O capítulo abordou todas as etapas realizadas no procedimento metodológico, que incluem a elaboração do projeto apresentado e aprovado pelo CEP-UERN, a apresentação do estudo de caso incluindo a justificativa e características do caso, a recolha de dados apontando os instrumentos de recolha adotados, o registro e tratamento de dados apresentando as ferramentas usadas e como os dados foram armazenados, e a análise e interpretação dos dados, a fim de apresentarmos as estratégias adotadas para contemplar a abordagem metodológica do nosso estudo. Apresentamos ainda o uso do *framework* 5W2H para apoiar a categorização das análises das unidades de registro obtidos com as fontes de evidências desse estudo. A partir da abordagem metodológica e do procedimento metodológico, desenvolvemos o estudo de caso apresentado no capítulo 4.

4. Estudo de Caso

Neste capítulo apresentamos o estudo de caso com a finalidade de compreendermos, a partir da triangulação dos dados, como ocorre o processo de ensino e aprendizagem de programação no Ensino Superior. Para cada uma das fontes de dados, realizamos a categorização das unidades de registro com base nas categorias definidas na fundamentação teórica. Com isso, foi gerado um arcabouço de dados organizadas a partir dos parâmetros 'O que', 'Quem', 'Onde', 'Quando', 'Porque', 'Como' e 'Quanto' que, ao serem sobrepostos, nos permitiram realizar a triangulação desses dados. Dessa forma, em um primeiro momento, realizamos uma análise vertical em cada uma das fontes, para posteriormente percebermos horizontalmente como os dados se entrelaçam, convergem ou divergem, de acordo com as nossas interpretações à luz da fundamentação teórica, como apresenta a Figura 4.

Figura 4 – *Representação do processo de triangulação de dados*



Nota: Autoria própria a partir de Oliveira (2020, p. 398).

O capítulo está organizado em duas seções: 'caracterização do caso' e 'percepções, vozes e palavras dos participantes do processo'.

4.1. Caracterização do Caso

Criado em 1998, o Curso estudado tem como objetivo formar profissionais preparados para especificar, conceber, desenvolver, implementar, adaptar, produzir, instalar e manter sistemas computacionais, bem como perfazer a integração dos recursos físicos e lógicos necessários ao atendimento das necessidades computacionais de organizações em geral, comprometidos com a ética profissional e com o desenvolvimento tecnológico (PPC, 2019).

A estrutura curricular do Curso está fundamentada em quatro grandes áreas de formação conforme orientação das Diretrizes Curriculares Nacionais (DCN) [no contexto brasileiro] da área de Computação:

a) Formação básica: Ciências Exatas; Lógica e Programação de Computadores; Lógica Digital e Eletrônica Digital.
b) Formação humanística: conhecimentos na área de Ciências Humanas e Sociais.
c) Formação complementar: conhecimentos nas áreas de Administração, Economia, Direito.
d) Formação tecnológica: Banco de Dados; Microprocessadores; Redes de Computadores; Sistemas Operacionais; Estruturas de Dados; Engenharia de Software e Inteligência Artificial.

Dessa forma, o nosso estudo está inserido na grande área formação básica (a), que contempla as UC de Lógica e Programação de Computadores.

Sobre o corpo docente, o Curso é composto por 19 professores, oriundos, além de cursos de Ciência da Computação, também de cursos de áreas afins, tais como Engenharia Elétrica e Engenharia de Computação. Também existem profissionais de outras áreas, tais como Física e Matemática, mas com pós-graduação em Computação. Tais professores além de comprovada experiência acadêmica, tanto no ensino como na pesquisa, também têm experiência prática, de atuação no mercado de trabalho (PPC, 2019).

Em relação à infraestrutura do Curso, este possui recursos próprios, tais como laboratórios, salas de professores, sala de reunião, além de alguns recursos físicos e equipamentos compartilhados pelos demais departamentos. Ao todo são seis laboratórios, dos quais três são de uso geral e três especializados para seus grupos de pesquisa.

Em relação às UC de programação do Curso, apresentadas em seu PPC (PPC, 2019, pp. 33-34), estas estão distribuídas entre o segundo e o quinto período. Cada uma possui seus pré-requisitos, carga-horária e os objetivos a serem atingidos, conforme consta em seus referidos PGCC. A Tabela 9 apresenta a caracterização das UC:

Tabela 9 – *Caracterização das Unidades Curriculares de programação*

Unidade Curricular	Carga horária (T*/P**)	Período	Pré-requisito	Objetivos
Construção de Algoritmos	90 horas 45T/45P	2°	Lógica Matemática aplicada à Computação	Proporcionar que o aluno aprenda a desenvolver algoritmos, implementando soluções segundo uma linguagem de programação
Estrutura de Dados	60 horas 30T/30P	3°	Construção de Algoritmos	Possibilitar ao aluno a aquisição de conhecimentos e habilidades necessárias à construção e manipulação de estruturas de dados que garantam a maior clareza, eficiência e coerência ao algoritmo que dependam destas estruturas.
Programação Estruturada	60 horas 30T/30P	3°	Construção de Algoritmos	Aprofundar o conceito de programação visto na disciplina de construção de algoritmos, bem como apresentar novos algoritmos e técnicas de resolução de problemas iterativas e recursivas, realizando experiências na construção de programas com interface gráfica com usuário.
Programação Orientada a Objetos	60 horas 30T/30P	4°	Construção de Algoritmos	Fornecer ao aluno os conceitos do paradigma de orientação a objetos e, ao final da disciplina, possibilitar ao aluno o desenvolvimento de habilidades para construção de programas utilizando a linguagem de programação Java e modelar aplicações orientadas a objeto através de diagramas UML
Programação Avançada	60 horas 30T/30P	5°	Programação Orientada a Objetos	Abordar os conceitos avançados de uma linguagem de programação, principalmente no tocante a: sincronização entre processos, exceções, programação distribuída e conectividade com banco de dados, fornecendo anteriormente a isso conceitos sobre padrões de projeto

Nota: *Teórica. **Prática. Autoria própria.

Vendo as características de cada uma das UC, podemos entender que são diferentes os conhecimentos e habilidades exigidos para aprender a programar, as dificuldades enfrentadas e as metodologias de ensino e aprendizagem que podem ser adotadas ao longo do processo.

Uma vez que apresentamos nesta seção o Curso, sua origem, os objetivos e mais especificamente sua estrutura e as UC de programação, a seção a seguir apresenta a triangulação dos dados das percepções, vozes e palavras dos participantes do processo de ensino e aprendizagem de programação.

4.2. Percepções, Vozes e Palavras dos Participantes do Processo

Compreender as percepções, vozes e palavras dos envolvidos no processo de ensino e aprendizagem de programação é um passo fundamental para o nosso estudo. As percepções dizem respeito aos dados coletados dos alunos respondentes dos questionários (Apêndice N), as vozes, por

sua vez, dizem respeito ao que nos falaram os seis alunos (Apêndice O) e os cinco professores (Apêndice P) nas entrevistas e as palavras são fruto das análises de documentos oficiais do Curso. Para facilitar a análise e interpretação dos dados, cada unidade de registo está enumerada de forma sequencial de 1 a n, para cada um dos participantes, de acordo com a ordem em que as unidades foram categorizadas. A Tabela 10 explica o formato da referenciação às unidades de registro e a interpretação.

Tabela 10 – *Estrutura de referência às unidades de registro*

Forma de participação	**Id**	**Primeira unidade de registro**	**Última unidade de registro**	**Exemplo de referência**	**Interpretação**
Questionário	Axx	.1	.n	A02.2	Segunda unidade de registro do aluno 02, respondente do questionário
Entrevista	ALUx	.1	.n	ALU1.10	Décima unidade de registro do aluno 1, participante da entrevista
Entrevista	PROFx	.1	.n	PROF3.1	Primeira unidade de registro do professor 3, participante da entrevista

Nota: Autoria própria.

As análises estão organizadas nos parâmetros 'O que', 'Quem', 'Onde', 'Quando', 'Porque', 'Como' e 'Quanto', fruto da triangulação dos dados e relacionadas às categorias de cada parâmetro. A partir da estruturação das categorias, realizamos a análise dos questionários e a leitura flutuante dos depoimentos coletados e dos documentos oficiais do Curso, como forma de nos apropriar do contexto. Os dados foram analisados com base nas categorias e posteriormente identificamos as ideias centrais para então apresentarmos as análises e interpretações obtidas.

4.2.1. Parâmetro 'O Que'

O parâmetro 'O que' está estruturado em quatro categorias: 'background', 'conhecimentos', 'fatores que auxiliam' e 'habilidades'.

4.2.1.1. Background. Esta categoria diz respeito à formação e experiências dos alunos antes de ingressar no Curso. O objetivo é percebermos estas podem influenciar o processo de aprendizagem de programação. Para coletar os dados dos questionários para esta categoria, fizemos as seguintes afirmações:

1- A minha experiência com programação me ajudou a aprender a programar.

2- Foi necessário que eu tivesse conhecimento em matemática para aprender a programar.

Para a primeira afirmação, o aluno é levado a refletir sobre a sua vivência com a programação anteriormente ao Ensino Superior e se esta influencia ou não no seu processo de aprendizagem. De acordo com Aureliano et al. (2016), Eltegani & Butgereit (2015) e A. Gomes & Mendes (2010), o

conhecimento prévio em programação e a experiência anterior com a programação são fatores que influenciam na aprendizagem. 72% dos alunos participantes concordam com a fala dos autores. De acordo com o aluno A23, o fato de ter conhecimento em programação antes de cursar as UC permitiu ter maior facilidade em aprender a programar.

Os alunos ingressantes em um curso superior possuem um grande desafio pela frente, desde a sua adaptação a uma nova rotina de estudo, até aprender e se deparar com conteúdo nunca vistos anteriormente (Ambrósio et al., 2011), apesar de que, de acordo com o PPC do Curso, "é comum o ingresso de alunos que já atuam no mercado informalmente, ou apenas com formação técnica" (PPC, 2019, p. 17), indicando que é comum os alunos terem experiência anterior, mesmo sem ter formação na área.

Os alunos participantes das entrevistas contemplam diferentes contextos de escolas anteriores: quatro vieram de escolas particulares e dois de escolas públicas, sendo um deles advindo de um curso técnico na área de Informática.

Dos seis alunos entrevistados, apenas o advindo do IFRN teve aulas de programação em sua formação escolar. Os demais alunos nunca tinham visto programação em seu percurso acadêmico, mas alguns relataram ter alguma noção básica por ter tido contato anterior, por conta própria, como relatado por dois alunos:

> Quando eu entrei eu tinha uma noção básica do que eram as coisas. Mas é claro que aquela noção básica que eu tinha lá quando entrei no primeiro período não era nada que fosse a realidade dos períodos que vinham depois [risos]. Mas eu nunca tinha tido nenhuma experiência com programação mesmo, mas antes de entrar eu já fazia alguns *sitezinhos* em HTML [*HyperText Markup Language*], essas coisas, eu sempre gostei dessas coisas [...] mas em relação à programação mesmo eu nunca tinha mexido não antes de entrar no Curso. (ALU1.1)

> Antes do Curso eu tive contato com programação uma vez, que o meu primo ele fazia o curso de Ciência da Computação, então eu já vi ele programando algumas vezes e pelo que eu me lembro, faz anos, faz uns cinco anos eu acho, só [fiz] um *Hello World* na tela e acho que não passou disso na época. (ALU6.1)

As experiências vividas pelos ALU1 e ALU6 antes de entrar no Ensino Superior são um pouco diferentes. O ALU1 conheceu HTML e chegou a desenvolver sites, ou seja, teve experiência prática, embora os considere *sitezinhos*, ou seja, para ele algo sem muita relevância. Já o ALU6 teve experiência como expectador, vendo alguém programando algo e teve uma única experiência de prática

desenvolvendo um *Hello World*[6]. Entretanto ambas as experiências trazem para os alunos um diferencial em relação a outros que nunca viram programação.

O *background* dos alunos ALU2, ALU3 e ALU4, é de nenhum contato anterior com a programação. O ALU2 ingressou na Universidade em meio a uma greve e, demonstrando espírito de autonomia e proatividade, buscou conhecer detalhadamente as UC ofertadas no Curso. Vendo que no Curso teria programação, buscou estudar para o início das aulas. De acordo com ALU2,

> De conhecimento eu não tinha nenhum, quando eu entrei na [nome da Instituição omitida] assim de programação, nada, nada, nada... eu tinha visto Pascal... eu acho que a maioria dos alunos inclusive, entram assim meio sem saber né programação [...] mas eu tinha visto alguma coisa de Pascal já antes mas eu não tinha muita noção. [...] porque, por exemplo, eu fazia escola particular, eu não tinha noção do que era Ciência da Computação. (ALU2.1)

O relato do aluno ALU2 traz à tona que, assim como ele, os demais colegas de sua turma também não possuem conhecimento anterior em programação, antes de ingressar no Ensino Superior. E ainda apresenta a falta de direcionamento nas escolas, que poderiam auxiliar os alunos na escolha do curso superior. Algo semelhante é relatado pelo aluno ALU4, também advindo de uma escola particular. O aluno apresenta saber que existe programação, mas não tinha noção de como os programas são desenvolvidos:

> Bom, é... [pausa] antes do Curso, antes de passar no SiSU [Sistema de Seleção Unificada], eu não tinha nada assim, eu sabia que existia as coisas, mas eu não sabia o que era uma linguagem, por que que tinha, como era feito, essas coisas. (ALU4.2)

O ALU3 relata nunca ter tido contato com programação, como apresentado a seguir:

> Nenhum [contato]! Logo quando eu comecei o Curso que eu via meus colegas sempre falando de programação, que passavam a noite programando, e não sei o que... eu ficava 'meu Deus do céu, o que é isso?', eles falam tanto de programação e eu aqui sentado sem saber nem o que eles estão falando. (ALU3.2)

Além da falta de experiência anterior de ambos, nota-se ainda a heterogeneidade da turma em que o ALU3 estava inserido: enquanto ele não possuía conhecimento algum em programação, seus colegas de classe já tinham experiência e contato constante com a programação.

Na segunda afirmação, buscamos conhecer sobre a importância do conhecimento em matemática para aprender a programar (Ambrósio et al., 2011; E. O. da Silva & Falcão, 2020). Dos

[6] O *Hello World* é bastante conhecido pelos programadores. É utilizado tanto para obtermos o primeiro contato com a linguagem de programação, quanto como uma forma de verificar se o ambiente está preparado adequadamente. Além disso, é um recurso importante para facilitar o entendimento básico de programação.

alunos participantes do questionário, 62,5% concordam com o que dizem os autores. Entretanto, percebemos que alguns alunos não reconhecem a importância da matemática para a programação, o que pode ser explicado por que alguns alunos só compreendem a necessidade de ter conhecimento em matemática em programação quando se trata de resolução de problemas matemáticos ou que envolvam cálculos de forma explícita (Ambrósio et al., 2011).

Para os professores entrevistados, o conhecimento em matemática é fundamental para a aprendizagem de programação. Para o PROF1 "durante o percurso o que é essencial é matemática" (PROF1.3). Para o professor PROF2,

> Ele [o aluno] precisa ter uma formação matemática, necessária para se for o caso, o problema requerer alguma coisa de matemática, ele ter a condição de pelo menos identificar 'ah, isso aqui requer o conceito de função, ou então, ah isso aqui requer o conceito de geração randômica, ou precisa você usar uma combinação linear que da álgebra resolve o problema'. (PROF2.4)

O PROF3 endossa que,

> Ele [o aluno] precisa ter uma base matemática muito forte, porque a matemática ajuda a você a raciocinar algoritmicamente. Não é que algoritmo é matemática, mas a forma de raciocínio é muito próxima pra você desenvolver, porque os problemas matemáticos você segue regras, a gente segue regras pra resolver, e a gente aplica algoritmos pra solucionar eles, mesmo quando a gente está resolvendo manualmente. Então se você tem uma base matemática forte, você está acostumado a pensar algoritmicamente mesmo que você não saiba que é algoritmicamente. (PROF3.1)

Sobre as vozes dos alunos, para o ALU1, a matemática não faz muita diferença para aprender a programar e cita a necessidade apenas para a resolução de problemas que envolvam cálculos ou funções matemáticas, ou ainda apenas para Computação Gráfica. O aluno ALU4 concorda com os professores e autores ao citar "acho que ajuda muito você ver a matemática [anteriormente]" (ALU4.12).

4.2.1.2. Conhecimentos. Na categoria 'conhecimentos' fizemos as seguintes afirmações:

1- A minha compreensão em relação aos conceitos básicos de algoritmos e programação me ajudou a aprender a programar.
2- Precisei ter a compreensão da sintaxe e da estrutura do programa para aprender a programar.
3- Foi necessário que eu tivesse pensamento computacional para aprender a programar.

A intenção da primeira afirmação é perceber se, ao compreender os conceitos básicos de algoritmos, que é pré-requisito para as UC Estrutura de Dados, Programação Estruturada e Programação Orientada a Objetos, o aluno tem mais facilidade em aprender a programar, como defendido por Franzen et al., (2018) e G. Santos et al. (2020). Como resultado, verificamos que apenas 12,5% dos alunos discordam dos autores, demonstrando que a grande maioria (87,5%) compreendem que os conhecimentos em algoritmos são necessários para aprender a programar.

Para o aluno A14, "O fato de ter cursado a disciplina de Lógica um semestre antes de cursar a disciplina de Construção de Algoritmos ajudou bastante no processo de absorção e assimilação do conteúdo" (A14.1). Para o aluno ALU4 "em algoritmos [...] eu tive uma base, teve um pouco de 'portugol', um pouco de algoritmos, como fazer um fluxograma que tipo já ajuda a ter uma base de conhecimento" (ALU4.7).

De acordo com o PROF1, a aprendizagem de programação necessita que o aluno compreenda o que está fazendo e isso depende da compreensão dos conceitos básicos:

> Quando você programa qualquer coisa, tem que saber por que você está programando, por que você está fazendo aquele *insert*, porque está colocando aquela biblioteca lá, porque está declarando aquela variável como um inteiro, um *float* [...] por menor que seja o seu exemplo, mas você tem que saber, você tem que dominar o porquê daquilo ali. (PROF1.1)

Na opinião do professor PROF3 "se a pessoa conseguir pensar em forma de algoritmo, ela consegue desenvolver qualquer algoritmo" (PROF3.50). As falas dos professores e alunos convergem para a necessidade de conhecer os conceitos básicos para aprender a programar, corroborando com os autores Franzen et al., (2018) e G. Santos et al. (2020).

A segunda afirmação diz respeito à percepção do aluno em relação a ter conhecimento específico na sintaxe e semântica de um programa para aprender a programar (Aparicio & Costa, 2018; Chang et al., 2000; Costa, 2019; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015; Ortiz-Ortiz et al., 2018; A. Santos et al., 2010, 2013; Skalka & Drlik, 2018; Souleiman, 2017). Dos 32 participantes da pesquisa, 31 concordaram com a afirmação. De forma oposta, o professor PROF3 acredita que o ensino de programação deve ser realizado a partir dos conceitos básicos, mostrando para que servem alguns comandos, desenvolvendo o raciocínio lógico, sem focar em uma sintaxe de linguagem de programação. A ideia do professor é "fazer de uma maneira que os alunos aprendam a programar, não aprendam uma linguagem, [...] o meu enfoque é para que eles consigam fazer um *if* ou um laço em qualquer linguagem de programação" (PROF3.44).

A terceira afirmação é sobre o pensamento computacional e a necessidade deste no processo de aprendizagem. De acordo com as respostas dos alunos, 8 concordam totalmente (25%), 18 concordam em parte (56,3%) e 6 discordam em parte (18,8%). Dessa forma, a maioria dos alunos corrobora com os estudos de Silva & Falcão (2020), Wing, 2014, Al-Imamy et al. (2006), Aparicio & Costa (2018), T. Castro et al. (2011), R. M. S. Pereira (2017) e Rezende & Bispo (2018).

Apesar de nenhum dos professores ou alunos entrevistados citar o termo 'pensamento computacional', os professores citam que para aprender a programar é preciso "pensar diferente" (PROF3.52), "moldar a forma de pensar" (PROF4.1) e "pensar algoritmicamente" (PROF3.1). O professor PROF3 usa o termo "lógica computacional" (PROF3.1) para se referir a uma necessidade básica para aprender a programar. O PPC do Curso aponta como um dos seus objetivos "Capacitar discentes para definir o pensar/fazer a partir da compreensão lógica da dinâmica e das necessidades empresariais e institucionais, visando o projeto e construção de sistemas através do pensamento computacional, com criatividade e competência" (pp. 9-10).

4.2.1.3. Fatores que Auxiliam. Os fatores que auxiliam podem ser intrínsecos ou extrínsecos aos alunos. Para esta categoria fizemos duas afirmações:

1- Foi necessário que eu fosse proativo para aprender a programar.

2- Precisei ter autonomia para aprender a programar.

A intenção da primeira afirmação é perceber se o aluno considera a proatividade como um fator necessário para aprender a programar, como apresentado por Chuchulashvili et al. (2016) e Mendes et al. (2012). Os dados obtidos com os questionários apontam que 85% dos alunos concordam com o que dizem os autores. Para o PROF1,

> Programação não é algo que do dia para a noite você vai estar um super programador, não! Tem que persistir, você tem que ser curioso, você tem que fazer seu programa lá, botar para rodar, deu defeito? Vai lá nos *logzinhos*, nas mensagens ver onde é que está o erro, procura entender o porquê daquele erro, e aí é nesse sentido aí, você tem que ser persistente, tem que ser curioso, e apesar do nome proativo ser um nome que bota muito medo né, hoje em dia, 'ah, eu vou ter que ter proatividade para poder me sobressair', mas proatividade é só você pensar no que é que você pode fazer a mais do que é cobrado em sala de aula. (PROF1.8)

O professor PROF4 corrobora com os autores ao falar que "[é necessário] ser proativo pra não ficar só esperando o conteúdo ou não se contentar com aquele conteúdo que é passado só na aula, na sala de aula e nos exercícios, mas ficar procurando mais coisas" (PROF4.8). A proatividade é ainda citada pelos professores PROF1 e PROF2 em suas falas. O PPC cita a proatividade fazendo uma

referência ao profissional de Informática: “É importante que esse profissional seja curioso, proativo, crítico, dedicado, persistente e que tenha raciocínio lógico e elevada capacidade de abstração” (PPC, 2019, p. 12).

A segunda afirmação versa sobre a autonomia, citada por Al-Imamy et al. (2006) e Chuchulashvili et al. (2016) como um fator importante para aprender a programar. Os dados dos questionários apontam que apenas um dos alunos respondentes discorda desta afirmação. O PPC também traz a autonomia como um fator importante para os profissionais da área de Informática, versando que é necessário ter “elevado grau de independência que o permita a aquisição de conhecimento complementar de forma autônoma e de realizar as determinadas ações sem auxílio externo” (PPC, 2019, p.13).

4.2.1.4. Habilidade. Na categoria ‘Habilidade’ apresentamos duas afirmações:

1- Precisei ter habilidade em resolver problemas para aprender a programar.

2- Foi necessário que eu tivesse pensamento lógico para aprender a programar.

Sobre a primeira afirmação, buscamos verificar a percepção dos alunos em relação à necessidade de ter a habilidade de resolução de problemas (Ambrósio et al., 2011; Dijkstra, 1989; Gomes, 2010; Medeiros, 2019; Medeiros et al., 2020; Oliveira et al., 2015; da Silva & Falcão, 2020; da Silva et al., 2015; Villalobos et al., 2009; Aparicio & Costa, 2018; Bosse & Gerosa, 2017; T. Castro et al., 2011; T. H. Castro et al., 2008; Chang et al., 2000; Chuchulashvili et al., 2016; Costa, 2019; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015; Nawahdah et al., 2015; T. A. N. de Oliveira & Rebouças, 2018; Ortiz-Ortiz et al., 2018; Rezende & Bispo, 2018; A. Santos et al., 2010, 2011, 2013; Skalka & Drlik, 2018; Souleiman, 2017), uma vez que é um dos principais objetivos da programação. A opinião dos alunos mostra que 93,75 dos entrevistados concordam com o que os autores apresentam.

Todos os professores entrevistados citam a importância de se ter a habilidade de resolução de problemas. Para o professor PROF2, “se você não souber resolver um problema, você não sabe programar para que o computador resolva” (PROF2.14) e “você sente que está aprendendo a programar é quando você enfrenta o desafio de resolver o problema, e consegue resolver” (PROF2.2). Na opinião do PROF5, “a principal competência que alguém tem de entender para aprender a programar é a capacidade de estruturar a resolução de problemas” (PROF5.1). O PPC do Curso complementa que

> De uma forma mais ampla a avaliação da aprendizagem nos cursos de Ciência da Computação um dos objetivos está definido em torno de capacitar o aluno a apresentar

> soluções para uma séria de problemas encontrados no mundo real das pessoas e das organizações. (PPC, 2019, p 87)

Em relação à segunda afirmação, da necessidade de ter pensamento lógico, 90% dos respondentes do questionário concordam com a afirmação, sendo que destes, 72% concordam totalmente, corroborando com os autores apresentados na fundamentação teórica (Ambrósio et al., 2011; Araújo et al., 2019; Gomes, 2010; Oliveira et al., 2015; Santos et al., 2015; Santos et al., 2020; Al-Imamy et al., 2006; Bosse & Gerosa, 2017; T. Castro et al., 2011; T. H. Castro et al., 2008; Costa, 2019; T. A. N. de Oliveira & Rebouças, 2018; Ortiz-Ortiz et al., 2018; Rezende & Bispo, 2018; Skalka & Drlik, 2018).

O ALU4 cita a lógica como um conhecimento necessário para aprender a programar, e acrescenta que ao adquirir conhecimento em lógica pôde ir também obtendo novos conhecimentos importantes para aprender a programas. Em sua fala, o aluno expressa que conseguiu modificar sua percepção sobre programação, a partir de compreender a lógica:

> aí a partir daí eu comecei a ter mais ou menos um pouco da lógica, e a partir daí foi meio que moldando minha cabeça a pensar como estruturar um código, ou seja, o que é que eu vou precisar, como vai ser o algoritmo daquele até chegar o resultado de um problema, como solucionar [...], foi melhorando como estruturar alguns tipos de problemas, estruturar a lógica é importante para resolver aquele problema, como declarar uma variável, alguma coisa do tipo, coisas simples assim. (ALU4.4)

O professor PROF1, em sua fala, aborda que “O que é essencial é matemática, evidentemente, e lógica, tanto a lógica matemática por assim dizer, como o raciocínio lógico” (PROF1.3). Para o PROF2, “primeiro lugar tem que ter a questão da lógica, eles têm que entender o problema, e quais os passos para resolver. Isso aí é a questão de você organizar o raciocínio lógico de atacar um determinado problema” (PROF2.3). O professor PROF3 por sua vez nos diz que “A lógica de computação, a lógica computacional, a lógica de Boole essa que se estuda em disciplinas de lógica, eu acho que é importante se ter algum domínio” (PROF3.1), embora não ache que a lógica seja essencial para o processo de aprendizagem.

Os professores citam outras habilidades importantes tais como trabalhar em grupo (PROF1.6-7), ter atenção (PROF1.8), ser curioso (PROF1.8), ser criativo (PROF1.8; PROF4.9) e ter disciplina (PROF4.9). O ALU4 cita ainda como habilidades: entender o processo de desenvolvimento de software, entender como melhorar o código, saber organizar o código, bem como ter conhecimento de como diminuir a complexidade do código, otimizando-o.

O ALU3, por sua vez, cita que uma habilidade que ele adquiriu e que o ajudou no processo de aprendizagem foi a separação do tempo, ou seja, a organização do seu tempo de estudo, em detrimento de outras atividades do cotidiano. Neste caso, observamos esta habilidade não apenas para aprender programação, mas para qualquer área de conhecimento: separar tempo para o estudo, para o trabalho, saber organizar o tempo e a forma como estuda é uma habilidade importante para o sucesso da aprendizagem.

4.2.1.5. Conclusão. O parâmetro 'O que' nos ajuda a responder os objetivos 'caracterizar os fatores que podem influenciar no processo de ensino e aprendizagem' e 'explicitar os conhecimentos e habilidades necessários para aprender a programar'.

Na categoria 'background' encontramos evidências de que ter experiência anterior pode auxiliar no processo de aprendizagem de programação, embora alunos que nunca tenham tido contato com a programação também consigam aprender, ou seja, não chega a ser um fator sem o qual o aluno não irá ter sucesso na aprendizagem. O conhecimento em matemática traz posicionamentos divergentes entre alunos, professores e autores, embora possamos considerá-lo um fator importante para se aprender a programar. Na categoria 'conhecimentos', a triangulação dos dados converge para o que abordamos na fundamentação teórica, apesar de que não tenhamos obtido dados explícitos, acerca da importância do pensamento computacional nas vozes dos professores e alunos. Na categoria 'fatores que auxiliam' podemos afirmar que a autonomia e a proatividade são fatores fundamentais para o processo de aprendizagem de programação, entretanto, estes fatores não são exclusivos para a aprendizagem de programação, mas para as mais diversas áreas do conhecimento, podemos caracterizar estes fatores como fatores pessoais. Por fim, na categoria 'habilidade' ficou evidenciada que é preciso ter habilidade de resolução de problemas e pensamento lógico para aprender a programar.

4.2.2. Parâmetro 'Quem'

O parâmetro 'Quem' diz respeito ao papel que os professores, colegas e outros podem exercer para auxiliar no processo de ensino e aprendizagem de programação, e está estruturado em três categorias: 'papel docente', 'papel dos colegas' e 'papel de outros'.

4.2.2.1. Papel do Docente. Nesta categoria apresentamos três afirmações aos alunos respondentes dos questionários:

1- Me sentir motivado pelos professores me ajudou a aprender a programar.

2- O *feedback* fornecido pelo professor fora da sala de aula me ajudou a aprender a programar.

3- O apoio que o professor me deu em sala de aula foi importante para que eu aprendesse a programar.

A primeira afirmação diz respeito à motivação dada pelo professor para que o aluno sinta interesse em aprender a programar, como relatado por Guimarães & Boruchovitch (2004). Todos os alunos concordam com esta afirmação, de forma que 75% dos respondentes concordam totalmente e 25% concordam em parte. As respostas dos alunos vão de acordo com a temática abordada em nossa fundamentação teórica.

Para os alunos entrevistados, o papel do docente vai além de ministrar suas aulas. Ele perpassa pelo papel de 'pai' (ALU1.8), de orientador (ALU3; ALU4) e de amigo e incentivador (ALU2):

> nas disciplinas de programação, por exemplo, professores são muito, como eu posso dizer, amigos.... Você a qualquer momento pode conversar com eles, são aqueles *nerds* que nem você, que tá ali e sabe de tudo [pausa]. Eu acho que a proximidade que existe na [nome da Instituição omitida], entre professor e aluno, eu acho que incentiva muito. Não é aquele negócio deu a aula, vai embora, tchau, não. O professor continua lá, fica tirando dúvida, incentivando, 'ah, você está com algum problema deixe eu saber qual é?!' [breve pausa]. 'Professor, tô [Sic] meio perdido, o que é que eu vou fazer da minha vida?'. Aí ele vai lá e dá uma mão a você. (ALU2.10)

A voz do aluno ALU2 demonstra a empatia dos professores em querer ajudar mesmo em situações externas à sala de aula e reconhece com gratidão a abertura que os professores têm para conversar. Ele se vê no professor ao citar "aqueles *nerds* que nem você", demonstrando que são semelhantes, entretanto, reconhece a importância do papel do docente ao dizer que ele "tá ali e sabe de tudo", no sentido que o professor está disponível e com muito conhecimento a ser compartilhado. O papel do docente ultrapassa, assim, as barreiras da sala de aula, sendo participantes da formação humana dos seus alunos.

Para o aluno ALU5, mesmo sendo uma UC difícil, a motivação do professor pode ser determinante para o processo de aprendizagem. Ele reconhece que quando o professor está motivado, acaba estimulando os alunos a sentirem o mesmo ânimo e vontade de aprender. Na visão do aluno,

> Têm professores que eles têm uma didática melhor, que eles trazem exemplos, trazem coisas novas, que estimulam a turma, estimula a você, te dá um ânimo, te dá um gás, e a forma como ele explica, os exemplos que ele traz, a animação dele por si só [...], por mais que seja uma matéria superdifícil, mas a gente vai, por que vai com a empolgação dele, a motivação dele. (ALU5.6)

O ALU3 aponta o papel do professor em encorajá-lo diante das dificuldades enfrentadas na aprendizagem de programação:

> Ele [o professor] dizia: 'a ideia é muito boa, agora vamos transformar, vamos fazer o código, vamos programar'. [...] 'você tem que ir procurando devagarinho, faça do seu jeito, aí eu vou olhar, se não der certo a gente vai tentar achar uma solução para isso'. (ALU3.13)

E quando o ALU3 pensou em desistir, alegando que tinha chegado ao seu limite, o professor mais uma vez interveio: "de jeito nenhum, não senhor, a gente só consegue aprender fazendo, fazendo e fazendo. Se você desistir na primeira tentativa, nunca vai saber se tem capacidade para isso" (ALU3.14).

Para o aluno ALU3, o papel docente foi de suma importância para que ele não desistisse nas UC ou até mesmo do Curso. Além de apresentar uma metodologia que conseguiu auxiliar o aluno, o professor reconheceu os momentos em que o aluno apresentou desânimo e o motivou para que ele continuasse. Para o aluno ALU3, o professor é aquele que ajuda, que se preocupa com sua aprendizagem e que quer seu sucesso:

> O que eu sempre falo com os meninos é que os professores sempre ajudam muito, a gente às vezes está com a cabeça pesada, mas quando chegava para assistir aula, via a empolgação do professor, às vezes eu estava lá calado no canto, e vinha e: 'cadê, você fez?', e eu: 'não professor", 'ah, vamos lá, faça aqui você', aí ele me mostrava onde é que estava e eu digo: 'então vamos pra frente!' [risos]. (ALU3.15)

Neste caso, o aluno passa também a desempenhar um papel importante ao incentivar outros alunos a reconhecer a ajuda dos professores na aprendizagem. A figura do professor é, portanto, reconhecida pelos alunos como fundamental no processo de aprendizagem, não apenas pelo currículo do professor e metodologias por este adotadas, mas também pelas atitudes afetivas demonstradas durante sua prática docente.

Com as vozes dos professores, percebemos que o incentivo dado aos alunos pode ocorrer adaptando a forma de os avaliar. Para os professores PROF e PROF4,

> É injusto você cobrar só por prova, porque na prova tem aluno que trava e não consegue, e você coloca um exemplo um pouco mais complexo que ele tem uma forma de raciocínio um pouco mais lenta que os outros, e ele até vai conseguir fazer, mas na hora da prova, no tempo de prova ele não consegue, e aí você fica injusto cobrar, tem que ter trabalhos, tem que ter atividades. (PROF3.14)

Para o professor PROF4,

> Então, antes os trabalhos eram mais difíceis, em menos tempo, e na verdade isso não estava incentivando os alunos a persistir, a procurar, isso estava desmotivando, fazendo mais com que eles desistissem do que tivessem força de vontade pra continuar. (PROF4.17)

Em relação à segunda afirmação, sobre o *feedback* dado pelo professor fora de sala de aula, 84% dos estudantes respondentes do questionário concordam que, de fato, esse aspecto foi importante para o seu processo de aprendizagem. Esta visão dos alunos corroboram com o que abordamos na fundamentação teórica (Bosse & Gerosa, 2017; T. Castro et al., 2011; Chuchulashvili et al., 2016; A. Gomes & Mendes, 2015; Nawahdah et al., 2015; Ortiz-Ortiz et al., 2018; Paredes et al., 2019; R. M. S. Pereira, 2017; Shadiev et al., 2013; Skalka & Drlik, 2018; Souleiman, 2017).

Enquanto prática docente, os alunos falam que o papel do professor é fundamental ao tirar dúvidas, indicar aos alunos como e onde tirarem suas dúvidas e que ferramentas utilizarem para estudar programação, mesmo fora de sala de aula (ALU2, ALU3, ALU5). Com isto, o professor ajuda seu aluno a desenvolver autonomia e proatividade, habilidades importantes para o processo de aprendizagem de programação.

Além dessas estratégias, a forma como o professor recepciona as dúvidas do aluno e como as explica, é um diferencial para o aluno. O aluno sente-se mais seguro quando faz suas perguntas ao professor, pois acredita que ele dará os melhores encaminhamentos. Para o aluno ALU3:

> É diferente de você perguntar ao professor, né? Professor: assim, assim... às vezes é uma besteirinha, né? Uma coisa assim que a gente não percebe, aí [o professor responde] 'não, é só isso assim...' e me tirava uma trave grande da vista. Foi muito confortante, muito prazeroso aprender do jeito que a metodologia que eles ensinam. (ALU3.12)

O papel do professor é 'destravar' o aluno e permitir que ele assimile o assunto de forma mais fácil. O conforto que o aluno sente com a metodologia adotada pelo professor demonstra a dedicação do docente durante o ensino de programação. Observar essa visão do aluno em relação ao papel docente reflete a importância da presença do professor na sua trajetória acadêmica, demonstrando satisfação em receber atenção e todo apoio do professor.

A terceira afirmação aborda o papel do professor como apoiador do aluno dentro de sala de aula, como apresentado por Bosse & Gerosa (2017), Mendes et al. (2012), Nawahdah et al. (2015), R. M. S. Pereira (2017), A. Santos et al. (2010, 2013) e Souleiman (2017). O apoio do professor pode ser motivando ou tirando dúvidas dos alunos dentro da sala de aula. De acordo com os dados obtidos, 90% dos alunos concordam que o apoio do professor é importante para o processo de aprendizagem de programação. Para o aluno A10,

> O que mais ajudou a programar foi a maneira de ensinar do professor que foi super atencioso e acompanhava o passo a passo ajudando a superar as dificuldades e motivando sempre a fazer novos exercícios para melhor fixar o aprendizado. (A10.1)

O aluno A02 afirma que "com o auxílio de um dos professores, que foi muito importante, consegui superar as dificuldades das matérias [UC]" (A02.1).

4.2.2.2. Papel dos Colegas. Na categoria 'papel dos colegas' apresentamos a seguinte afirmação:

1- Ser motivado pelos colegas me ajudou a aprender a programar.

A intenção desta afirmação é perceber como o aluno vê a influência dos colegas, no sentido de motivá-lo e instigá-lo a aprender a programar, como apresentado por Guimarães & Boruchovitch (2004). Os dados obtidos mostram que a percepção dos alunos corrobora com o que falam os autores, de forma que menos de 10% dos alunos participantes discordam da afirmação.

Para o ALU2, estar em contato com os colegas é o que mais o ajuda na aprendizagem, tirando-lhe dúvida quando necessário. Para ele, a amizade influencia muito durante o processo de aprendizagem:

> Amizade influencia muito né? Então se seus amigos não estão querendo, não estão querendo estudar, isso influencia muito, talvez... ainda bem que, pelo menos o meu grupo de amigos que assim eu converso mais, eles são também muito estudiosos, gostam muito de estudar... Também gostam de desenvolver, né? O foco deles é esse, então a gente sempre conversa sobre isso, e eu acho que isso colabora muito. (ALU2.13)

Assim, de acordo com ALU2, se os colegas gostam de estudar e estão focados, influencia também para que ele mantenha o foco. O papel dos colegas, neste caso, é de incentivo e colaboração. O papel de incentivo também é relatado pelo ALU3, no trecho:

> mas aí os meus colegas começaram 'não homem, o que é isso? Ninguém...' é, eu lembro muito a frase de [nome do colega omitido] que sempre diz assim: 'ninguém é velho o bastante que nunca possa aprender, você vai desistir assim?', aí eu digo: 'é, vamos tentar né?' Aí pronto e estou até aqui. (ALU3.17)

Este trecho é parte da fala em que o aluno alega se sentir 'velho' em relação aos seus colegas e, por isso, sente dificuldade para aprender a programar. O incentivo do colega com a fala "ninguém é velho o bastante que não possa aprender" tenta quebrar o pensamento do aluno ALU3, que acredita que os mais novos têm mais facilidade para aprender algo, e fazendo perceber ser possível aprender a programar. Além disso, o colega citado quebra preconceitos vividos pelo ALU3, ajudando-o a sentir-se

capaz de aprender. Para o ALU3, os colegas também estão sempre dispostos a ajudar, principalmente por terem experiência em programação.

Para ALU5, a união da turma é fundamental para a aprendizagem e aponta a disponibilidade dos colegas em ajudar, montando grupos de estudo para trocarem ideias e assim incentivar uns aos outros a aprender (ALU5.7).

Independente do papel desempenhado, os alunos entrevistados mostram a importância dos colegas em seu processo de aprendizagem. Estar em convívio com pessoas com os mesmos gostos, interesses, angústias e dificuldades é importante para o ser humano. A troca de experiências e a entreajuda em um processo de aprendizagem, independente da área de conhecimento, dá mais força e ânimo para os estudantes. Sentir-se parte de um grupo e estar acolhido, independentemente das diferenças, são incentivos para que estes enfrentem os preconceitos e estereótipos existentes.

Para os professores PROF2 e PROF4, o papel dos colegas pode ser de motivador, mas também de alguém com quem possa estudar, tirar dúvidas e apoiar. Para eles,

> às vezes você não vai conseguir resolver sozinho 'ah, tenho um colega que tá trabalhando esse mesmo problema' e eu estimulo muito, quando eu tô [Sic] dando essas disciplinas eu digo: 'olhe, enganchou, teve um problema, procure o colega lá, não é pra você copiar o código dele, mas é pra discutir, e tentar resolver, buscar ajuda'. (PROF2.21)

> Então por exemplo, que fatores podem auxiliar: a questão dos amigos, né? Dos colegas de turma, isso eu acho que é uma coisa muito importante por que nem sempre a pessoa é boa em todos os assuntos e quando você tem um colega que pode lhe auxiliar, e você pode auxiliar, ou quando se faz um trabalho em grupo e que o grupo consegue realmente é, produzir, você vê que os alunos se desenvolvem, principalmente os alunos que têm um bom grupo, que têm um bom grupo de estudo, então eles conseguem se desenvolver. (PROF4.24)

A fala do PROF4 é parecida com a do aluno ALU2, quando fala da importância de ter um grupo de alunos que estudam juntos e se incentivam. O PROF4 cita ainda o papel do monitor da UC de programação, com quem os alunos têm mais intimidade e mais abertura para tirar dúvidas e que com isso consegue auxiliar os alunos no processo de aprendizagem. Além disso, o mesmo professor aponta que "se eles tiverem também alguém a quem consultar, não só os próprios amigos da sala [...], mas também colegas de outras turmas, então também é importante, alunos de outras turmas que possam atuar ali como conselheiros, como um suporte" (PROF4.25).

4.2.2.3. Papel dos Outros. Para além dos docentes e colegas de turma, os alunos entrevistados citaram papel de outros envolvidos no seu processo de aprendizagem de programação.

Para o aluno ALU2, é importante fazer parte de grupos e fóruns de dúvidas de programação, com pessoas de fora de seu convívio acadêmico, mesmo que para tirar dúvidas pontuais, pois estes têm outras vivências e experiências sobre os assuntos abordados na Universidade (ALU2.18). Além disso, o aluno cita o papel de um projeto do Curso (ALU2.15), desenvolvido pelo Programa de Educação Tutorial (PET) que tem como objetivo apresentar o Curso a alunos que ainda estão no Ensino Médio, de forma que os alunos conheçam sua estrutura, seus objetivos e a formação oferecida, evitando que alunos cheguem no curso sem conhecê-lo ou sem saber o que será visto ao longo do percurso acadêmico. Por fim, o ALU2 cita ainda o papel de um colega do Ensino Médio, que lhe falou sobre o Curso: "Eu vim conhecer [o Curso] por causa de um amigo meu, que inclusive estuda comigo hoje, no Ensino Médio, aí dali eu comecei a focar: vou fazer computação, mas eu nem tinha noção do Curso em si" (ALU2.16).

O aluno ALU3 destaca a importância do papel da família para o seu desenvolvimento no Curso e UC de programação: "mas aí tanto em casa, a família, quanto os meninos lá diziam: 'calma, paciência, todo mundo tem um jeito de aprender, cada um tem a sua forma de entender a situação, se todo mundo fosse igual'" (ALU3.19). O mesmo aluno fala do papel de seu filho, que cursa Informática em um curso técnico, em seu desempenho em programação. A disponibilidade do filho e o retorno que este pode oferecer para o ALU3 é motivo de orgulho para ambos. Em sua fala, o ALU6 também aponta a importância do papel de um familiar que, por ser da área de Informática, o influencia e motiva a estar no Curso.

Para o aluno ALU5, ser aluno de um curso técnico na área de Informática foi fundamental para a escolha do curso de graduação: "eu não sabia o que fazer antes do [nome do curso omitido], [...] quando eu entrei no [nome do curso omitido] e eu comecei a entrar nesse mundo da Informática e ver as possibilidades, aí eu comecei a me apaixonar" (ALU5.9). Por ter uma vivência com a programação e a área de Informática no Ensino Médio, o aluno pôde ter maior clareza na sua escolha para o Ensino Superior. O papel do curso anterior foi importante para sua formação, uma vez que, por proporcionar UC práticas, deu a oportunidade de ter experiências de programação antes de ingressar na Universidade, diferente dos demais entrevistados.

O aluno ALU5 cita ainda o papel do Curso como um local de boa convivência, considerando 'uma família' e fala do papel da Universidade em auxiliá-lo em momentos pessoais de dificuldade:

> e aí quando foi no segundo semestre eu procurei o auxílio da [nome da instituição omitida], com a psicopedagoga e foi muito importante, ela me ajudou bastante, me ajudou nos cronogramas e com o acompanhamento dela, montando rotinas e tudo, eu consegui dar a volta por cima, consegui dar uma aliviada, e consegui respirar, e aí minhas notas começaram a melhorar e eu fui me animando com o Curso, daí eu falei: 'agora vai, foco, e vamos para frente que agora vai dar certo'. E depois desse apoio eu não pensei mais em desistir. (ALU5.11)

A Universidade e o setor de psicopedagogia conseguiram auxiliá-lo a desenvolver competências como organização e planejamento. O PPC indica como os alunos podem contactar com o setor de atendimento psicológico e social da Universidade (PPC, 2019, p. 106).

Com a realização da análise, percebemos que professores, colegas de turma, familiares e instituições desempenham papeis que são fundamentais e que se complementam para beneficiar o processo de ensino e aprendizagem. Embora o nosso foco seja a aprendizagem de programação, podemos inferir que os papeis citados pelos entrevistados podem dar suporte aos alunos independente da área de atuação. Reconhecer a ajuda de terceiros no processo de aprendizagem é de grande valia, pois ajuda ao aluno a recorrer a estes nos momentos oportunos de sua trajetória acadêmica.

4.2.2.4. Conclusão. O parâmetro 'Quem' nos ajuda a 'caracterizar os fatores que podem influenciar no processo de ensino e aprendizagem de programação', tendo em vista o papel desempenhado por professores, alunos, familiares e instituições.

Na categoria 'papel do docente' vimos que se torna imprescindível para o processo de ensino e aprendizagem a motivação, o feedback fora de aula e o apoio dado pelo professor dentro de sala de aula. Com a triangulação dos dados, a fala dos alunos se complementam e evidenciam a importância do professor em seu percurso acadêmico. Embora existam as mais diversas fontes de informação e conhecimento disponíveis nos mais diversos formatos, a figura do docente é, de fato, considerada pedra fundamental para o ensino. Assim, a nossa análise nos permite afirmar que o que nos falam as fontes de dados, convergem para o que apresentamos na fundamentação teórica. O 'papel dos colegas' também é importante na vida acadêmica do aluno de programação, no tocante à entreajuda, à motivação, especialmente nos momentos de desânimo e dificuldades enfrentadas ao longo do processo de aprendizagem. No papel 'dos outros', encontramos aspectos emergentes, tais como apoios sociais fornecidos pela Universidade e projetos de extensão que auxiliam os alunos para conhecer melhor a área de Informática, além de fatores que ultrapassam os muros da Universidade: como o da família, que tem um papel fundamental no processo, dando apoio e incentivo aos alunos no percurso acadêmico. Os resultados da triangulação dos dados nos fazem perceber que ter a

possibilidade de receber ajuda de outros envolvidos no processo torna-se um fator importante para o processo de ensino e aprendizagem, corroborando com o que abordamos na fundamentação teórica.

4.2.3. Parâmetro 'Onde'

O parâmetro 'Onde' busca compreender onde os alunos estudam, buscam informações e como eles percebem a importância de ter um ambiente de aprendizagem adequado e recursos de qualidade para o seu processo de aprendizagem de programação. Este parâmetro está estruturado em duas categorias: 'onde estuda' e 'onde busca informação'.

4.2.3.1. Onde Estuda. Em relação à categoria 'onde estuda', todos os alunos participantes da pesquisa, estudam no Curso e Todos os professores participantes nele. O Curso é de regime presencial, entretanto quando realizamos a pesquisa as aulas estavam ocorrendo na modalidade remota devido à pandemia causada pelo novo coronavírus (SARS-CoV-2). Para contemplar esta categoria fizemos as seguintes afirmações no questionário:

1- Ter uma infraestrutura adequada no curso (laboratórios e sala de aula) auxilia no processo de aprendizagem de programação.
2- O ambiente de estudo influencia no processo de aprendizagem de programação.

A primeira afirmação baseia-se em Berssanette (2016) e busca verificar a importância de se ter laboratórios, salas de aula, estrutura de apoio adequados para se ter um melhor aproveitamento no curso. O resultado obtido mostra que todos os alunos participantes do questionário concordaram totalmente com esta afirmação, evidenciando que eles reconhecem que um curso que oferece um ambiente adequado de estudo pode auxiliá-los na aprendizagem.

Os professores PROF3 e PROF2 corroboram com Berssanette (2016) ao falar que "o ambiente universitário também é importante, não só a estrutura, mas a turma, os colegas, se eles se envolvem, se eles não se envolvem, o quanto um é capaz de ajudar o outro" (PROF3.13) e "fator que nas universidades públicas [...] atrapalha, que é a dificuldade de infraestrutura [...] Você tá em laboratórios que não são os melhores, não são os melhores computadores, tem limitações estruturais, e isso atrapalha um pouco" (PROF2.45-46). O professor PROF3 complementa ainda que:

> A Universidade além de ter a disciplina, o conteúdo, a disciplina conteudista que passa o conteúdo, tem também que dar a estrutura para que o aluno se integre e ele consiga praticar dentro da Universidade. A gente consegue, eu creio que a gente tem estrutura pra isso, a [nome da Instituição omitida] [...] tem estrutura e dá a possibilidade ao aluno que quiser e puder ficar lá e executar, e treinar, e fazer, e estudar, não só algoritmos, mas qualquer disciplina. (PROF3.6)

Na segunda afirmação, sobre o ambiente de estudo, tencionamos verificar se o aluno concorda que, tendo um local adequado para estudar, seja em casa, trabalho, Universidade, poderá ter melhor aproveitamento na aprendizagem de programação, como apontado por Chuchulashvili et al. (2016) e Eltegani & Butgereit (2015). Cerca de 80% dos alunos concordam com a afirmação, corroborando com o que abordam os autores que versam sobre este assunto. O professor PROF1 sinaliza que "estudar e estar com o celular do lado, não ajuda de jeito nenhum. Você estudar assistindo um seriado não ajuda" (PROF1.22), ou seja, é necessário ter um ambiente adequado para que o aluno possa estar concentrado para estudar. O professor PROF4 enfatiza que,

> Porque digamos, [...] por exemplo [o aluno] não fica no laboratório, ele não tem um recurso bom, então ele vai pra casa, às vezes não tem um ambiente de estudo, né, um ambiente de estudo pra que ele possa estudar, que ele possa praticar, às vezes ele não tem um computador bom, às vezes mora com a família, não tem um quarto, não tem um espaçozinho que ele possa usar lá, então isso aí é, pode atrapalhar bastante. (PROF4.46)

Por outro lado, para este mesmo professor "Se ele [o aluno] tiver uma boa máquina, uma conexão à internet boa, se tiver bons livros, né? Você tiver acesso a material, então uma pessoa dessas tem tudo na mão" (PROF4.12).

Os professores concordam que, dependendo do ambiente de estudo, o aluno pode ter diferentes desempenhos em seu processo de aprendizagem. Se por um lado um ambiente adequado favorece a aprendizagem do aluno, por outro lado um ambiente sem a estrutura adequada irá atrapalhar o processo de aprendizagem.

4.2.3.2. Onde Busca Informação. Nesta categoria pretendemos mostrar as vozes dos alunos sobre onde buscam recursos para estudar. A escolha, por parte dos alunos, de materiais didáticos, de um ambiente de aprendizagem e de ferramentas apropriadas depende de vários fatores e pode ser dificultada devido ao dinamismo e complexidade da área de Informática e pela abundância de opções e falta de critérios e padrões de comparação para fazer a seleção formal e rigorosa dentre as possibilidades (Brusilovsky et al., 2014).

Assim, é necessário maturidade e prática do aluno para que esse aprenda a buscar as informações, além de orientação, por parte dos docentes, de como e onde os alunos podem encontrar conteúdos e informações que possam auxiliá-los no processo de aprendizagem. Como os alunos estão cada vez mais conectados à internet e usando recursos digitais, é notória a diminuição do interesse destes por buscar informações em livros didáticos físicos, recorrendo a ferramentas e recursos

disponíveis na internet, sejam artigos, e-books, repositórios ou fóruns de apoio. O PROF4 afirma que costuma indicar livros com exemplos para que os alunos possam praticar. Para ele,

> A pessoa não sabe por onde começar a estudar, como é que começa a estudar, que material é bom ou não é bom, e muitos materiais são muito fracos, muito limitados, ensinam de uma forma errada, então, você tem uma quantidade absurda de materiais, mas você tem poucos que realmente valem a pena, então eles ficam desnorteados, sem saber como estudar, ou o que procurar. [...] Nas disciplinas eu tenho tentado, por exemplo, dar indicações de canais no Youtube, que tem bons conteúdos, ou páginas com livros ou tutoriais que realmente o material seja de boa qualidade. (PROF4.32)

De acordo com o professor, a grande quantidade de material disponível pode dificultar o aluno a encontrar materiais confiáveis e de qualidade para o seu estudo, e afirma que é necessário dar um direcionamento aos alunos para que estes possam encontrar os melhores recursos para estudar.

O aluno ALU1 confessa que usou muito pouco o acervo disponibilizado na biblioteca da Universidade e, mesmo já estando no sétimo período do Curso, só se recorda de ter pegado dois ou três livros na biblioteca para estudar. O aluno ainda fala que gosta "muito mais de um [material do tipo] PDF, um buscar na Internet" (ALU1.10). Além disso, complementa que para além do material disponibilizado pelo professor da UC, é importante buscar exemplos e outros materiais na Internet, para complementar o que está sendo no Curso.

O aluno ALU2 cita três principais fontes de informações em que usa como apoio e orientação para a sua aprendizagem de programação: as documentações oficiais das linguagens de programação, grupos no Telegram e comunidades de aprendizagem de linguagens de programação formadas por alunos de outras Universidades e sites com fóruns de ajuda como, por exemplo, o Stackoverflow (ALU2.18).

Para o ALU2, estar envolvido nestas comunidades cria perspectivas na sua aprendizagem e abre um leque de possibilidades e visões em relação à programação. O fato de o aluno sair da sua zona de conforto, mostra sua capacidade de iniciativa, proatividade e autorregulação da aprendizagem.

O ALU3 cita a internet como uma fonte de ajuda para as suas dúvidas e o ALU4 cita repositórios Git, como GitHub e o GitLab, como bons locais de encontrar material e código para ajudá-lo na aprendizagem (ALU4.17). Assim como o ALU2, o ALU4 também cita o Stackoverflow como fonte de material de apoio. O uso destas ferramentas possibilita aos alunos a visualização de soluções de problemas de programação, desenvolvidas por outras e disponibilizadas na Internet para consultas, estudos e uso dos códigos.

4.2.3.3. Conclusão. O parâmetro 'Onde' nos ajuda a 'Inventariar metodologias e ferramentas usadas para o ensino e aprendizagem de programação' e 'Caracterizar os fatores que podem influenciar para o aluno aprender a programar', uma vez que busca elucidar onde o aluno estuda e busca seus recursos para a aprendizagem, mas também consegue nos dar chances de 'identificar as dificuldades enfrentadas por professores e alunos durante o processo de ensino e aprendizagem', quando, por exemplo, o aluno não tem acesso a ambiente adequado para estudar ou a recursos de apoio que sejam confiáveis.

A categoria 'onde estuda' torna claro que a infraestrutura acadêmica, no que concerne salas de aulas, laboratórios, são fatores fundamentais para o processo de ensino e aprendizagem. Se o curso oferece uma estrutura adequada para professores e alunos, a possibilidade de maior aproveitamento por parte destes se torna mais evidente. Na mesma linha de pensamento, o ambiente de estudo dos alunos, seja em casa, no laboratório ou no trabalho, tem papel importante no seu processo de aprendizagem, tendo em vista que esta é beneficiada quando o aluno consegue estar concentrado, em um ambiente sem ruídos, com boa iluminação, com recursos adequados. A triangulação dos dados convergem com o que abordam os autores presentes na fundamentação teórica, deixando evidente que estes fatores, de fato, contribuem para o processo de ensino e aprendizagem. A categoria 'onde busca informação' permite conhecer as principais fontes em que os alunos buscam informações. Com a análise desenvolvida, descobrimos que estas são variadas e que dependem do assunto abordado e do grau de maturidade dos alunos. É nítido, ainda que, se por um lado o grande número de materiais disponíveis possibilita que alunos e professores possam escolher as melhores ferramentas e recursos, por outro essa abundância dificulta a seleção de materiais de qualidade, trazendo dificuldades para o processo de ensino e aprendizagem. As análises nos mostram uma convergência existente entre as opiniões das fontes de dados da nossa pesquisa, que corroboram também com a fundamentação teórica apresentada.

4.2.4. Parâmetro 'Quando'

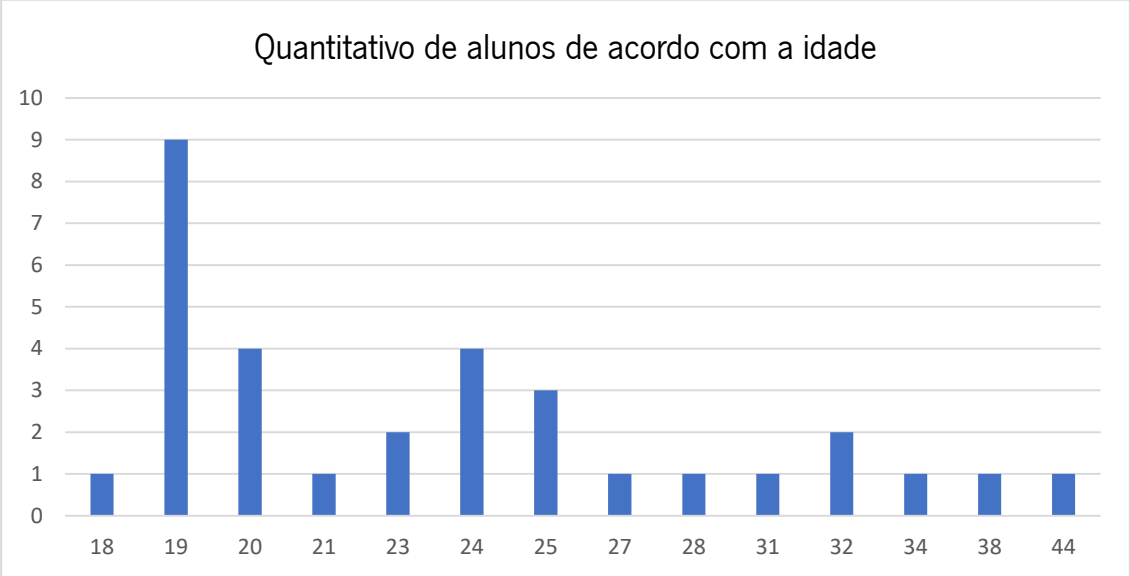
O parâmetro 'Quando' nos dá a possibilidade de conhecer os respondentes e sua relação com o Curso. As UC de programação do Curso estão inseridas a partir do segundo período com a UC 'Construção de Algoritmos' (PPC, 2019). Nesta UC os alunos têm o primeiro contato com uma linguagem de programação e com o paradigma de programação estruturada. Ao longo do Curso, em diferentes períodos, os alunos se deparam com diferentes UC de programação, as quais possuem diferentes estruturas e objetivos. Neste parâmetro, apresentamos a trajetória ao longo do Curso, dos

professores e alunos entrevistados em três categorias: 'entrada no curso', 'período regular' e 'UC de programação'.

4.2.4.1. Entrada no Curso. A entrada no Curso é anual, sempre no primeiro período letivo de cada ano. Atualmente, o ingresso de novos alunos pelo Exame Nacional do Ensino Médio (ENEM) (PPC, 2019). Para esta categoria, perguntamos aos alunos a idade que tinha na altura da pesquisa e o ano em que ingressaram no Curso. O Gráfico 1 apresenta a caracterização dos respondentes ao questionário de acordo com a idade.

Gráfico 1

Caracterização dos participantes de acordo com a idade



Nota: Autoria própria.

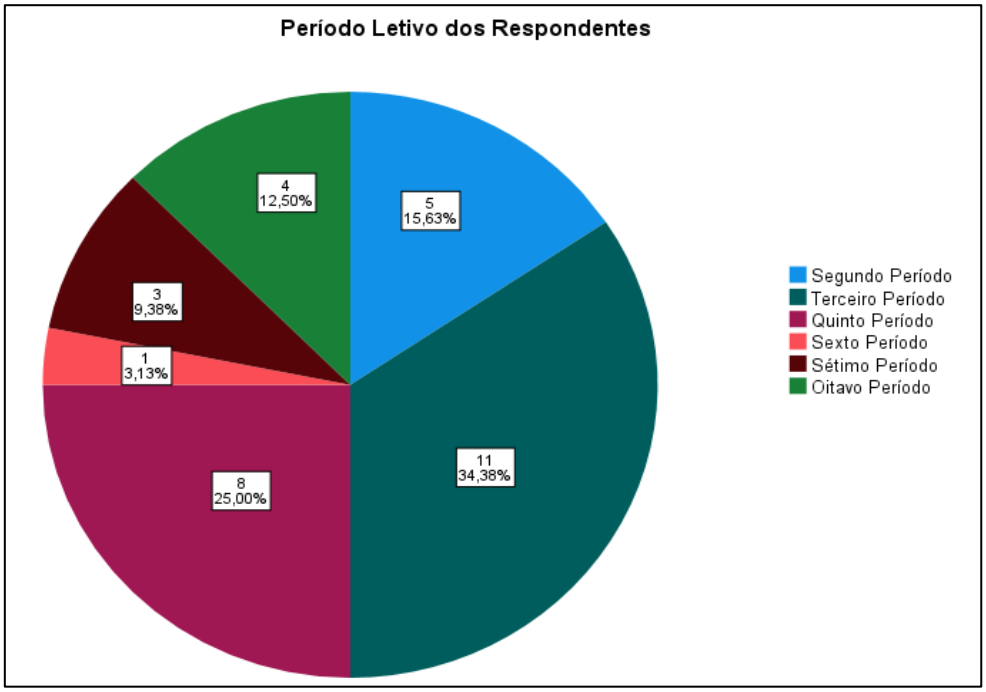
Como mostra o Gráfico 1, a idades dos respondentes está compreendida entre 18 e 44 anos. A média de idade dos respondentes é de 24 anos. A moda da idade é 19 anos (representando 28,1% dos participantes da pesquisa) e a maioria dos respondentes tem idade entre os 18 e 25 anos (representando 75% dos participantes). Em relação aos participantes da entrevista, dois alunos têm 19 anos, um tem 21 anos, um com 24, um aluno com 29 e outro aluno com 44 anos. A média das idades dos participantes das entrevistas é de 26 anos e a moda 19 anos.

Dos alunos participantes da entrevista, um ingressou no Curso em 2015 (ALU3), dois em 2016 (ALU1 e ALU5), e três em 2018 (ALU2, ALU4 e ALU6). Sobre os professores, o PROF1 ensina programação desde 2006, o PROF2 desde 2002, o PROF3 desde 2001, o PROF4 desde 2004 e o PROF5 começou a ensinar programação em 2016.

4.2.4.2. Período Regular. Na categoria 'período regular' buscamos conhecer o período em que os alunos estavas matriculados quando a pesquisa foi realizada. O Gráfico 2 apresenta os dados obtidos.

Gráfico 2

Caracterização dos participantes de acordo com o período letivo em que estão matriculados



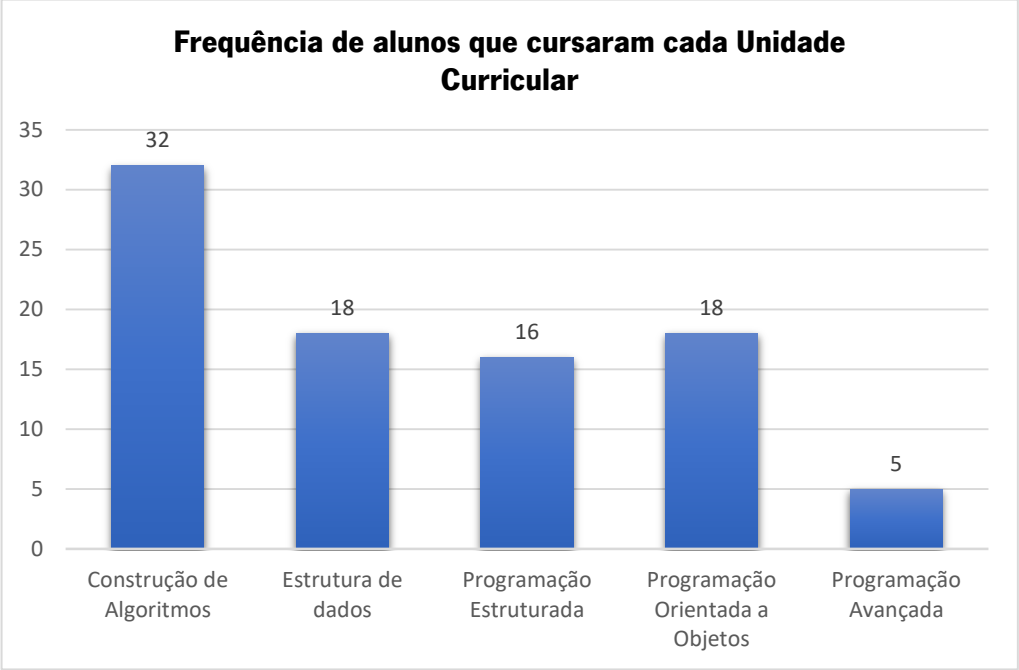
Nota: Autoria própria.

De acordo com o Gráfico 2, metade dos alunos respondentes do questionário estavam no segundo e terceiro período do Curso. Os demais alunos estavam matriculados no quinto, sexto, sétimo e oitavo períodos. Nenhum aluno do quarto período participou da pesquisa. Em relação aos alunos que participaram das entrevistas, três estavam regularmente matriculados no quinto período (ALU2, ALU4 e ALU6) e três no sétimo período (ALU1, ALU3 e ALU5).

4.2.4.3. UC de Programação. Em relação à categoria 'UC de programação', a intenção é saber dos alunos quais UC estavam cursando ou já haviam cursado e dos professores quais UC de programação geralmente ensinam. Os dados em relação aos respondentes dos questionários, estão apresentados no Gráfico 3.

Gráfico 3

Caracterização dos participantes de acordo com as UC cursadas ou em curso



Nota: Autoria própria.

O Gráfico 3 mostra que todos os alunos já haviam cursado ou estavam cursando Construção de Algoritmos, 56,3% Estrutura de Dados e Programação Orientada a Objetos, 50% Programação Estruturada e 15,6% Programação Avançada. De acordo com os dados obtidos com as entrevistas, a Tabela 11 apresenta as UC ministradas por cada professor e cursada por cada aluno.

Tabela 11 - *ID do entrevistado e UC de programação lecionadas ou cursadas*

Id entrevistado	**UC de Programação**
PROF1	Programação Estruturada, Programação Orientada a Objetos, Programação Avançada.
PROF2 e PROF3	Construção de Algoritmos, Estrutura de Dados, Programação Estruturada
PROF4	Construção de Algoritmos, Programação Orientada a Objetos, Programação Avançada
PROF5	Construção de Algoritmos, Estrutura de Dados, Programação Avançada.
ALU1 e ALU5	Construção de Algoritmos, Estrutura de Dados, Programação Estruturada,

	Programação Orientada a Objetos e Programação Avançada.
ALU2, ALU3, ALU4 e ALU6	Construção de Algoritmos, Estrutura de Dados, Programação Estruturada e Programação Orientada a Objetos.

Nota: Autoria própria.

Todos os alunos entrevistados já cursaram Construção de Algoritmos, Estrutura de Dados, Programação Estruturada e Programação Orientada a Objetos. Apenas os ALU1 e ALU5 já haviam cursado a UC Programação Avançada quando participaram da entrevista, tendo, portanto, cursado todas as UC de programação do Curso. De acordo com a Tabela 11, podemos observar o caso do ALU3 que está há mais de cinco anos no Curso e que embora esteja no sétimo período, ainda não cursou a UC Programação Avançada que é ofertada no quinto período. Isso se deve a acontecimentos em sua trajetória acadêmica, tais como desistências e reprovações em UC que são pré-requisitos.

Em relação aos professores, quatro ministram Construção de Algoritmos (PROF1, PROF2, PROF3 e PROF4), três ensinam Estrutura de Dados (PROF2, PROF3 e PROF5), três Programação Estruturada (PROF1, PROF2 e PROF3), dois Programação Orientada a Objetos (PROF1 e PROF4) e três Programação Avançada (PROF1, PROF4 e PROF5). O fato de mais de um professor ensinar a mesma UC é interessante para que possam dividir experiências, revezar as UC ministradas em diferentes anos letivos, dar possibilidade para o aluno tirar dúvidas com diferentes professores e ainda possibilitar que uma turma seja dividida entre dois professores, diminuindo o tamanho da turma, possibilitando uma melhor experiência no processo de ensino e aprendizagem.

4.2.4.4. Conclusão. O parâmetro 'Quando' nos ajuda a 'caracterizar os fatores que podem influenciar para o aluno aprender a programar' e 'identificar as dificuldades enfrentadas por professores e alunos durante o processo de ensino e aprendizagem, e o que estas dificuldades acarretam', uma vez que nos apresenta quando os alunos e professores ingressaram no curso, as UC cursadas e ministradas e as percepções acerca do percurso.

A categoria 'entrada no curso' nos permite caracterizar o perfil do aluno, no tocante à sua idade, o que nos faz compreender que se trata de um público jovem, muitas vezes recém saído do Ensino Médio e provavelmente com pouca ou nenhuma experiência no mercado de trabalho. Os anos de entrada nos mostram que alguns alunos passaram do tempo regular do curso, que são quatro anos, o que nos faz refletir que podem ter passado por dificuldades que puderam culminar no atraso, em desistências ou reprovações. Estes fatores são evidenciados posteriormente nas falas dos alunos entrevistados. Na categoria 'período regular' temos uma fotografia do percurso acadêmico de cada aluno participante da pesquisa: enquanto uns estão iniciando o curso, outros estão nos momentos

finais, e outros no meio da caminhada. Esses fatores nos permitem ter diferentes visões e perspectivas destes em relação ao processo de ensino e aprendizagem de programação. Na categoria 'UC de programação' conhecemos as UC cursadas pelos alunos e ministradas pelos professores. Percebemos que todas as UC de programação do curso estão contempladas em nosso estudo, uma vez que todas foram sinalizadas como cursadas e como ministradas. Assim, podemos ter diferentes visões de professores e alunos para uma mesma Unidade Curricular, tendo em vista as diferentes experiências vividas por estes.

4.2.5. Parâmetro 'Porque'

O parâmetro 'Porque' diz respeito às dificuldades que podem existir no processo de ensino e aprendizagem de programação e as possíveis consequências causadas pelas dificuldades. Para tanto, está estruturado em duas categorias: 'dificuldades enfrentadas' e 'consequências das dificuldades'.

Com a análise realizada, os dados obtidos revelam que os professores conhecem profundamente as principais dificuldades dos alunos e os alunos possuem uma percepção das dificuldades e do que elas trazem de consequências em sua caminhada.

4.2.5.1. Dificuldades Enfrentadas. Na categoria 'dificuldades enfrentadas', fizemos as seguintes afirmações no questionário:

1- A mudança do Ensino Médio para o Ensino Superior dificulta a aprendizagem de programação.
2- Ter pouco conhecimento em matemática dificulta a aprendizagem de programação.
3- Não ter conhecimento em Língua Inglesa dificulta o processo de aprendizagem de programação.
4- O aluno pode ter dificuldade em aprender a programar por não possuir experiência em programação.
5- A natureza dos problemas computacionais e como resolvê-los algoritmicamente dificultam a aprendizagem de programação.
6- Não compreender um conceito-chave dificulta a aprendizagem de programação.
7- Uma turma com muitos alunos dificulta o processo de aprendizagem de programação.
8- Uma turma com alunos de diferentes perfis e níveis de conhecimentos em relação à programação dificulta o processo de aprendizagem de programação.
9- A autoconfiança baixa do aluno pode prejudicar o processo de aprendizagem de programação.
10- Um aluno desmotivado tem dificuldade no processo de aprendizagem de programação.
11- Não conseguir interpretar o problema a ser resolvido dificulta a aprendizagem de programação.

12- Estudar pouco ou de maneira passiva pode dificultar o processo de aprendizagem de programação.
13- A dissociação entre os exercícios vistos em sala de aula e o mundo real dificulta a aprendizagem de programação.
14- A linguagem de programação usada pelo professor pode dificultar a aprendizagem de programação.

Na primeira afirmação, buscamos identificar a percepção dos alunos em relação a transição entre os níveis de ensino e como esta mudança pode afetar o seu desempenho acadêmico, como abordado por Ambrósio et al. (2011). Os dados levantados apontam que 69% dos participantes da pesquisa concordam que a adaptação ao Ensino Superior pode dificultar o processo de aprendizagem. Para além das diferenças entre os níveis de ensino, ainda há o problema social, especialmente dos alunos oriundos de escolas públicas, como assinala o aluno A12: "sou oriundo de escola pública, onde o ensino possui uma deficiência crônica e sair de um Ensino Médio com várias lacunas e ter que se adequar a uma nova realidade, demanda um tempo" (A12.1).

Um dos problemas enfrentados pelos alunos, quando entram no Ensino Superior, é o processo de adaptação às novas rotinas, horários e conteúdo. Para alguns dos professores entrevistados, o problema de adaptação se dá pelo fato de que os alunos não conhecem o curso nem o que um profissional da área Informática faz (PROF1, PROF4). Para o PROF3, a forma como os alunos são preparados na sua trajetória acadêmica antes de entrar no Ensino Superior pode dificultar o processo de aprendizagem de programação. O aluno ALU5, ao comparar a Universidade com sua instituição de ensino no Ensino Médio, afirma que são "dois mundos diferentes, sistemas de ensino diferentes" (ALU5.20). Percebemos que os relatos dos professores e alunos indicam que a transição do aluno para um novo nível de ensino pode trazer dificuldades que demandam tempo e esforço para a adaptação. Podemos ainda sugerir que este fator não é isolado para o ensino e aprendizagem de programação, mas pode afetar outras áreas do conhecimento também.

Para a segunda afirmação, buscamos perceber se o aluno compreende que a falta de conhecimento em matemática dificulta a aprendizagem de programação, como abordado na nossa pesquisa (Franzen et al., 2018; Gomes & Mendes, 2007; Medeiros et al., 2020; Bosse & Gerosa, 2017; T. H. Castro et al., 2008; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015, 2010; Rezende & Bispo, 2018; Shadiev et al., 2013). Como resultado, cerca de 60% dos alunos que responderam ao questionário concordam que não ter uma boa base matemática pode dificultar a

aprendizagem de programação, corroborando com o apresentado pelos autores citados em nossa fundamentação teórica.

Em relação às entrevistas, para os professores PROF1 e PROF3, a maior dificuldade dos alunos é na base, especialmente base matemática e de raciocínio lógico, corroborando com o que abordamos no capítulo 2 desta tese (Franzen et al., 2018; Gomes & Mendes, 2007; Medeiros et al., 2020; Bosse & Gerosa, 2017; T. H. Castro et al., 2008; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015, 2010; Rezende & Bispo, 2018; Shadiev et al., 2013).

Para os alunos ALU4 e ALU6, problemas que envolvem conhecimentos em matemática causam dificuldades, principalmente quando se trata de resolução de problemas específicos da matemática, embora sejam assuntos vistos antes de entrar na Universidade. Nos relatos desses alunos, percebemos que ambos viram os assuntos matemáticos no Ensino Médio, entretanto nem sempre lembram do assunto e com isso precisam de mais tempo para resolver o problema por terem de pesquisar, compreender para então abstrair o problema para o algoritmo e assim desenvolver o código. De acordo com Piva Jr & Freitas (2010), a maioria desses alunos entra no Ensino Superior sem uma base adequada em disciplinas como Português e Matemática o que implica em sérias dificuldades na interpretação de textos e na resolução de equações matemáticas, como citado pelos alunos.

Sobre a terceira afirmação, a intenção é saber o que os alunos percebem sobre a Língua Inglesa para a programação como abordado por Franzen et al. (2018), e tendo em vista que as linguagens de programação geralmente possuem termos em língua estrangeira e as ferramentas IDE possuem versões em inglês. Com os dados obtidos nos questionários, verificamos que, assim como em relação à percepção sobre o conhecimento em matemática, cerca de 60% dos alunos concordam que não ter conhecimento em Língua Inglesa dificulta o processo de aprendizagem, o que vai ao encontro com o que aborda Franzen et al. (2018).

Em relação às entrevistas, os alunos ALU4 e ALU5 citam a falta de conhecimento na Língua Inglesa, como abordado por Franzen et al. (2018), como um fator que dificulta no processo de aprendizagem de programação, seja para a leitura de materiais de apoio, seja para a interpretação de erros no momento da execução de um programa. De acordo com ALU1, a Língua Inglesa não é um conhecimento necessário para aprender a programar, é apenas um facilitador para encontrar material na Internet (ALU1.2), de forma que não percebe, na aprendizagem em si, diferença em conhecer ou não a Língua. Por outro lado, o ALU4 acredita que o inglês é sim necessário para aprender a programar: “Em relação ao inglês, muita documentação, essas coisas está em inglês. [...] então é necessário, é muito necessário na área ter o inglês” (ALU4.5).

De acordo com ALU4, o inglês vai além da programação em si. Ele observa a questão dos termos técnicos da documentação, que fazem parte do dia a dia do profissional de computação. O ALU5 também acha o conhecimento em inglês importante para o aprendizado, principalmente na interpretação de erros ocorridos durante a implementação. Quando são erros mais recorrentes, o aluno cita que até consegue entender, mas quando são erros diferentes ou que ele nunca tinha visto, precisa recorrer a sistemas de tradução.

A visão do professor PROF1 é um pouco diferente da do autor e dos alunos ALU4 e ALU5, para ele "apesar da maioria das programações, das linguagens de programação terem em sua estrutura palavras-chave em inglês e tudo, mas é algo que não é necessário você 'ah, vou mergulhar no inglês para poder entender', não." (PROF1.6). A fala do professor indica que o fato do aluno não saber inglês não pode ser um fator que gera dificuldades para a aprendizagem, não considerando este conhecimento necessário para aprender a programar. Embora haja esta controvérsia entre a necessidade do inglês ou não, a Língua Inglesa é um diferencial no currículo de um profissional da área de Informática. Pode não ser um fator determinante para a aprendizagem de programação, mas facilita no processo de aprendizagem e na inserção no mercado de trabalho.

A quarta afirmação objetiva saber se, para os alunos, a falta de experiência em programação dificuldade o processo de aprendizagem (Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2010). Verificamos que 62,5% dos alunos respondentes dos questionários concordam com os autores. Para o aluno A05, "certo que programação não é somente sintaxe porém para alguém como eu que nunca havia tido contato com programação foi desafiador a linguagem C" (A05.1). De acordo com o aluno A11, "minhas maiores dificuldades aconteceram no início da faculdade por nunca ter visto nada relacionado a programação" (A11.1). O aluno A12 afirma que "em um primeiro momento as maiores dificuldades estiveram relacionadas ao fato de ser o primeiro contato com algoritmos, lógica e o pensamento matemático para resolução de problemas computacionais" (A12.2). Os depoimentos trazidos pelos alunos juntamente com as respostas dos questionários corroboram com o que apresentamos em nossa fundamentação teórica.

Em relação aos dados obtidos com as entrevistas, para o professor PROF1 "muita gente entra na computação e em cursos relacionados e vai programar e tem aquele choque, não sabe o que é isso, não sabe pra quê que é isso" (PROF1.19) e complementa: "não sabe o que ele está fazendo ali [...] não sabem se é realmente aquilo que eles querem da vida, mas chega lá e vê um bocado de comandos, de lógica diferente do que ele está acostumado" (PROF1.26). Já o professor PROF2 acredita que um aluno que nunca programou, mas tem facilidade em lógica não terá muita dificuldade

em aprender a programar (PROF2.12). O aluno ALU2 cita que no início do Curso sentiu muitas dificuldades, e que ainda sente, mesmo já estando no quinto período do Curso, por não ter experiência na área, o que o leva muitas vezes a ter que pesquisar sobre o assunto, antes de resolver um problema. Entretanto, o aluno faz uma reflexão e acredita que estar sempre estudando e pesquisando é normal para todo profissional da área (ALU4.24).

A intenção da quinta afirmação é verificar se a natureza complexa da programação é um fator que prejudica na aprendizagem, como apontado por Gomes & Mendes (2007).

De acordo com as respostas dos alunos, 62,5% concordam com a afirmação, indo de acordo com Gomes e Mendes (2007). Para o aluno A10, a "maior dificuldade em programação é transformar um problema real em um código e poder resolvê-lo" (A10.2). Para o aluno A22, "A parte da programação é relativamente simples, no entanto ao manifestar algoritmicamente a prática, revela ao aluno que programar é um pouco abstrato e que requer treino e persistência" (A22.1). O aluno A29 corrobora com os colegas ao falar que "tive um pouco de dificuldade na abstração de alguns conceitos para passá-los para o código" (A29.2). O professor PROF2 cita que "independentemente do conteúdo, é claro que programação tem uma complexidade maior" (PROF2.10) e o PROF3 avisa que "[Construção de Algoritmos] é a disciplina mais fácil do mundo de entender quando você vê um problema o professor fazendo, mas é muito complexo quando você vai desenvolver" (PROF3.20). Para o professor PROF4 os alunos "conseguem chegar até por exemplo, condicionais, conseguem chegar sem problemas, mas a partir do momento que chega uma parte mais complexa, por exemplo com laços, com vetores, funções, aí é que o pessoal tem mais dificuldade" (PROF4.38). As falas dos professores evidenciam a complexidade da programação, tão abordada e enfatizada no nosso estudo., a qual traz à tona dificuldades enfrentadas por professores e alunos.

A sexta afirmação procura perceber se os alunos compreendem que não aprender um conceito-chave ou conceitos básicos de programação pode dificultar a aprendizagem (Alves et al., 2019; Gomes, 2010; Medeiros, 2019; Moreira et al., 2018; Silva & Trentin, 2016; Aparicio & Costa, 2018; Eltegani & Butgereit, 2015; Piteira et al., 2017; Rezende & Bispo, 2018; A. Santos et al., 2011).

Dos alunos participantes, 90,6% concordaram com a afirmação e nenhum aluno discordou totalmente. Os dados obtidos mostram que na visão dos alunos é relevante compreender todos os conceitos básicos para se ter menos dificuldade em aprender a programar. Para o professor PROF2, "pular etapas, ou tentar entender um assunto sem entender o que tá, o básico dele, o antes dele, então não entender, deixar acumular esses assuntos, sem compreender o anterior, é fatal, dificilmente a pessoa vai conseguir evoluir dessa forma" (PROF2.52).

O professor PROF4 cita o seguinte exemplo para mostrar a necessidade de compreender os assuntos básicos de programação:

> Você vai ver [...] condicionais. Se você não souber os comandos em uma forma sequencial você não consegue entender condicionais, da mesma forma se você não conseguir entender um condicional [...] e você querer aprender laços, onde isso vai acontecer várias vezes, né, então você não vai entender laços, e se você não entender nem condicional nem laços, dificilmente você vai fazer uma função que faça aquilo dali. (PROF4.41)

Dessa forma, aprender a programar é um processo iterativo, em que o assunto ou passo seguinte depende do anterior. O ensino e aprendizagem de programação exige dos professores e alunos um maior esforço, especialmente por abordar assuntos que não estão inseridos no percurso acadêmico do Ensino Fundamental e Médio. Para os professores, dificuldades causadas devido a conhecimentos específicos de programação vão desde a capacidade de abstração do aluno (PROF2.33), a transformação do algoritmo para uma linguagem de programação (PROF5.66), até a adaptação ao uso de ferramentas próprias da programação (PROF5.27). Para os alunos, a aprendizagem é afetada pelos conhecimentos específicos no tocante ao Paradigma Orientado a Objetos (ALU1.23-24), transição de uma linguagem de programação para outra (ALU1.24) e executar o código e solucionar os erros (ALU2.29), como é citado por Gomes (2010), Santos & Menezes (2019) e Silva et al. (2015). A visão do ALU3 corrobora com a do professor PROF5, ao citar que solucionar um problema e conseguir transformar o algoritmo em código fonte é uma dificuldade existente (ALU3.31). O aluno ALU3 acrescenta ainda como dificuldade o desenvolvimento da lógica da programação, indo ao encontro do que é abordado por Moreira et al. (2018).

O aluno ALU4 cita a dificuldade nos assuntos iniciais de programação, como lógica e estruturas de condição, e como organizar o código-fonte (ALU4.24). Para o aluno ALU2, a passagem da UC de Construção de Algoritmos para a Programação Estruturada e Estrutura de dados é uma dificuldade pois, além da mudança de linguagem de programação, o aluno também começa a conhecer o paradigma de programação estruturada (ALU2.27-28). Essa passagem ocorre do segundo para o terceiro período do Curso. A fala do ALU1 complementa a do ALU2 quando este aborda como dificuldade a quebra de paradigma, enfrentada na transição dos conteúdos vistos na UC Programação Estruturada para Programação Orientada a Objetos, que ocorre durante a passagem do aluno do terceiro para o quarto período do Curso (ALU1.23-24). Os alunos ALU1 e ALU4 pontuam o conteúdo de ponteiros como algo difícil de aprender (ALU1.22; ALU4.25). Para o ALU1, mesmo tendo entrado com uma base em programação, sempre existem dificuldades ao longo da trajetória (ALU1.22).

Para o aluno ALU2, outra dificuldade é conseguir debugar o código (ALU2.29), ou seja, como interpretar e resolver os erros que ocorrem durante a compilação e execução do código, fator este relatado por A. J. Gomes (2010) em sua tese de doutoramento que cita que a compilação de um programa também é uma dificuldade enfrentada pelos alunos, bem como o tratamento de erros do código.

Para a sétima afirmação, procuramos saber a percepção dos alunos em relação a turmas numerosas, se isto pode afetar no processo de aprendizagem, como citado por Mendes et al. (2012), A. Santos et al. (2010, 2013) e Souleiman (2017). Pouco mais de 50% dos alunos concordaram que uma turma numerosa dificulta o processo de aprendizagem. Nas entrevistas nenhum professor ou aluno fez alusão a dificuldades relacionadas ao tamanho da turma.

Já em relação à oitava afirmação, que versa sobre à heterogeneidade da turma mais de 50% dos alunos respondentes do questionário discordam com a afirmação de que turmas heterogêneas dificultam o processo de aprendizagem de programação, indo de encontro com o apontado por Mendes et al. (2012), A. Santos et al. (2010, 2013) e Souleiman (2017).

Entretanto, nas respostas abertas, alunos relatam que "companheiros de turma também tinham o costume de mostrar o quanto seus êxitos superavam os fracassos de outros (que me incluía, também, na lista dos fracassos), isso era muito desmotivador" (A02.2) e que se sentia "um pouco envergonhado [por ter dificuldades] pois tinha bastante gente na turma que tinha uma facilidade de entendimento bem maior que o meu" (A30.1). Para o aluno A15,

> Infelizmente cada aluno apresenta condições e níveis bastante variados fazendo com que o professor tivesse que ir e voltar durante a aula para englobar a todos. Esse processo fazia com que alguns se perdessem e até mesmo se sentissem confusos e desmotivados ao aprendizado. (A15.1)

As falas dos três alunos apresentam diferentes impactos devido à heterogeneidade da turma: para os alunos A02 e A30 a heterogeneidade trouxe dificuldades pessoais, por perceber que os demais colegas tinham maior facilidade de aprender, lhes trazendo constrangimento e desmotivação. Já para o aluno A15, os diferentes perfis de alunos trazem dificuldades para o professor acompanhar os alunos, causando desmotivação no processo de aprendizagem. Embora as situações abordadas pelos alunos sejam diferentes, corroboram com os autores sobre as dificuldades desencadeadas pela heterogeneidade da turma.

Os dados coletados nas entrevistas apontam que a heterogeneidade é citada pelos professores PROF2, PROF3 e PROF4 e pelo aluno ALU3. Em tom de desabafo, o PROF2 aponta que "É duro, é

difícil ensinar programação por que nem todos têm o mesmo nivelamento" (PROF2.31), isso porque exige do professor a adoção de estratégias que possam motivar os alunos dos diferentes níveis (seja de conhecimento, seja de experiências), para que tanto alunos que não tenham nenhum conhecimento em programação quanto aqueles que já programavam antes de entrar no curso possam participar de forma efetiva do processo de aprendizagem. Para tanto, muitas vezes é necessária a preparação de material didático (conceitos, exercícios, trabalhos) diferentes para contemplar os diferentes perfis de alunos, até que atinjam um nivelamento adequado para se conseguir equiparar a forma de ensino, o que acaba exigindo maior esforço por parte do professor.

Para o professor PROF4 a heterogeneidade da turma pode atrapalhar o andamento da aula uma vez que o professor "fica naquele dilema se continua por que alguns já conseguiram ou se você espera e aí quem já terminou vai ficar sem ter o que fazer... então fica algo bem complicado" (PROF4.36). Para o professor PROF3 o problema causado pela diferença de níveis dos alunos ocorre

> Principalmente em construção de algoritmo, que é uma disciplina básica, [...] chegam alguns alunos que vêm já tendo noção de programação, outros chegam sabendo programar, e outros chegam sem nunca terem visto nada, aí torna que fica difícil a aula porque ou tá muito básica pra uns, ou tá muito complexa pra outros, e a gente tem que equilibrar, então fica difícil às vezes de aplicar alguma outra estratégia". (PROF3.22)

Esta fala do professor remete à questão de as turmas terem alunos de diferentes níveis, o que dificulta a escolha da metodologia docente a ser adotada no processo de ensino de programação.

No que diz respeito à nona afirmação, sobre as dificuldades decorrentes da autoconfiança baixa (Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015), todos os alunos concordam com a afirmação, sendo que 81,25% concordam totalmente e 18,75% concordam em parte. Para o aluno A02, "devido a problemas pessoais acabei desenvolvendo uma 'baixa estima' sobre o quando eu poderia aprender e fazer [...] por isso acabava não comparecendo às aulas ou até trancando" (A02.3). Para o aluno, a baixa estima o prejudicou no processo de aprendizagem, desencadeando suas ausências nas aulas e desistência da UC, indo ao encontro com o que apontam os autores acima citados. Aliada à autoestima baixa, para o professor PROF4, a falta de persistência é uma atitude negativa dos alunos, segundo ele, "não são persistentes, [...] assim que eles têm a primeira dificuldade a atitude deles é [...] vou desistir, vou trancar a disciplina [...], então você tem muito mais problemas com relação a isso" (PROF4.60).

Em relação décima afirmação, que versa que a desmotivação do aluno dificulta o processo de aprendizagem, como apresentam A. Gomes & Mendes (2015), Mendes et al. (2012), Ortiz-Ortiz et al.

(2018), Piteira et al. (2017), A. Santos et al. (2013) e Skalka & Drlik (2018), os dados obtidos com o questionário apontam que todos os alunos concordam com a afirmação, sendo que 90,6% concordam totalmente e os demais concordam em parte. O aluno A30 corrobora com os autores no seguinte relato: "quando aconteceu de eu ter perdido o foco em apenas uma aula, virou uma bola de neve, e eu não conseguia acompanhar o conteúdo" (A30.2).

Para o professor PROF2, "o maior problema que um aluno, principalmente nessa, na disciplina de programação, é ele estar desmotivado por algum motivo" (PROF2.37). E continua: "às vezes a desmotivação tem a ver com outro fator, é a própria dificuldade dele, mas se ele tiver minimamente motivado ele pode superar essa dificuldade [...] eu acho que o que mais atrapalha é a falta de motivação" (PROF2.38), que, segundo ele, pode desanimar os próprios colegas e até mesmo o próprio professor. O professor PROF4 concorda com o PROF2 ao falar que "a questão de desmotivação, né... você tem uns alunos que são muito desmotivados [...] a grande questão é que a gente tem uma quantidade maior de alunos que são desmotivados" (PROF4.59).

A décima primeira afirmação tem como objetivo conhecer a visão do aluno acerca da importância da interpretação dos problemas para a programação (Medeiros, 2019; Moreira et al., 2018; Nobre & de Menezes, 2002; Santos & Menezes, 2019; da Silva et al., 2015; Tavares, 2018; Aparicio & Costa, 2018; Bosse & Gerosa, 2017; T. H. Castro et al., 2008; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015, 2010; Mendes et al., 2012; Ortiz-Ortiz et al., 2018; Rezende & Bispo, 2018; Souleiman, 2017). De acordo com os dados obtidos com os questionários, 93,7% dos alunos entendem que não conseguir interpretar um problema dificulta a aprendizagem de programação. Essa habilidade também pode estar associada ao *background* do aluno, especialmente à dificuldade em interpretação textual (Franzen et al., 2018).

Para o professor PROF5, "se a gente for analisar com bem clareza, é onde reside a dificuldade de programação: é entender o que se pede" (PROF5.23). O professor PROF3 fala que os alunos precisam entender por que eles têm que fazer algo e não simplesmente fazê-lo (PROF3.50). O PROF1 conclui que uma aptidão importante para o programador é perceber o problema e abstraí-lo (PROF1.6). A preocupação dos professores corrobora com as dos autores citados uma vez que um dos pilares da programação é a resolução de problemas e só se consegue resolver um problema corretamente se conseguir interpretá-lo.

A intenção da décima segunda afirmação é perceber se os alunos reconhecem que não estudar constantemente e de forma profunda pode prejudicar o seu processo de aprendizagem, como afirmado por Biggs & Tang (2011) e Clark et al. (2011). Como resultado, verificamos que todos os

alunos estão em concordância de que a forma como estuda programação influencia no processo de aprendizagem, sendo que 90,6% concordam totalmente com a afirmação. Neste caso, percebe-se que os alunos reconhecem que dedicar pouco tempo de estudo, ou estudar sem muito aprofundamento, é um fator que dificulta a aprendizagem de programação.

O professor PROF1 aponta que a forma como o aluno estuda, por exemplo, com celular do lado, assistindo televisão, pode dificultar na aprendizagem (PROF1.22). O PROF5, por sua vez, cita que a falta de comprometimento dos alunos, em querer estudar a fundo, dificulta também o processo de aprendizagem (PROF5.37). A fala do segundo professor remete ao que abordam Biggs & Tang (2011) e Clark et al. (2011), sobre a superficialidade e passividade dos alunos na forma de estudar.

Para o professor PROF3, se os alunos estudam "reproduzindo o que o professor faz eles nunca vão aprender" (PROF3.12) e dá o seguinte conselho aos alunos:

> Não ache que vocês estão aprendendo comigo fazendo, vocês têm que aprender com vocês resolvendo, por aqui eu tô [Sic] desempenhando o meu raciocínio, e não é um raciocínio único, cada um pode ter um raciocínio diferente, você tem que desenvolver o seu para resolver os mesmos problemas que eu resolvi usando esse aqui. (PROF3.15)

As dificuldades na aprendizagem podem se dar por algumas 'atitudes dos alunos', como por exemplo fazer cópias dos trabalhos dos colegas, ou pedir para alguém fazê-los por eles (PROF3.25), prejudicando, além da aprendizagem, o processo de avaliação por parte dos professores.

Para o professor PROF1, uma atitude do aluno que pode atrapalhar no processo de aprendizagem é achar que já sabe de um assunto e por isso não se interessar no que o professor está ensinando, principalmente quando são conceitos mais básicos e iniciais (PROF1.28), sendo que, como apresentado anteriormente, os conceitos básicos são essenciais para a aprendizagem de programação.

O ALU3 cita como fator que pode acarretar a impossibilidade de estudar constantemente é disponibilidade do aluno. O ALU3 sempre trabalhou durante todo o Curso e percebe a diferença entre ser um estudante trabalhador e um estudante com dedicação exclusiva para o estudo. Para ele, estar integralmente dedicado ao estudo, permite que o aluno possa aprender mais e da forma melhor.

A falta de tempo para se dedicar aos estudos é um dos principais fatores que podem prejudicar o processo de aprendizagem de programação (Giraffa & Mora, 2013; Medeiros, 2019; Moreira et al., 2018), levando muitas vezes os alunos à desistência ou reprovação, como ocorreu como ALU3. Entretanto, este fator o motivou a estabelecer um horário e a organizar o tempo de estudo, o que foi fundamental no seu processo de aprendizagem.

A décima terceira afirmação refere-se à importância existente entre o que é visto em sala de aula e a contextualização com o mundo real (Al-Imamy et al., 2006; Mendes et al., 2012; Rezende & Bispo, 2018). De acordo com as respostas do questionário, 65,63% dos alunos concordam que o hiato entre a teoria e a prática dificulta o processo de aprendizagem de programação, corroborando com os autores citados. O aluno A03 cita que desistiu da UC porque "o professor não conseguiu apresentar um uso real daquela linguagem, ter que aprender algo que você sabe que nunca vai usar é desmotivador" (A03.1). Os resultados dos questionários mostram que os alunos sentem falta de perceber onde os conceitos vistos em sala de aula podem ser aplicados no dia a dia, para se ter maior motivação e sucesso na aprendizagem. O ALU5 revela que desistiu de uma UC porque "não conseguia associar aquilo que o professor estava falando ao nosso cotidiano [...] não estava conseguindo associar nada a nada" (ALU5.28). Outra fala do aluno ALU5 fala da dissociabilidade que ocorre entre a teoria e a prática: "e eu ficava: 'pra que é que eu vou estudar isso se eu não sei nem onde é que eu vou usar?', e aí eu comecei a me decepcionar com a matéria" (ALU5.21).

Para ele, não estar clara a aplicabilidade na prática do que é visto em sala de aula prejudica a sua aprendizagem. É papel do professor apresentar problemas e situações práticas, sempre que possível, para que o aluno compreenda melhor os assuntos ministrados. Para Freire (1987), "a verdadeira reflexão crítica origina-se e dialetiza-se na interioridade da 'práxis' constitutiva do mundo humano" (p. 15). Dessa maneira, por meio da relação teoria e prática, pode-se chegar a uma verdadeira reflexão crítica, realizando assim, através da educação, práticas transformadoras da realidade em que o sujeito se encontra. Para o autor, a práxis é vista como um instrumento utilizado para que seja possível ocorrer uma transformação verdadeira, pois é necessário conhecer a realidade para agir sobre ela. Freire (1987) ainda afirma que a práxis é reflexão e ação dos homens sobre o mundo para transformá-lo.

Para tentar dirimir este problema, da distância entre teoria e prática, uma ação importante que vem sendo desenvolvida pelos docentes do Curso são os Projetos Integradores. Trata-se de um projeto interperíodos, a ser desenvolvido dentro do semestre letivo, que envolve o conteúdo das disciplinas que estão sendo trabalhadas dentro dos períodos do Curso.

> Além de promover a articulação entre a teoria e prática, os projetos integradores têm incentivado a inovação e o empreendedorismo entre os discentes dos vários períodos do Curso, incluindo a isso atributos necessários ao mercado de trabalho, tais como, liderança, trabalho em equipe, proatividade, dentre outros. (PPC, 2019, p. 26)

Assim, devido à sua natureza complexa, o processo de ensino e aprendizagem de programação exige esforço por parte dos docentes em busca de estratégias e metodologias diversificadas que auxiliem na efetividade da aprendizagem, bem como uma grande dedicação por parte do estudante para aquisição dos conhecimentos e da habilidade prática necessária (Coutinho et al., 2018).

A décima quarta afirmação diz respeito à ocorrência de dificuldade na aprendizagem devido à linguagem de programação adotada pelo professor (Aparicio & Costa, 2018; Bosse & Gerosa, 2017; A. Gomes & Mendes, 2015; Piteira et al., 2017). Com os dados obtidos, foi expressivo o número de alunos que discordam dessa afirmação (46,8%), entretanto a maioria acredita que a linguagem de programação adotada pelo professor pode dificultar o processo de aprendizagem. Sobre este aspecto, o aluno A05 apresentou o seguinte depoimento:

> Estudei a matéria de construção de algoritmos duas vezes. Reprovei na primeira e justifico meu fracasso com a linguagem que foi abordada que foi a C. O professor sempre era dedicado em repassar o conhecimento. Porém a linguagem em si se tornou um problema. (A05.2)

Para além das dificuldades já apresentadas neste parâmetro, as vozes dos professores indicam problemas no tocante à formação docente (PROF5), infraestrutura do curso (PROF2, PROF4, PROF5) e o próprio perfil do aluno, que muitas vezes entram no curso sem ter noção no que será formado (PROF1, PROF2, PROF4, PROF5). Na opinião dos alunos, outras dificuldades referem-se à metodologia adotada pelo professor (ALU1, ALU3, ALU5) e ao material e ferramentas adotadas (ALU1, ALU5, ALU6). As três falas a seguir, remetem a três diferentes alunos que falam da visão destes em relação à metodologia dos professores de programação:

> Pelo que eu me lembro eu não estava conseguindo acompanhar muito as aulas do professor [que deu a matéria]. [...] eu fiquei meio desmotivado, mas não foi nem a matéria em si, foi o modo como foi dada... Eu acho que sim, a metodologia foi o que mais pesou. (ALU1.25)
>
> porque só a teoria eu ficava 'voando' sem compreender mais ou menos o tipo de assunto. (ALU3.26)
>
> Ele não tinha uma didática muito boa. (ALU5.22)

Para o ALU1, o método do professor não o estimulava o desmotivando a ir às aulas e compreender o conteúdo dado. Para o aluno, o problema não era o conteúdo, mas a forma como o professor ministrava suas aulas. Na perspectiva do aluno ALU3, a forma como o professor apresenta o conteúdo o fazia não estar centrado na matéria, não entendendo o assunto que estava sendo dado. Podemos ver que as falas dos alunos convergem ao entendimento de que o papel do professor é fundamental para a sua aprendizagem. Percebemos que por mais que existam materiais disponíveis,

aulas on-line com outros docentes e os próprios colegas de sala, é o professor da Unidade Curricular o principal responsável pela formação daquele aluno.

De acordo com ALU5, as ferramentas de apoio adotadas pelo professor também podem atrapalhar o processo de aprendizagem de programação:

> Teve um professor que na programação Java ele indicou IntelliJ, acho que é esse o nome, ele começou usando esse, mas nos laboratórios que a gente usava, os computadores disponibilizados para a gente só tinham o NetBeans, então teve um momento que a gente não estava conseguindo acompanhar muito bem por que ele indicava coisas que no IntelliJ eram ícones diferentes e aí a gente perdia muito tempo tentando encontrar aquela função no NetBeans. (ALU5.17)

Percebemos que o professor utilizou uma ferramenta que era diferente da disponibilizada aos alunos nos laboratórios de ensino de programação. Com isso, na tentativa de ensinar com uma ferramenta, enquanto os alunos utilizavam outra com o mesmo propósito, acabou por prejudicar o desenvolvimento e aprendizagem dos alunos. É claro que, enquanto programadores, devemos ser capazes de nos adaptar às mais diversas ferramentas que existem, uma vez que o que mudam são algumas funcionalidades, mas a lógica da programação em si é a mesma. Entretanto, usar ferramentas diferentes, entre alunos e professor, pode causar mais dificuldades do que benefícios.

Outros fatores que podem gerar dificuldades no processo de ensino e aprendizagem são questões familiares e atitudes negativas dos alunos. Os fatores pessoais são diferentes, mas podem estar entrelaçados, de forma que um fator pode desencadear outro. As vozes dos professores e alunos mostram que: “a pressa, a descrença em si próprio, você achar que não pode, isso aí é uma atitude negativa, isso aí é uma atitude que não vai levar você a desenvolvimento nenhum” (PROF1.21), que “é uma fase difícil da vida que você às vezes tá [Sic] com um relacionamento difícil ou com a família, [...] você está cheio de decisões importantes na vida” (PROF2.40) e que há ainda a “condição psicológica, emocional, de lidar com problemas, de como resolver aquele problema, se ele consegue ou não resolver o problema [...] a pessoa e fica criando bloqueios para que ela não consiga resolver”(PROF4.39). O professor PROF4 e o aluno ALU2 citam as questões familiares como exemplos que podem originar dificuldades no processo de ensino e aprendizagem:

> E tem a questão familiar, né? Que às vezes você não tem o incentivo pra fazer, pra estudar [...] Se tiver filhos, se tiver obrigações dentro da própria casa, ou seja, tem que sustentar a casa e tudo isso aí contribui negativamente. (PROF4.50-51)

> Porque eu vejo muitos alunos que 'ah, tô querendo desistir do Curso, e tal', mas não é nem questão do Curso em si, é geralmente ou porque está tendo problema familiar ou não sabe ainda por exemplo a área que quer dentro do Curso. (ALU2.30)

As dificuldades enfrentadas pelos professores e alunos ao longo do processo de ensino e aprendizagem podem desencadear diferentes consequências no percurso acadêmico e na vida dos envolvidos no processo. A categoria a seguir apresenta os dados obtidos sobre este assunto.

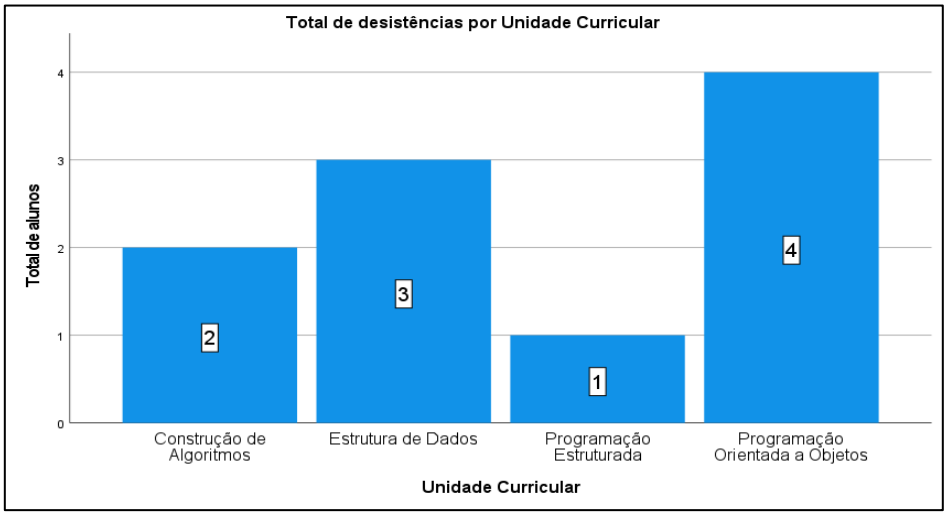
4.2.5.2. Consequências das Dificuldades. Para esta categoria, fizemos duas perguntas:

1- Você já desistiu de alguma UC de programação? Se sim, qual(is)?

2- Você já reprovou em alguma UC de programação? Se sim, qual(is)?

Em relação à primeira pergunta, as desistências informadas pelos participantes dos questionários, o Gráfico 4 apresenta o número de desistentes em cada uma das UC de programação.

Gráfico 4 - *Desistências de alunos por Unidade Curricular*



Nota: Autoria própria

De acordo com os dados obtidos, sete alunos (21,9%) afirmaram já ter desistido de alguma UC de programação ao longo do Curso. De acordo com o Gráfico 4, não houve ocorrência de desistência da UC Programação Avançada. Observamos que o número de desistências por UC é superior ao número de alunos que afirmaram ter desistido, o que implica dizer que um aluno pode ter desistido de mais de uma UC ao longo do Curso.

Diante das dificuldades, a desistência pode ocorrer em relação a uma UC específica de programação ou até mesmo do curso. Para contextualizar a análise, consideramos sinônimos os termos desistir, abandonar e trancar uma disciplina. Para o professor PROF4, os alunos "assim que eles têm a primeira dificuldade a atitude deles, é mais a principal é vou desistir, vou trancar a

disciplina, vou desistir do curso, vou trancar o curso, então você tem muito mais problemas com relação a isso" (PROF4.60). Na opinião do professor PROF3,

> como essas disciplinas de programação acontecem muito no início do Curso, muitas vezes desmotivam os alunos a desenvolver o restante do Curso. Então às vezes alguns alunos chegam em Construção de Algoritmos, aí são reprovados, e aí tentam pagar de novo, e são reprovados, e aí eles abandonam o Curso, desistem, ou deixa pra lá a área de programação achando que pode pagar a qualquer hora e vão desenvolver outras disciplinas que não precisam muito de programação, mas o nosso Curso basicamente ele vai precisar de programação todos os semestres e ele começa a não se dar bem em nenhuma disciplina, e isso aí é um fator gerador de desmotivação, mas que não começa nas disciplinas de programação, é que começam nas disciplinas de cálculo que vêm antes da programação, por que as disciplinas de cálculo têm o mesmo perfil que eu te falei das dificuldades. (PROF3.34)

Para o professor PROF2, as dificuldades enfrentadas podem causar a desistência do Curso, o que pode afetar tanto o percurso acadêmico do aluno quanto pode influenciar na vida do aluno fora da Universidade.

O trecho em que o aluno ALU1 fala "e muita gente desistiu" (ALU1.29), diz respeito às dificuldades enfrentadas na Unidade Curricular Programação Orientada a Objetos. Mesmo com o baixo rendimento, o que levou boa parte da turma a quarta avaliação, o aluno ALU1 afirma que não desistiu, e no final obteve êxito. Entretanto, ele afirma que muitos desistiram, para não enfrentarem a pressão de mais uma avaliação.

Já os alunos ALU3 e ALU5 afirmam que pensaram em desistir logo no início do Curso (ALU3.37; ALU5.25). Para o ALU3 essa vontade ocorreu por sentir-se em um nível de conhecimento inferior em relação aos demais colegas. No início do curso ele não se sentia parte daquele grupo, seja pelo fator idade, seja por entrar no Curso sem saber o que seria visto pela frente:

> Na primeira semana do curso deu vontade de desistir. Porque é [breve pausa], quando o pessoal começou a falar que seria matemática pura e aplicada e que seria muito difícil, então eu [pensei]: 'aqui não é minha área'. [...] Muitas das vezes eu dizia assim: 'não, eu estou no canto errado, não é possível... que o pessoal que já tinha bagagem, que já tinha mexido com programação, resolvia tão rápido e eu ficava lá, como se diz, martelando o dedo, eita! É porque eu estou no canto errado, isso aqui não é para mim'. (ALU3.37-39)

Além disso, o aluno apresenta que seu *background* foi uma das dificuldades que o levou em pensar em desistir do Curso, bem como sentir-se que estava na área errada, devido ao conteúdo presente na matriz curricular do Curso.

A desistência do ALU5 se deu pela dificuldade em conseguir adaptar-se ao contexto universitário e conciliar suas atividades, bem como ao seu baixo rendimento no primeiro semestre do Curso:

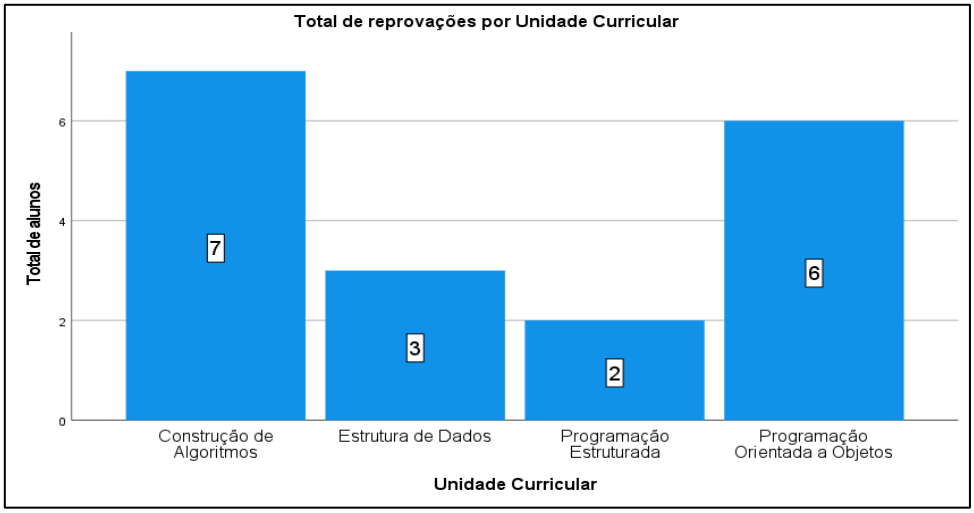
> Eu pensei em desistir logo quando eu entrei no Curso, no primeiro semestre. [...] Então eu pensei realmente em desistir, e então eu terminei o primeiro semestre me arrastando, e aí eu tranquei antes de iniciar o segundo semestre, e então eu resolvi parar e dar um tempo. (ALU5.25-26)

Reconhecer seu limite e 'trancar' o semestre, ou seja, desistir antes de iniciá-lo naquela altura, ajudou o aluno a se reestruturar emocionalmente e se reorganizar para retomar posteriormente ao Curso. Para ele, dar esse tempo foi fundamental para que ele pudesse ter melhor aproveitamento nas disciplinas de programação e no Curso.

Sobre a segunda pergunta, sobre a reprovação, de acordo com os dados obtidos com os questionários, 13 dos 32 alunos respondentes afirmaram já ter reprovado alguma UC de programação, totalizando 40,6% dos respondentes. O Gráfico 5 apresenta, por UC, o total de alunos que informaram já ter reprovado.

Gráfico 5

Reprovações de alunos por Unidade Curricular



Nota: Autoria própria

O número de reprovados é bastante expressivo e aponta ser um fator preocupante e recorrente. Da mesma forma que observamos sobre as desistências, o número de reprovações por UC

é superior ao número de alunos que afirmaram ter reprovado, o que implica dizer que um aluno pode ter reprovado em mais de uma UC ao longo do Curso. Nenhum dos alunos participantes da pesquisa afirmou ter reprovado na UC Programação Avançada.

Ainda de acordo com o Gráfico 5, podemos observar que as reprovações estão concentradas em Construção de Algoritmos e em Programação Orientada a Objetos. Estas UC têm peculiaridades que podem justificar a alta taxa de reprovação: enquanto Construção de Algoritmos é o primeiro contato do aluno com uma Unidade Curricular de programação no Curso, Programação Orientada a Objetos é uma Unidade Curricular em que há a mudança de paradigma e linguagem de programação adotada, o que pode causar mais dificuldades aos alunos. Para o professor PROF2, "não são raras as situações de alunos de computação que pagam algoritmos 3, 4 vezes, pra poder conseguir, né? Ele às vezes se matricula 4, 5 vezes pra poder conseguir. Fora aqueles que desistem" (PROF2.63). O aluno A03 faz o seguinte relato: "Eu reprovei pois eu me sentia desmotivado, sem foco, sem vontade" (A30.3).

Dos seis alunos entrevistados, dois haviam sido reprovados em pelo menos uma Unidade Curricular. O aluno ALU1 associa sua reprovação em Programação Estruturada à forma como o professor lecionou a disciplina, o que lhe causou desmotivação até ser reprovado (ALU1.30).

As altas taxas de reprovação e desistência em UC de programação é uma situação que afeta principalmente os alunos novatos, pois estas UC geralmente são o início dos cursos. De acordo com Mendes, Paquete, Cardoso, & Gomes (2012), muitos autores sugeriram possíveis causas para as dificuldades dos alunos, como a natureza da programação, o histórico dos alunos e as atitudes de estudo bem como as estratégias pedagógicas comumente usadas em disciplinas de algoritmos e programação. Foi o que aconteceu com o aluno ALU3, que sofreu uma reprovação logo no primeiro semestre do Curso: "logo no início eu reprovei lógica. O pessoal dizia assim: 'não reprove lógica' aí eu fui, consegui passar essa fase, aí eu reprovei em programação, Programação Orientada a Objetos" (ALU3.41).

A Unidade Curricular Lógica Matemática Aplicada à Computação, apesar de não abordar conteúdos de programação, é pré-requisito para Construção de Algoritmos, ofertada no segundo semestre do Curso (PPC, 2019). Por reprovar nesta Unidade Curricular, o aluno ALU3 acabou atrasando um ano, uma vez que a oferta das UC obrigatórias no Curso é feita anualmente.

Outra consequência das dificuldades enfrentadas pelos alunos é a desmotivação. Para o professor PROF3, "o primeiro período que possui muitas disciplinas de matemática, aí já vem uma

desmotivação, e aí quando chega em algoritmos ele já chegam desmotivados em algoritmos, não conseguem desenvolver o raciocínio algorítmico, e isso aí desestimula mais ainda" (PROF3.35).

Para o ALU1, a desmotivação veio devido à metodologia do docente, o que o levou a faltar muitas aulas. Para o ALU2, não saber a área que vai seguir dentro da carreira de profissional na área de Informática o deixa desmotivado e "meio perdido" e o ALU5 atribui sua desmotivação ao fato de ter que mudar sua rotina e adaptar-se à Universidade.

Com as dificuldades, outra consequência é o baixo rendimento dos alunos nas disciplinas de programação, como apresentado nas falas dos alunos ALU1 e ALU5. Para o ALU1, o baixo rendimento foi provocado pelas dificuldades em POO, enquanto para o aluno ALU5, o baixo rendimento foi decorrente da relação entre a Universidade e o estágio, bem como à adaptação à nova rotina de estudo. De acordo com o ALU1, "Desistir não, mas, programação eu não lembro se foi avançada, ou se foi POO, mas eu e quase a turma toda ficou na 4ª prova, precisando de muito" (ALU1.28), indicando o baixo rendimento. Já o ALU5 fala que "o meu rendimento na [nome da Instituição omitida], as minhas notas do primeiro semestre da [nome da instituição omitida] foram muito, muito baixas [...] e aquilo me afetou emocionalmente, e eu fiquei muito triste, porque não estava acostumado" (ALU5.26).

De acordo com o ALU1, as dificuldades afetaram não apenas ele, mas a turma como um todo, de forma que a maioria dos alunos foi para a quarta avaliação, devido às notas baixas da UC, enquanto para o aluno ALU5, o baixo rendimento afetou-lhe emocionalmente, por ser uma situação nunca enfrentada anteriormente em sua trajetória acadêmica. O ALU1 ainda cita que as dificuldades enfrentadas causaram 'trauma' em uma linguagem de programação específica. Segundo ele,

> Até hoje tenho algum trauma de Java [risos], [...] hoje em dia praticamente eu não mexo com Java, é uma linguagem que realmente eu não sou muito familiarizado, eu acho que boa parte disso foi das dificuldades que eu tive nas matérias de POO, eu acho que com certeza isso. (ALU1.27)

O contexto da fala do aluno diz respeito às dificuldades enfrentadas com a quebra de paradigma de programação estruturada para o paradigma orientado a objetos, que está vinculada à mudança de professor e da linguagem de programação adotada.

Embora as dificuldades, em sua maioria, tragam transtornos e consequências negativas para a vida acadêmica dos estudantes, dois alunos conseguiram enxergar a motivação diante dos desafios de aprender a programar:

> No caso do C, que é Estrutura de Dados que eu aprendi em C, na verdade me motivou mais a tentar entender mesmo ponteiros, porque era algo muito difícil para um aluno do segundo período, mas no segundo período a gente ainda está naquela de ainda está no pique para tentar entender as coisas, então me motivou mais. (ALU1.26)
>
> Assim, acho que talvez um aproveitamento delas teria sido melhor se eu não tivesse essas dificuldades, mas de certa forma a dificuldade gera o melhoramento, tipo, do profissional, do aluno, então acho que também é necessário de certa forma, mas se eu não tivesse essa barreira, eu teria ido mais além, teria um melhor aproveitamento por exemplo, ou seja, tipo, eu poderia ter ido procurar mais documentação em inglês se eu soubesse mais, ou como perguntar tipo, tem coisas que às vezes você não sabe, como ah o que é que é isso, tipo, achar aquela resposta, isso ajuda muito, e também a matemática às vezes. (ALU4.29)

Ambos os alunos refletem positivamente sobre as diferentes dificuldades enfrentadas durante a aprendizagem de programação. Para o ALU1, a motivação vem devido à vontade de aprender e superar os desafios, e associa esta motivação ao fato de ainda estarem no início do Curso. Já o ALU4, confessa que, se não fossem as dificuldades, certamente poderia aproveitar melhor o Curso, mas reconhece que as dificuldades melhoram também o desempenho dos alunos e do profissional.

4.2.5.3. Conclusão. O parâmetro 'Porque' nos permite claramente 'identificar as dificuldades enfrentadas por professores e alunos durante o processo de ensino e aprendizagem, e o que estas dificuldades acarretam', uma vez que nos apresenta evidências sobre as principais dificuldades vivenciadas por professores e alunos ao longo do processo de ensino e aprendizagem de programação e as consequências trazidas por estas para ambos.

Na categoria 'dificuldades enfrentadas', a triangulação dos dados revela que a adaptação no Ensino Superior é uma dificuldade vivenciada por alunos por evidenciar problemas ocorridas na sua formação básica, dificuldades de adaptar-se às novas rotinas e ao ambiente acadêmico, as fontes de dados convergem para este aspecto e podemos considerar, de fato, como uma dificuldade. Em relação à dificuldade trazida pela falta de conhecimento em matemática, percebemos que alunos e professores também estão de acordo com essa afirmação, corroborando com a fundamentação teórica. Em relação às dificuldades enfrentadas pelo pouco conhecimento em inglês, professores e alunos divergem entre si: os alunos creditam algumas de suas dificuldades para compreender a programação, já na visão de um dos professores, não saber inglês não deve ser um fator que impeça o aluno de aprender a programar. Embora haja esta controvérsia entre a necessidade do inglês ou não, a Língua Inglesa é um diferencial no currículo de um profissional da área de Informática. Pode não ser um fator determinante

para a aprendizagem de programação, mas facilita no processo de aprendizagem e na inserção no mercado de trabalho, por isso, podemos inferir que embora não seja um fator sem o qual não se aprenda, é um fator que pode abrir possibilidades para quem o tem. No tocante à dificuldade causada pela falta de experiência, é certo que, no início do curso alunos que nunca tenham programado sintam mais dificuldades em aprender em comparação com aqueles que já tenham tido experiências ou trabalhem da área de desenvolvimento. Um dos maiores problemas da falta de experiência se dá pelo fato dos alunos ingressarem no curso sem saber o que um profissional da área de Informática faz, o que em alguns casos pode gerar frustrações, desmotivação e desistência do curso. Por outro lado, um aluno que entra no curso sem nunca ter tido contato com a programação entra sem hábitos próprios e assim pode ser 'moldado' com o rigor acadêmico para aprender programação, dessa forma pode ser um aspecto favorável para estes alunos. A triangulação dos dados sobre este aspecto mostra uma preocupação de professores e alunos em relação à entrada do curso sem experiência anterior, corroborando com o que apresentamos na fundamentação teórica.

De acordo com as análises, percebemos que a complexidade da programação e a não compreensão dos conceitos básicos trazem dificuldades para o processo de ensino e aprendizagem. É fato que o ato de programar envolve competências, conhecimentos e habilidades muitas vezes negligenciadas ao longo do percurso acadêmico do aluno, o que o torna complexo. A programação é iterativa e recursiva, de forma que, para avançar na aprendizagem não se pode ignorar os conceitos básicos, uma vez que estarão sempre presentes. Podemos afirmar que estes dois fatores trazem dificuldades a alunos e professores, com base na triangulação dos dados, o que converge com o que abordamos na fundamentação teórica.

Sobre as turmas numerosas, as fontes de dados não nos permitem evidenciar que estão de acordo com este fator dificultar o processo de aprendizagem. Entretanto, no tocante ao processo de ensino, turmas numerosas podem prejudicar o professor na gestão da turma, no apoio individualizado aos discentes e ainda na avaliação formativa destes, trazendo desafios e dificuldades para o processo de ensino e aprendizagem. Por outro lado, pode-se pensar em estratégias para dirimir este fator, como divisão da turma ou adoção de estratégias que possibilitem um melhor acompanhamento do professor. Já em relação à heterogeneidade da turma, a visão dos alunos respondentes do questionário vão de encontro com o que apresentamos na fundamentação teórica e com a percepção e fala dos professores. Se por um lado esses alunos acreditam que os diferentes perfis e níveis de conhecimento dos alunos não é um fator que dificulta a aprendizagem, outros alunos depoentes trazem experiências negativas vivenciadas, decorrentes desse fator. Da mesma forma, professores afirmam que a

heterogeneidade da turma pode acarretar dificuldades, especialmente nos anos iniciais do curso. Apesar da triangulação nos trazer visões divergentes, é possível deduzir que, dependendo do que é experienciado por professores e alunos, a heterogeneidade pode trazer dificuldades no processo de aprendizagem.

Sobre a autoconfiança baixa e a desmotivação, a triangulação dos dados nos mostra claramente que estes são fatores que trazem dificuldades para alunos e professores. O aluno com autoconfiança baixa e desmotivado não consegue estar focado e se fazer presente e participante do processo de aprendizagem. O professor que ensina alunos que estão com essas características também pode sentir dificuldade em gerir a situação, e podendo ficar desmotivado. As interpretações trazidas convergem, portanto com o que notamos na fundamentação teórica. Outro fator que a triangulação dos dados nos permite afirmar é que não conseguir interpretar os problemas a serem resolvidos gera dificuldades no processo de aprendizagem de programação. Um dos principais objetivos do desenvolvimento de software é solucionar problemas, dos mais simples aos mais complexos, entretanto, para que esta solução esteja a contento, é preciso desenvolver um sistema informático que possa cumprir os requisitos necessários, o que só se consegue fazer com a interpretação correta do que se pede. Não entender o que é solicitado ou não conseguir interpretar o problema afeta diretamente no processo de aprendizagem. Da mesma forma, não estudar com profundidade dificulta o processo de ensino e aprendizagem de programação, especialmente pela natureza complexa da programação e pela necessidade de se ter um conhecimento dos conhecimentos básicos muito forte para poder avançar. Atitudes dos alunos tais como replicar o que o professor passa em sala de aula ou pedir para que colegas façam seus trabalhos, mascaram o que eles acreditam ser exitoso, por conseguirem cumprir a Unidade Curricular, sem estarem aprendendo a programar. A triangulação dos dados nos revela que alunos e professores reconhecem que a falta de compromisso ou de tempo para estudar afeta o processo de aprendizagem, corroborando com o trazem os autores citados na fundamentação teórica.

A falta de relação entre a teoria e a prática também dificulta o processo de aprendizagem e pode causar desmotivação nos alunos que não percebem a aplicabilidade do conteúdo visto nas UC. Sobre a linguagem de programação adotada pelo professor, os dados da triangulação não nos trazem evidências, uma vez que as entrevistas não revelam este aspecto e os dados dos questionários estão muito equilibrados entre os que concordam e os que não concordam. Entretanto, analisando este fator no tocante à fundamentação teórica, podemos refletir que não existe a linguagem ideal nem o método ideal de se ensinar programação. Alunos podem sentir dificuldades devido a uma linguagem, enquanto

outros alunos não sentem. Ainda, corroborando com as falas dos professores em outras categorias, mais importante do que aprender a sintaxe ou semântica de uma linguagem, é aprender a lógica da programação e pensar algoritmicamente, para que o aluno seja capaz de aplicar os conhecimentos na linguagem mais adequada para solucionar o problema.

Para além das dificuldades elencadas, a triangulação dos dados nos revelam que fatores tais como a formação docente, a infraestrutura do curso, se este não possibilitar um ambiente adequado para o ensino e aprendizagem, a metodologia adotada pelo docente e as ferramentas que este utiliza, que pode ser causada pela falta de formação adequada ou pela falta de infraestrutura do curso. O nosso estudo nos aponta que várias são as dificuldades enfrentadas pelos alunos e professores ao longo do processo de ensino e aprendizagem de programação, o que nos permite caracterizar, de fato, este processo como complexo e desafiador para ambos, corroborando, com o que apontamos na justificativa do desenvolvimento desta tese e com a necessidade de pesquisar profundamente sobre este assunto.

A categoria 'consequências das dificuldades' nos permite reconhecer aspectos pessoais e vividos no percurso acadêmico do aluno causado pelas dificuldades enfrentadas. Os dados obtidos nos mostram que é frequente que alunos desistam ou reprovem, este com maior frequência, em UC de programação, e que esta recorrência é gerada pelas mais diversas dificuldades que apontamos anteriormente. É claro que nem todos os alunos que sentem dificuldades optam pela desistência ou trancamento da Unidade Curricular, ou chegam a ser reprovados, entretanto a desistência e a reprovação são causadas pelas dificuldades vivenciadas, corroborando com o que apresentamos na fundamentação teórica. A desmotivação é outra consequência das dificuldades, assim como o baixo rendimento nas UC. A desmotivação tanto é vista como uma dificuldade quanto como uma consequência das dificuldades. Ambos os fatores também são apontados por autores presentes na fundamentação teórica. A desmotivação pode ser ainda mais frequente que a desistência e reprovação, entretanto sua manifestação é silenciosa e difícil para o professor detectá-la em seus alunos e tentar ajuda-lo, principalmente quando se trata de um contexto em sala de aula numerosa ou heterogênea em que o docente não consegue acompanhar os alunos individualmente.

De maneira geral, podemos concluir que a triangulação dos dados nos ajudou a ter maior clareza e certeza dos aspectos que contribuem mais fortemente para as dificuldades do processo de ensino e aprendizagem e as potenciais consequências destas dificuldades. Conhecendo os aspectos, conseguimos ter uma visão de como tentar diminuir ou extinguir as dificuldades vivenciadas pelos alunos e professores para assim, evitar que as consequências estas ocorram com frequência.

4.2.6. Parâmetro 'Como'

Neste parâmetro buscamos analisar como ocorre o processo de ensino e aprendizagem de programação em três categorias: 'metodologia docente', 'metodologia discente' e 'ferramentas utilizadas'. São várias as formas de ensinar e aprender programação. Escolher a forma apropriada é um desafio para professores e alunos, os quais precisam conhecer diferentes metodologias e ferramentas, testá-las e adaptá-las de acordo com seu contexto e suas necessidades.

Como apresentado anteriormente, Jenkins (2001) aponta que a melhor maneira de aprender a programar será diferente para pessoas diferentes, mas, qualquer que seja a forma adotada de estudar, deve certamente envolver a prática, o uso de material de referência e recorrer a aulas de programação.

4.2.6.1. Metodologia Docente. Esta categoria busca conhecer as estratégias e práticas adotadas pelos docentes. Para tanto, fizemos duas afirmações:

1- A metodologia adotada pelo professor influencia no processo de aprendizagem de programação.
2- A linguagem de programação adotada pelo professor pode influenciar no processo de aprendizagem de programação.

Em relação à primeira afirmação, todos os alunos participantes da pesquisa concordam que o processo de aprendizagem de programação pode ser influenciado ou depende de como o professor ensina, corroborando com o que abordam Kumar & Khurana (2012). Esta informação é importante para nossa pesquisa uma vez que muitas das dificuldades enfrentadas pelos alunos podem ser diminuídas ou mitigadas pela abordagem adotada pelo professor, seja em relação aos conhecimentos específicos de programação, ou seja pela desmotivação do aluno. A metodologia adotada também pode favorecer para que os alunos adquiram ou melhorem suas habilidades, conhecimentos e competências, para assim, ter um maior aproveitamento durante o processo de aprendizagem.

De acordo com o aluno A08, "os professores sempre demonstraram clareza e facilidade para repassar o conteúdo, as linguagens de programação apresentadas também facilitaram bastante" (A08.1). Para o aluno A11, "o que mais me ajudou a aprender a programar foi a didática utilizada pelos professores aliada à paciência que tive para aprender com os erros" (A11.2). O aluno A26 afirma que "a abordagem do professor era interessante, felizmente não desisti de nenhuma disciplina e fui aprovado nas que paguei" (A26.1). As falas dos alunos complementam os dados obtidos com a afirmação e com o que apresentam os autores Kumar & Khurana (2012) sobre a metodologia docente.

De acordo com a análise realizada com o PPC do Curso, o documento engloba algumas estratégias para o ensino das UC de programação, tais como: interdisciplinaridade, articulação teoria

com a prática, flexibilidade, contextualização, democratização e a indissociabilidade entre o ensino, pesquisa e extensão.

Outro aspecto relevante que contribui para a articulação entre teoria e prática é o incentivo, por parte do corpo docente para que o Trabalho de Diplomação seja voltado para uma aplicação prática, na resolução de problemas reais. O PPC aborda ainda que os docentes buscam fazer a utilização de exemplos do cotidiano dos alunos, formulados a partir de dados regionais ou de conhecimento geral do corpo discente. Um aspecto metodológico importante no processo ensino-aprendizagem do Curso é a disponibilização de equipamentos suficientes para as aulas práticas, de forma a atender ao critério de um aluno por equipamento disponível. De acordo com o PPC,

> A metodologia adotada no Curso [nome do curso omitido] prima pela busca de estratégias que possam motivar o aluno, fazendo com que ele busque superar suas dificuldades como a falta de habilidades matemáticas, promovendo o combate à apatia, à baixa autoestima, à evasão e a reprovação. (PPC, 2019, p. 138)

Para o professor PROF1, a metodologia adotada dependerá da turma, mas sugere como uma estratégia adequada "fazer com que o aluno entenda através de analogias e através de justificativa do por que dele estar fazendo aquilo" (PROF1.40). De acordo com o professor, é importante explicar o 'por quê', "começando pelo básico, aí depois introduzindo as demais características de programação" (PROF1.41).

O PROF2 reflete que adotou por muito tempo a estratégia de mostrar exemplos de códigos prontos para que os alunos repetissem, mas notou que a com a estratégia o aluno "começa a fazer as coisas, até consegue fazer funcionar um determinado código, mas que com se você muda alguma coisa, pede algo diferente, o aluno não consegue pensar para abstrair e programar" (PROF2.73), ou seja, o aluno apenas replicava o código sem de fato aprender nem saber o que estava fazendo. Ao perceber isso, o professor afirma que "de um certo tempo para cá, o que é que eu tenho feito: eu tenho mostrado a lógica do que eu quero, o que é que eu quero, ou seja, o que é o problema, e como resolvê-lo" (PROF2.74), neste sentido o professor complementa dizendo que

> Tento mostrar várias formas diferentes de resolver um mesmo problema, aí vou mostrando níveis de programação pra resolver aquele problema, o mais trivial, mesmo que seja o mais forçado para o computador, em termos de processamento, aí a gente vai resolvendo. Aí eu começo com problemas bem simples, os triviais. (PROF2.75)

Com essa estratégia, o professor vai aumentando o nível de dificuldade, de acordo com a reação da turma, dando o tempo necessário para que os alunos sejam autores do conhecimento e não

meros repetidores do que o professor apresenta em sala de aula. No uso de suas estratégias, o professor PROF2, assim como o PROF1 frisa que "depende muito da turma, depende do nível da turma" (PRO2.81). O professor corrobora com Jenkins (2001), ao falar que

> De maneira geral não tem muita receita pronta não, porque às vezes você pega um grupo diferente e aí você tem que criar na hora, às vezes como professor você tem um *insight* em uma aula de fazer uma coisa que ajude, né, e que nunca fez antes, mas é assim mesmo. (PROF2.87)

O professor PROF3 cita que estimula os alunos com exercícios para eles "realmente fazerem exemplos, fazerem algoritmos, não copiarem os meus algoritmos, tentarem fazer os deles" (PROF3.47). E complementa que não aplica diferentes estratégias porque percebe que a que ele usa tem efeito, entretanto busca "ser bem flexível com respeito a esse ensino, no decorrer do período, por que depende muito da turma, do conhecimento da turma, do tamanho da turma, do envolvimento deles" (PROF3.48) e modifica a estratégia de acordo com a reação da turma.

O professor PROF4, por sua vez, acredita que "para ensinar programação, existem estratégias diferentes de acordo com o passar da disciplina" (PROF4.76). Ele cita que "normalmente nas aulas iniciais, nas aulas mais introdutórias, eu costumo fazer pequenos exemplos desconectados um do outro, então são exemplos sobre várias coisas diferentes. Então cada assunto, vários exemplos" (PROF4.77). Com os exemplos aplicados, o professor permite que os alunos tenham uma maior vivência da prática de resolução de problemas. O PROF4 continua sua fala dizendo que quando os alunos adquirem um conhecimento maior, volta para os exemplos anteriores, explicando novos conceitos e mostrando diferentes técnicas de resolver o mesmo problema, mas melhorando a forma de programar, mostrando as vantagens de se aprender e aplicar novos conceitos. Outra estratégia utilizada pelo PROF4 é de desenvolver projetos na UC. Segundo ele, "toda disciplina tem o projeto final, [...] o objetivo sempre é ir fazendo, construindo ele aos poucos, por etapas, então em cada avaliação a gente tem uma evolução, então há um certo ciclo de desenvolvimento evolutivo" (PROF4.77). Com os projetos, os alunos aprendem de forma iterativa a programar, além de ter a possibilidade de resolver problemas reais, considerada uma prática importante para a aprendizagem de programação (Al-Imamy et al., 2006; Mendes et al., 2012; Rezende & Bispo, 2018). O professor PROF4 relata ainda outra metodologia que adota:

> as minhas disciplinas elas são muito práticas então eu gosto sempre de colocar exercícios, fazer normalmente mesclando aulas teóricas com aulas práticas, então a gente vê um conteúdo e na outra aula a gente faz exemplos e exercícios, né, e ultimamente também uma

> construção digamos assim mais colaborativa, então ao invés de eu mostrar o exemplo e resolver, [...] eu chamo cada aluno que estava no laboratório para fazer uma parte da solução. (PROF4.79)

E complementa que "o aluno fazer o programa de maneira participativa é boa porque eu também incentivo os alunos a contribuírem com os colegas [...] que aí quando ele não sabe eles ajudam dizendo o que é que tem que fazer" (PROF4.80). A estratégia do PROF4 auxilia os alunos a trabalhar em equipe e de forma colaborativa, habilidades estas importantes para a área de Informática, como aborda Silva et al. (2015). Outras estratégias adotadas pelo professor é incentivar os alunos a participar de competições de programação e o uso de robôs virtuais (PROF4.81). O professor PROF5 fala que a metodologia adotada se baseia especialmente na resolução de problemas, definindo um problema a ser abordado e instigando os alunos para construir soluções para resolver o problema (PROF5.54).

Os alunos entrevistados apontaram situações positivas e negativas em relação à metodologia docente. Do ponto de vista positivo, temos as vozes dos alunos ALU2, ALU3 e ALU5:

> Por exemplo, um algoritmo de automatização, o professor [nome do professor omitido] mostrou também algoritmos de sorteio, *Bubble Sort*, *Quick Sort*, esses algoritmos já bem antigos, mas assim, que eu achei melhor que só ficar naquelas questões do URI, toda vida aquele negócio. (ALU2.40)

O aluno apresenta uma visão positiva sobre o professor apresentar outros exemplos e usar outras estratégias, mesmo com algoritmos considerados antigos pelo aluno, em detrimento de usar apenas o URI que, embora seja uma plataforma dinâmica e atrativa, para o aluno não é interessante, por considerar o uso repetitivo ao mencionar "toda vida aquele negócio".

> Mas os professores tanto de Algoritmos quanto de Programação Estruturada trabalhavam na prática, por que davam o conteúdo, mas levavam a gente na prática, e mostravam como fazer, vendo a disciplina na prática mesmo aí foi que me ajudou muito. (ALU3.47)
>
> Pronto, uma coisa que o professor [nome do professor omitido] falou é que programação é você, na prática, você aprende na prática, vai tentando. Aí ele passava uns exemplos na sala e deixava o resto para você fazer em casa, né, pra tentar. E sempre colocava exercícios na plataforma para a gente ir tentando, e incentivava não pegar só os exemplos que tinha na plataforma, mas pesquisar mais por que como hoje tudo é mais fácil, né? (ALU3.49)

As falas acima têm em comum a palavra 'prática', repetida diversas vezes pelo ALU3. O mesmo aluno cita ainda o incentivo do professor e a inclusão de exercícios para o processo de

aprendizagem, bem como orientar os alunos a buscarem novos desafios e tarefas a serem realizadas. Na voz do aluno, é possível sentir que esta metodologia o traz mais confiança e o leva a maiores desafios, que o permitem aprender mais.

> Todo início de aula ele [o professor] dá uma revisada no que ele falou antes, ele pergunta, ele vai dando dose, ele dá um assunto e vai perguntando a turma se realmente entendeu, se é isso mesmo, eu acho muito interessante o modo dele dar aula. (ALU3.52)

A fala do aluno ALU3 apresenta outra estratégia adotada pelo professor, que é a da revisão. Esta estratégia é abordada por A. Santos et al. (2011), e com ela os alunos têm a oportunidade de remeter aos conteúdos vistos anteriormente, tirando dúvidas e relembrando o assunto dado, conseguindo amadurecer as ideias e avançar. Além da revisão, o aluno cita ainda que o professor avança de forma gradual os conteúdos dados, o que pode ser interessante para o contexto das turmas de programação, que geralmente possuem alunos dos mais diferentes níveis de conhecimentos e habilidades (Oliveira et al., 2015). O aluno ALU5 afirma que "eles são ótimos professores, são bem didáticos, foram professores excelentes para mim, e a forma como eles ensinaram, como eles traziam novos assuntos, novas coisas de programação, acho que isso foi bem interessante" (ALU5.38) e complementa:

> Teve uma matéria, que acho que foi Estrutura de Dados e Programação Estruturada, os professores estavam usando a mesma linguagem, que era linguagem C, e basicamente um complementava o assunto do outro, então acho que isso foi bem enriquecedor para mim, por que cada um tem sua forma de ensinar, né? Mas acho que os dois juntos ensinando numa mesma linguagem, e mostrando coisas diferentes de como a gente podia usar, eu acho que isso foi muito útil. [...] Tem uns professores que eles gostam de trazer curiosidades, dizer por que você está estudando aquela matéria, qual o intuito, onde você pode usar aquilo... associando a teoria à prática, eu acho que isso para mim faz muita diferença (ALU5.37)

O ALU5 apresenta sua visão em relação às disciplinas de Estrutura de Dados e Programação Estruturada que são ministradas no mesmo período, mas geralmente por professores diferentes. No Curso, essas duas disciplinas são consideradas complementares e, para facilitar o processo de aprendizagem, geralmente são ensinadas utilizando a mesma linguagem de programação. O ALU5 reconhece esta estratégia como importante para o processo de aprendizagem, pois embora sejam UC do início do Curso, são complexas e consideradas difíceis por abordar assuntos abstratos que podem dificultar a compreensão dos alunos. O aluno cita ainda que ao associar o conteúdo com a realidade o

professor pôde trazer uma experiência diferente e enriquecedora para a sua aprendizagem em programação.

Ao contrário das falas apresentadas, existem situações em que o aluno discorda da metodologia do professor, a qual pode ser responsável pelo insucesso do aluno. Dois dos alunos entrevistados apresentaram insatisfação em relação à metodologia dos professores de programação. O aluno ALU1 cita que estava achando a aula do professor muito teórica, com muitas projeções de *slides*, o que o desmotivou e o fez desistir de continuar na disciplina. Para o aluno ALU5, "têm matérias que você entra assim, não tem muito ânimo do professor e você não encontra funcionalidades, digamos assim, naquilo que ele está falando, você não consegue ver onde vai aplicar, e a matéria acaba ficando chata, ficando monótona" (ALU5.41). Vejamos que ambos os alunos reconhecem a necessidade da prática para que a aprendizagem seja eficaz corroborando com o pensamento de Freire (1987).

Aliado à metodologia, o recurso de apoio adotado pelo docente é importante para orientar professores e alunos no processo de ensino e aprendizagem, respectivamente. Para apresentar este aspecto, apresentamos trechos das falas dos alunos em que eles citam os materiais comumente utilizados pelos docentes, e sua visão a respeito disso.

> E assim, em POO e em avançada, essas demais, essas disciplinas mais avançadas de programação, é muitos *slides* que os professores passam são, e só focado só naqueles *slides* você não vai conseguir muita coisa. Por que [breve pausa] muitas vezes as provas nem são baseadas nos *slides*... Eu paguei POO com professor [nome do professor omitido], e os *slides* de professor [nome do professor omitido] eles são, tipo, 50% do que ele fala e do que ele pede para a gente fazer. (ALU1.53-54)

De acordo com o aluno ALU1, os professores utilizam *slides* em suas aulas, entretanto, não são suficientes para a aprendizagem do aluno. Em sua fala, o aluno demonstra descontentamento no fato de o professor apresentar um material que muitas vezes para o aluno não servirá de apoio para as provas e alerta para o fato de que os alunos não devem estudar apenas pelos *slides*, pois não conseguirão êxito na UC.

Em relação à segunda afirmação, a intenção é verificar a percepção dos alunos sobre a linguagem de programação adotada pelo professor para ensinar a programar. Os resultados obtidos mostram que 81,2% dos alunos acreditam que a linguagem de programação adotada pelo professor influencia no processo de aprendizagem de programação. Esta afirmação é trazida pelos autores Aparicio & Costa (2018), Bosse & Gerosa (2017), A. Gomes & Mendes (2015) e Piteira et al. (2017), que apontam que pode existir essa correlação entre o processo de aprendizagem e a linguagem de

programação adotada. De acordo com Skalka e Drlík (2018) um dos domínios de conhecimento e habilidade necessários para a aprendizagem de programação é o domínio da linguagem de programação.

Para o aluno A05, "a 2ª vez [que cursou a UC] foi com um outro professor que abordou outra linguagem [de programação] nela aprendi muito e comecei a desenvolver raciocínio lógico nas questões" (A05.3). Apesar de, em nosso estudo, não estarmos focando no ensino e aprendizagem de uma linguagem de programação específica, é importante apresentar que, de fato, as linguagens de programação podem dificultar ou auxiliar o processo, dependendo da abordagem usada pelo professor, do assunto ensinado e da complexidade e profundidade dos assuntos.

A estratégia adotada pelo PROF3 permite que os alunos façam os trabalhos e avaliações da UC na linguagem de programação que quiserem. Para o ALU3, o professor apesar de adotar uma linguagem específica para o ensino de programação, permite que o aluno adote uma linguagem que tenha preferência, permitindo uma experiência flexível no processo de aprendizagem. Em programação, a flexibilidade é importante, pois o ensino não fica vinculado à sintaxe e semântica de uma linguagem, mas à lógica da resolução de problemas e ao desenvolvimento de algoritmos por meio da abstração. Com esta atitude, o professor vai com o pensamento contrário ao de Souleiman (2018), cujo trabalho aponta que o ensino de programação nos cursos introdutórios geralmente focam principalmente no ensino na sintaxe e semântica de uma linguagem de programação, deixando de fora métodos de solução de problema. Ao contrário, o professor citado pelo aluno ALU3, parece focar na resolução de problemas e na lógica, e não na linguagem de programação, a qual passa a ser o meio, e não o fim.

> O professor [nome do professor omitido] deu algoritmos em Python, mas ele dizia assim: "a gente vai dar em Python aqui porque [...] tem uma facilidade maior para o aluno, mas a língua que você escolher [...] eu lhe ajudo do mesmo jeito, não tem problema. Essa aqui eu estou escolhendo essa, mas não vamos ficar preso a uma linguagem só. (ALU3.50)

Para o aluno ALU4, a simplicidade da linguagem Python, adotada pelo professor o ajudou a compreender os conceitos de programação. Por ser uma linguagem de mais fácil compreensão, o aluno afirma não ter que se preocupar tanto com a estrutura da linguagem em si, como ocorre com a linguagem C, por exemplo, e focar na resolução de problemas. Nesse sentido, o aluno concorda com a linguagem adotada pelo docente e se sente motivado e mais animado para aprender a programar. Ele se vê livre de paradigmas e de estruturas complexas, o que torna o processo de aprendizagem mais leve e proveitoso. De acordo com o aluno,

> O que também foi importante, [...] foi com professor [nome do professor omitido], com algoritmos, com Python. [...] Que ajuda mais a ter uma coisa mais construtiva, mais para entender mesmo como é que é a prática daquilo e me ajudou muito por que era uma linguagem mais simples até certo ponto, por que não tem tanto aquele negócio de C, de ponto e vírgula, *int*, *long*... era uma coisa mais dinâmica que ajudou a entender o que é um processo, o que é um *for*, o que é um *while*, essas coisas, como estruturar (ALU4.41-42)
>
> Aí pelo menos com Python, eu também gostei por que meio que me ajudou a pensar mais também na lógica mais, e não ligar muito, ligar no sentido daquela coisa mais baixo nível, como em C, colocar um ponteiro, e a parte de memória, você não liga muito para isso, e eu pelo menos gostei de como foi feito por que meio que antes a pessoa pensa primeiro na lógica em termos de Python, já que é dinâmico, ele faz isso só, para depois ter conhecimento mais de C, em estrutura de dados, essas coisas, para entender como é que funciona aquilo, tipo "por baixo dos panos" (ALU4.44)

4.2.6.2. Metodologia Discente. Na categoria 'metodologia discente' apresentamos duas afirmações:

1- A metodologia usada para estudar influencia no processo de aprendizagem de programação.
2- O tempo que dedico para estudar influencia no processo de aprendizagem de programação.

A primeira afirmação tem por objetivo conhecer a percepção do aluno sobre sua forma de estudar e se esta pode afetar o seu aprendizado, como é apontado por Biggs & Tang (2011), Clark et al. (2011) Chuchulashvili et al. (2016), Eltegani & Butgereit (2015) e A. Gomes & Mendes (2015). Os resultados mostram que cerca de 94% dos alunos concordam que a forma como estudam influencia no processo de aprendizagem, embora a grande maioria (62,5%) concorde apenas em parte. Apenas dois alunos discordaram da afirmação apresentada.

Na opinião dos professores, a palavra-chave para estudar programação é 'prática'. A fala do PROF1 nos diz que para aprender programação é preciso "[...] pegar os exemplos mais básicos, e começar a implementar. Não tem pra quê ficar decorando comando nem nada não. Implementa com o livro do lado, com o livro do lado ou com a referência do lado também" (PROF1.34). Os professores PROF3 e PROF4 corroboram com o PROF1, quando opinam que

> Algoritmos basicamente é dedicação. Se eles chegarem, se dedicarem àquilo e trabalharem àquilo [...]. Se eles treinarem algoritmos, se eles tiverem um tempo pra se dedicar diariamente ou três vezes por semana, a dizer assim: 'eu vou sentar aqui uma hora pra desenvolver

> problemas que eu não sei solucionar' aí entender o problema e solucionar, entender o problema e solucionar, eles conseguem desenvolver isso aí. (PROF3.40)
>
> se você não procurar resolver mais problemas, procurar mais materiais, até porque a aula é num tempo curto, então o professor não consegue passar tudo, né, de todas as formas, todos os exemplos, então você tem que procurar por fora, você tem que aprender uma linguagem, daqui a pouco tem outra nova, então você tem que procurar aprender essa linguagem nova baseado no que você já tem. (PROF4.69)

Os professores são enfáticos em relação à prática e à buscar resolver exercícios além dos apresentados em sala de aula. O PROF4 ainda cita o fator carga-horária das UC de programação que por ser pouco tempo, não tem como ver todo conteúdo apenas em sala de aula.

Os alunos, ao longo de suas respostas sobre como estudam, abordaram diferentes aspectos. Para o aluno ALU1, depende do conteúdo estudado, de forma que de acordo com a complexidade do assunto, o estudo pode ser baseado mais na teoria ou na prática, pode exigir mais esforço do aluno, ou a necessidade deste se reunir com outros colegas para tirar dúvidas e realizar as tarefas. Outra forma de estudar, de acordo com os alunos é por meio da revisão do conteúdo visto em sala. Para o ALU1, "antes de começar o próprio semestre para pagar essas matérias a gente tem que dar uma revisada pelo menos o básico já que sabe que vai ser uma linguagem de programação que a gente nunca usou" (ALU1.34). Já o aluno ALU6, adota a estratégia de revisão após as aulas. Ele afirma que presta bastante atenção no que é apresentado em sala pelo professor, para posteriormente estudar revisando o conteúdo.

Assim como abordado pelos professores, os alunos apontam que praticar os conceitos de programação é essencial para o processo de aprendizagem, uma vez que programar é considerada uma habilidade que se adquire praticando. Para o aluno ALU1, o que mais o ajudou em POO e Programação Avançada foi praticar fora das aulas além do que era passado pelo professor (ALU1.33). Para os alunos ALU4, ALU5 e ALU6,

> Às vezes quando o professor disponibiliza algum tipo de problema ou algum código feito eu tento entender como é que é feito aquela estrutura e tentar resolver outros problemas. Essas coisas ajudam um pouco a ter essa lógica. Tipo, praticar para desenvolver melhor, então geralmente eu faço isso: resolver o que o professor passa, pesquisar fora. [...] Tenta fazer, tipo praticar, eu tento sempre melhorar, tentar praticar, pelo menos entender um pouco daquilo que o professor passa, pelo menos nas que já passaram. (ALU4.32-33;35)

> Eu particularmente acho que os exercícios me ajudam muito. [...] Vão dos níveis fáceis até os mais difíceis e a gente vai fazendo conforme a gente vai desenvolvendo. [...] Meu aprendizado é dessa forma: teoria, exercício e eu vou praticando. (ALU5.29-31)
>
> Aí eu vou praticar da minha maneira, vamos supor, vou fazendo uns projetinhos mesmo individuais, vamos ver se eu acerto um número aleatório que a máquina gerou... aí eu fico treinando desse jeito e funciona pelo menos para mim, colocando a mão na prática, né? (ALU6.14)

A prática dos conteúdos, à qual os alunos se referem, vão desde praticar ou treinar exercícios passados pelos professores a problemas que eles encontram fora da sala de aula. Podemos perceber que os alunos reconhecem que precisam realizar atividades práticas para aprender a programar de forma efetiva e não se conformam apenas em realizar aquilo que é apresentado pelo professor, demonstrando assim autonomia e proatividade, competências importantes para a aprendizagem de programação (Chuchulashvili et al., 2016; Mendes et al., 2012).

Outra estratégia adotada pelos alunos é a busca de recursos adicionais, o que também demonstra a autonomia e proatividade dos alunos. Ao buscar outros recursos, para além dos apresentados e indicados pelo professor nas aulas, o aluno pode aumentar as possibilidades de consulta, bem como conhecer materiais e ferramentas que melhor se adaptem ao seu contexto.

Em alguns momentos, os alunos veem a necessidade de realizar o estudo individual e em outros o estudo em grupo. De acordo com os alunos, a escolha pelo método depende da UC e das dificuldades enfrentadas durante a aprendizagem. O aluno ALU2, é o único entrevistado que afirma preferir estudar sozinho, independente da matéria, mas reconhece que é um aspecto que precisa melhorar. Já a opinião do aluno ALU1 é diferente:

> Eu acho muito difícil estudar sozinho, principalmente por que como eu falei quando você chega nessas disciplinas de POO e [Programação] Avançada, você vê muita coisa nova que você nunca viu, e toda aquela base que você teve em Construção de Algoritmos e Estrutura de Dados ela meio que como o próprio nome diz ela é uma base, você fica perdido basicamente, quando você começa as primeiras aulas de Avançada e POO e acho que são essas duas matérias que eu mais me aliei a grupos. (ALU1.37)
>
> Mas em relação à Construção de Algoritmos e Estrutura de Dados, foi mais sozinho mesmo. Só revisando os *slides*, e tal, como eu falei essas duas matérias elas são mais voltadas a teóricas, a colocar na sua cabeça como funciona a base de tudo. (ALU1.38)

O aluno ALU4 corrobora com a opinião do aluno ALU1, ao afirmar que a escolha por estudar individualmente ou em grupo depende do que irá estudar:

> Assim, acho que tem que ter um pouquinho dos dois, porque ajuda tipo, grupo eu gosto, por exemplo, eu não tenho alguma coisa, está faltando, tipo eu não tenho aquele conhecimento, e alguém sabe, eu tento perguntar 'ei, você já fez isso?' ou alguma coisa assim, e a pessoa tipo dá a experiência dela, então dá uma ajuda, tipo, digamos assim, a ter um conhecimento a mais. Mas também eu acho que a prática individual ajuda por que às vezes também você tem vergonha, não sabe como perguntar, ou alguma coisa do tipo, e isso ajuda ah, 'como fazer isso?'. Parece ser besta, mas é necessário. Então eu acho que um pouquinho dos dois é bom, um meio termo. (ALU4.34)

Na opinião do ALU3, ele prefere estudar em grupo, entretanto nem sempre é possível devido aos seus horários de trabalho. Ele fala de duas situações: a primeira, uma tentativa de estudar sozinho, com recursos adicionais, e a segunda, quando se reúne com os colegas para estudar:

> Por mais que eu, que eu trabalho a noite e tenho uns horários livres, por mais que eu assistisse videoaula, eu ficava assistindo videoaula, mas não era do mesmo jeito de que ter uma pessoa explicando, às vezes a noite você está assistindo uma videoaula, a gente não entende uma coisa, volta, volta, volta, mas não consegue. [...] Como o período que eu tinha mais tempo era a noite, então cedo da noite eu me juntava com os colegas. (ALU3.42-43)

No caso do ALU3, o seu contexto de aprendizagem nem sempre permite que este estude em grupo. Entretanto, ele apresenta insatisfação quando estuda sozinho, uma vez que não tem a quem recorrer ou com quem tirar dúvidas, quando estas surgem. O aluno demonstra insegurança em estudar sozinho e a necessidade da convivência com os colegas de sala para sua aprendizagem.

Uma forma eficiente encontrada pelo ALU2 para aprender programação é estudar fazendo a associação do conteúdo ao contexto real.

> Bom, na disciplina de algoritmos eu tento pegar o que o professor passa em sala, e procurar algum problema real assim para tentar resolver. Por exemplo, Python: que foi no meu caso com algoritmos, Python ele é muito bom para automação, então, por exemplo, fazer um algoritmo que automatiza alguma coisa para o seu dia, [...] então tentar colocar em prática mesmo aquilo que o professor passa. (ALU2.33)

De acordo com o aluno ALU2, ele busca aplicar os conhecimentos adquiridos para resolver problemas simples do cotidiano. Com esta visão, o aluno sai da sua zona de conforto e ultrapassa o

que é dado em sala de aula, para aprender conceitos novos a partir da aplicação prática, consolidando sua aprendizagem.

Para os alunos ALU4 e ALU6 outra forma de estudar programação é com a participação nas aulas, prestando atenção nos conceitos dados pelo professor, tentando entender como o professor ensina os assuntos. Os alunos consideram o papel do professor e das aulas importantes para o processo de aprendizagem e acreditam que a participação de forma atenta às aulas pode melhorar a sua aprendizagem e o seu desempenho.

O aluno ALU5 considera sua forma de estudar como um método próprio, diferente do adotado pela maioria dos alunos da área de Informática:

> Quando eu começo a estudar alguma matéria eu gosto muito de fazer resumos, durante a aula ou durante os momentos que o professor está lá ensinando, e eu gosto muito de fazer os meus próprios resumos, e isso me ajuda bastante também, é uma metodologia que serve muito para mim, [...] o meu método de aprender mesmo, é escrevendo é lendo, é fazendo os meus resumos, então acho que os meus resumos, aliado com várias práticas de exercício acho que isso me ajuda muito, a fixar os conteúdos e a fixar a matéria de uma melhor forma. (ALU5.29)

Ele demonstra segurança na sua forma de estudar e acredita que seu método é o mais eficaz para a sua aprendizagem. Podemos ver que ele, ao fazer resumos, não apenas reproduz o que o professor está ensinado, mas produz seu próprio material de apoio.

Sobre a segunda afirmação, 87,5% dos alunos concordam que o tempo dedicado para estudar influencia na aprendizagem de programação. A dedicação do tempo é um fator apresentado por diferentes autores (Giraffa & Mora, 2013; Medeiros, 2019; Moreira et al., 2018, Rezende & Bispo, 2018; A. Santos et al., 2011). Jenkins (2001) afirma que uma das estratégias para aprender a programar é a prática constante que, por sua vez, demanda tempo e disponibilidade do aluno. Para o aluno A26, "A leitura e prática são essenciais para o aprendizado" (A26.2), o aluno 29 assinala que "só com a prática diária é possível compreender [os assuntos]" (A29.3). Para o professor PROF4,

> só não se desenvolve, digamos assim, se não quiser. Então é interessante, e a questão do tempo, né? [...] se a pessoa tiver tempo para praticar, tempo pra estudar, é uma coisa muito boa, porque, assim é um, normalmente programação exige dedicação, né? Exige um tempo maior pra você pensar, pra você implementar, testar e tudo. (PROF4.13)

As falas do aluno e do professor corroboram com o que trazem os autores apresentados na fundamentação teórica. O tempo disponível permite que o aluno possa ter disponibilidade para estudar e praticar e, assim, ter mais chance de sucesso no processo de aprendizagem.

4.2.6.3. Ferramentas Utilizadas. Para esta categoria fizemos a seguinte afirmação*:*

1- As ferramentas adotadas pelo professor influenciam no processo de ensino e aprendizagem de programação.

Essa afirmação aborda sobre as ferramentas adotadas pelo docente no processo de ensino de programação. De acordo com os dados obtidos com os questionários, a percepção dos alunos está equilibrada entre os que concordam e discordam que a ferramenta usada pelo professor pode influenciar no processo de aprendizagem. De acordo com alguns autores apresentados na fundamentação teórica desta tese, os professores adotam uma ferramenta específica mas deixam os alunos 'a vontade' para escolher a mais adequada (Gomes & Mendes, 2007; Raabe & Silva, 2005; Gomes, 2010). Por este motivo, podemos ter obtido o resultado em que 53,1% dos alunos concordam, entretanto 34,3% discordam em parte e 12,5% discordam totalmente com esta afirmação.

No tocante às falas dos professores, o professor PROF1 aponta que ele utiliza o site do C++ *references*[7] como instrumento de apoio para o estudo da linguagem C++, utiliza a IDE Visual Studio[8] e a ferramenta CPlex[9], especialmente para programação de metaheurística. O PROF2 usa o compilador GCC e revela que não possui preferência por nenhuma IDE, mas recomenda algumas aos alunos durante as aulas. Assim como o professor PROF1, o PROF2 usa o Visual Studio e o CPlex. O professor fala que "como eu permito que eles usem qualquer linguagem de programação, eu também permito a utilização de qualquer ferramenta" (PROF2.41), deixando livre para que o aluno possa escolher à que melhor se adapte.

O professor PROF3 usa a ferramenta Dev C++, que possui versão para os sistemas operacionais Windows e Linux, mas não limita a estas, e fala que os alunos usam ferramentas *on-line* enquanto outros preferem o Eclipse[10]. Assim como os demais professores, o PROF4, em relação às IDE, geralmente deixa em aberto para os alunos escolherem. Como ferramenta de ensino, o professor usa o Juiz online URI[11] que, para ele, "é muito grande, tem muitos problemas, aceita muitas linguagens diferentes, e eles sempre agregam problemas de várias competições, né, competições que existem, maratonas de programação" (PROF4.71).

Sobre a IDE, o professor fala que adota algumas como o Visual Studio, Eclipse e o NetBeans[12] mas não obriga que seus alunos as usem, deixando geralmente em aberto. Entretanto, o professor afirma que

[7] www.cplusplus.com/references
[8] https://visualstudio.microsoft.com/pt-br/downloads/
[9] https://www.ibm.com/br-pt/products/ilog-cplex-optimization-studio
[10] *https://www.eclipse.org*
[11] https://www.beecrowd.com.br/judge/en/login/?origem=1
[12] https://netbeans.apache.org

> Muitas vezes eu faço sem IDE, eu pego o código no editor de texto simples, compilo com uma linha de comando, por que se você usa a IDE, a IDE ele meio que esconde o que está acontecendo né por trás, como é que terá a compilação, quais são os arquivos envolvidos, etc, e algumas vezes é importante o aluno, principalmente o aluno de nível superior de Computação entender o que é que está acontecendo por trás das cenas né, e o IDE ele costuma esconder muito do processo. (PROF4.72)

Assim como o PROF4, o professor PROF5 usa o URI e editores de texto para compilar o código. Segundo ele,

> No que diz respeito a ferramentas, geralmente eu fazia uma abordagem um pouco mais ortodoxa eu foquei muito em trabalhar a ideia de que você tem para programar um editor de texto, você tem o ambiente de compilação, e você tem o ambiente de depuração, desacoplar isso. E eu tenho feito esse tipo de abordagem pra tentar distanciar o conceito de que você precisa ter o IDE para programar. (PROF5.48)

De acordo com as falas dos alunos, percebe-se que as ferramentas adotadas vão depender do conteúdo ministrado, da linguagem de programação, do que é disponibilizado nos laboratórios e da indicação do professor. Para o aluno ALU1,

> No começo, acho que nas quatro [UC] de programação a gente usava [as ferramentas], eu usava mais o indicado pelos professores. Por que tem aquela questão principalmente em POO e [Programação] Avançada que as provas são no computador lá da [nome da Instituição omitida] né? (ALU1.40)

De acordo com o aluno ALU3, o professor de Construção de Algoritmos "deixava a gente livre, dizia, 'pronto, tem o Visual Studio, tem o NetBeans...' para as ferramentas e ele dizia assim que podia usar a linguagem, estava livre" (ALU3.45). De acordo com o aluno ALU4,

> Assim, no início eu não tinha muito meu pessoal, eu não tinha muito essa desenvoltura por que eu não tinha nenhuma opinião formada, aí normalmente o que eu usava era tipo o PyCharm era tudo o que o professor realmente usava por que eu já estava meio que familiarizado com aquilo no Curso [...] depende muito do que o professor passa. (ALU4.38)
>
> Como a gente é de computação o professor indicou que a gente usasse o terminal mesmo do Python para a gente não ficar muito apegado às IDEs, para a gente usar realmente um terminal, saber como utilizar um terminal na verdade. (ALU5.34)
>
> Eu sempre pergunto para o professor qual ferramenta ele acha melhor, e eu testo essa ferramenta e se eu não conseguir configurar ela do jeito que eu gosto de programar, eu tento

> outra, então, acho que o PyCharm o professor [nome do professor omitido] me indicou e eu gostei. (ALU6.16)

Notamos que os alunos seguem geralmente as ferramentas indicadas pelos professores, principalmente no início do Curso, quando ainda não têm um maior conhecimento de programação nem das ferramentas disponíveis. Ao passo que os alunos vão ganhando experiência em programação, adquirem competência para escolher as ferramentas mais apropriadas para cada situação.

Sobre as ferramentas de aprendizagem de programação, as falas dos alunos remetem ao URI, Scratch[13] e Udemy[14]. Os recortes abaixo apresentam as falas dos alunos que nos permitiram inventariar as ferramentas usadas na aprendizagem de programação: "a gente usou [o URI] somente nas duas primeiras, Estrutura de dados e tal. Achei pra disciplina um site muito bom" (ALU1.39), "por que assim, o contato que a gente tinha com resolução de problemas mais nessas matérias, pelo menos no meu caso, era o URI, aquele de juízes online de questões" (ALU2.36), "o professor [nome do professor omitido] passava o URI" (ALU4.36), "eu acho muito útil, foi muito legal eu não conhecia essa plataforma [URI], mas é uma forma muito legal e eu acho que ajuda bastante" (ALU5.32), "Eu entrei no Curso teve aquele negócio com o Scratch" (ALU4.37), "cursos também que eu faço, que tem muito exercícios, né? Cursos fora da graduação, eu gosto muito da Udemy, aquela plataforma de cursos online, que são bem baratos os cursos, alguns gratuitos, para estudar melhor para a prova" (ALU6.20). O aluno ALU2 tem uma perspectiva diferente dos demais colegas sobre o URI, a opinião dele é que

> E eu acho que ele é muito importante sim, para o aluno ter uma noção de [breve pausa] fazer um algoritmo, resolver um problema, mas eu acho que depois de um tempo começa a ficar maçante, e aquilo, ele é bem acadêmico, não é algo que você vai usar no seu dia a dia. (ALU2.37)

O uso do URI é feito principalmente nas UC de Construção de Algoritmos e Estrutura de Dados, nos primeiros semestres do Curso. A ferramenta permite que os alunos desenvolvam a prática da resolução de problemas usando alguma linguagem de programação. Para os alunos ALU1 e ALU5, a experiência com o URI foi considerada útil para a aprendizagem. A opinião do ALU2 é que o URI é útil para a resolução de problemas, entretanto sua utilidade é muito acadêmica, de forma que não é interessante para usar no dia a dia. Percebe a falta de motivação do aluno em usar a ferramenta, ao achá-la maçante, principalmente com o passar do tempo, quando o aluno começa a adquirir mais conhecimento.

[13] https://scratch.mit.edu/
[14] https://www.udemy.com/

O uso do Scratch é citado apenas pelo ALU4, que afirma ter usado no início do Curso. O uso do Scratch é defendido na pesquisa realizada por Rezende e Bispo Jr. (2018), que citam que utilizar ferramentas que minimizam os detalhes das linguagens de programação, são capazes de facilitar o processo de aprendizagem de programação e assim "com foco nos algoritmos, o aluno pode expressar a sua criatividade em busca da solução do problema, priorizando o desenvolvimento da lógica, em vez de se preocupar com regras de sintaxe da linguagem de programação" (p. 492).

A utilização da plataforma Udemy é citada pelo aluno ALU6. O uso de uma plataforma fora do contexto acadêmico, como feito pelo ALU6, demonstra que ele busca outras formas de aprender. Ele mostra preocupação em realizar mais exercícios práticos, especialmente para se preparar para a prova, com o intuito de ter melhor avaliação acadêmica. Enquanto os alunos ALU1, ALU2, ALU4 e ALU5 afirmam usar ferramentas de apoio a aprendizagem indicadas pelo professor, o ALU6 apresenta competências de autonomia, proatividade e iniciativa para buscar ferramentas fora do contexto acadêmico, que podem ser úteis para a sua aprendizagem.

Outras ferramentas que são comumente ou até mais utilizadas que as ferramentas de aprendizagem são as IDE. Essas ferramentas se confundem com as ferramentas de aprendizagem porque, embora sejam utilizadas no dia a dia do programador, no ambiente real de seu trabalho, são usadas para a aplicação, na prática, dos conteúdos vistos no Curso. As vozes dos alunos apontam para o uso das IDE Eclipse (ALU1, ALU2, ALU4, ALU6), NetBeans (ALU1, ALU2, ALU3, ALU4, ALU5), Dev (ALU1, ALU5), Visual Studio (ALU2, ALU3, ALU4), citadas pelos professores entrevistados, e ainda as IDE Geany[15] (ALU1), PyCharm[16] (ALU3, ALU4, ALU5, ALU6), Android Studio[17] (ALU4), Falcon[18] (ALU5), Atom[19] (ALU2, ALU6), IntelliJ[20] (ALU6) e o editor de texto Sublime Text [21](ALU1). Além das IDEs, o aluno ALU6 aborda o uso de repositório de códigos como ferramenta para apoio ao processo de aprendizagem de programação, citando como exemplo o GitHub e o GitLab (ALU6.2).

Quando os alunos falam sobre as ferramentas que usam ao longo do Curso, é perceptível a maturidade que eles apresentam ao citar quando as usam e o propósito de escolhê-las. Se por um lado, no início as ferramentas usadas por eles eram sempre indicadas pelos docentes, com o passar dos períodos, e com as aprendizagens adquiridas, os alunos conseguem por eles próprios, escolher, indicar e adaptar as ferramentas de acordo com suas necessidades. Essa conhecimento é importante

[15] https://www.geany.org/
[16] https://www.jetbrains.com/pt-br/pycharm/download/
[17] https://developer.android.com/
[18] https://falcon.readthedocs.io/en/stable/
[19] https://atom.io/
[20] https://www.jetbrains.com/pt-br/idea/download/#section=windows
[21] https://www.sublimetext.com/

para o desenvolvimento de habilidades e competências do aluno para o aprendizado de programação, como citado por Souleiman (2017).

4.2.6.4. Conclusão. A triangulação dos dados realizada no parâmetro 'Como', nos permite 'inventariar metodologias e ferramentas usadas para o ensino e aprendizagem de programação' bem como nos ajuda a 'caracterizar os fatores que podem influenciar para o aluno aprender a programar', uma vez que as diferentes fontes nos apresentam como os professores ensinam, como os alunos estudam, que ferramentas ambos utilizam e como as metodologias e ferramentas adotadas podem auxiliar no processo de ensino e aprendizagem.

A categoria 'metodologia docente' permite convergir com os pensamentos dos autores abordados na fundamentação teórica sobre a influência da metodologia adotada pelo professor quando ensina programação. Além disso, é possível conhecermos diferentes estratégias adotadas pelo professor. Nesta categoria, fica clara que a metodologia adequada para se ensinar programação vai depender do assunto abordado, da estrutura da turma, do nível da turma e da infraestrutura do curso, cabendo ao professor perceber qual e quando deve usar, e quando se faz necessário adaptá-la ou modificá-la. Também é notória a sensibilidade do professor e o reconhecimento, por parte dos alunos, sobre o papel do professor em estar disponível, em buscar adotar as melhores estratégias de ensinar e apoiar os alunos no processo de aprendizagem, nos permitindo inferir a relação entre os parâmetros 'Como' e 'Quem' desta tese.

Fica evidente que independentemente do método adotado, este deve contemplar a prática de resolução de problemas enfatizada na articulação entre a teoria e a prática, para que os alunos consigam adquirir conhecimentos em interpretação e resolução, principalmente para tentar sanar as dificuldades apresentadas no parâmetro 'Porque'. Os dados nos trazem ainda evidências de que a metodologia adotada pelo professor, se não for adequada, pode trazer dificuldades aos alunos, acarretando desistência, reprovação e desmotivação. Sobre a linguagem adota, aqui temos um aspecto interessante em nossa triangulação. Se por um lado não conseguimos evidenciar que a linguagem adotada gera dificuldades no processo de ensino e aprendizagem, por outro lado, as fontes de dados explicitam que a linguagem auxilia o processo. Perceba que os dados apontam que o professor, apesar de adotar uma linguagem específica, deixa em aberto para que o aluno utilize a que tenha maior facilidade, e pode ser por este motivo que conseguimos deduzir que a linguagem de programação auxilia no processo de ensino e aprendizagem.

Na categoria 'metodologia discente', mais uma vez nos deixa clara que a melhor forma de estuda é por meio de prática de resolução de problemas, coadunando com a fundamentação teórica. A

triangulação dos dados nos permite afirmar este é o fator primordial para que os alunos possam aprender a programar. Com a prática, os alunos adquirem maturidade e maior propriedade dos conceitos da lógica da programação, independentemente da linguagem adotada, possibilitando aplicar os conhecimentos adquiridos para a resolução de problemas em diferentes situações e contextos. Sobre a dedicação, a triangulação dos dados também nos permite associar este fator à forma como o aluno estuda e a influência deste no processo de aprendizagem. Quanto mais tempo disponível para estudar, mais tempo o aluno terá para praticar, para buscar recursos de qualidade e para encontrar metodologias que beneficiem sua aprendizagem.

A categoria 'ferramentas utilizadas', não podemos evidenciar que estas influenciam no processo de ensino e aprendizagem, uma vez que a triangulação dos dados traz dados divergentes sobre este aspecto. Entretanto, a coleta dos dados permitiu-nos perceber que os alunos são influenciados pelos professores na escolha inicial das ferramentas adotadas mas que com o passar do tempo, ao conhecer novas ferramentas tornam-se capazes de escolher a que mais se adequam ao seu contexto (tanto em relação à estrutura, quanto em relação à facilidade do uso). Um fator emergente trazido na triangulação dos dados é que embora a literatura nos aponte diferentes ferramentas voltadas para o ensino e aprendizagem de programação, na prática as ferramentas utilizadas no processo são especialmente as IDE. Ou seja, professores e alunos não buscam ou não utilizam ferramentas de simulações, ou de programação visual, ou em bloco, por exemplo, mas ferramentas que são usadas no contexto profissional, e que tenha uma aplicabilidade prática no mercado de trabalho. Esta escolha diverge do que é apontado na fundamentação teórica, e nos leva a refletir que em cursos da área de Informática no Ensino Superior, os professores 'preferem' adotar ferramentas que certamente o aluno irá utilizar fora da Instituição ou ainda, que devido à carga-horária das UC, não se há 'tempo' para testar ferramentas de ensino e aprendizagem que 'fogem' do contexto real, embora estas estejam validadas na literatura.

4.2.7. Parâmetro 'Quanto'

O parâmetro 'Quanto' objetiva conhecer as expectativas dos professores e alunos com o processo de ensino e aprendizagem de programação e o que instiga e os motiva para ensinar e aprender a programar. Este parâmetro está estruturado em três categorias: 'importância da aprendizagem de programação', 'dedicação e iniciativa' e 'visão do futuro'.

Ao longo processo de aprendizagem, os alunos vão adquirindo novos conhecimentos e competências, e aprimorando os que já possuíam. Assim, o aluno começa a perceber a importância da

aprendizagem e enxergar possibilidades que antes estavam claras, especialmente para aqueles que nunca tinham tido contato com a programação antes de ingressar no Ensino Superior.

4.2.7.1. Importância da Aprendizagem de Programação. A fala dos alunos em relação a esta categoria remete a aspectos como motivação, obter êxito no Curso e se encontrar na área. Para os alunos ALU2, ALU4 e ALU5, aprender a programar é o que os motiva a estar no Curso. Embora os cursos da área de Informática permitam que os alunos se desenvolvam em diferentes áreas do conhecimento (PPC, 2019), as falas dos alunos abordam que, para eles, o que os motiva é a área de desenvolvimento de software.

Para os alunos ALU4 e ALU5, a aprendizagem de programação foi importante para que eles pudessem obter êxito, bem como a compreender como programar no cotidiano, ter novas expectativas e competências para escolher a melhor estratégia de desenvolvimento, dependendo do problema a ser resolvido, "por que você não fica preso a só uma forma de programar, você fica vendo as diferentes formas na programação, então isso foi muito relevante foi bem interessante" (ALU5.44).

A fala do aluno ALU3 apresenta o orgulho que este possui em aprender a programar e o diferencial da sua aprendizagem para a vida de outras pessoas:

> O que eu acho bastante interessante em programar, não sei nem se é realmente isso, é em ajudar, por que quando a gente está programando, você pega uma situação, você resolve, e deixa de uma forma simples para os outros conseguirem obter uma resposta de um jeito mais rápido, é muito fascinante, né? Saber que você conseguiu fazer com que aquela pessoa lá consiga resolver um problema dela de um jeito mais simples. E participar desse grupo seleto de pessoas que conseguem resolver problemas para os outros, é muito interessante. (ALU3.53)

A fala do aluno corrobora com o pensamento de Using et al. (2010), que afirma que a arte da programação inclui o conhecimento de ferramentas e linguagens de programação e habilidades de resolução de problemas, e com de Freitas (2015), segundo o qual "o software exerce papel importante na vida do ser humano ajudando, auxiliando, conduzindo e executando as mais diversas tarefas com maior eficiência (cada vez mais rápido) e eficácia (cada vez melhor)" (p. 16).

Sobre a motivação para aprender a programar, o aluno A03 aponta que "é ver como o conteúdo da aula poderia ser usado para resolver um problema" (A03.2). Para o aluno A10, "A forma de interpretar os problemas reais e conseguir desenvolver uma ferramenta que resolva esse problema é muito gratificante". O aluno A12 assinala que "o que me motiva mais a aprender é a magia e o poder que temos em criar coisas novas". Para o aluno 19,

> O que eu mais gosto nas disciplinas [UC] de programação são os projetos finais que os professores geralmente passam no final das disciplinas, pois envolvem todo conhecimento adquirido ao decorrer do semestre, e se torna um desafio muito instigante de se resolver e desenvolver, e isso me motiva bastante. (A19.1)

Sobre poder ter várias formas de resolver o mesmo problema, o aluno A04 afirma que "ter mais de uma visão sobre o mesmo problema para resolver, e ter mais tempo e saber que tipo de área na programação que gosto, ajudou a ter vontade de trabalhar em novos projetos" (A04.1). O aluno A14 fala que "o que mais gostei de ver ao cursar a disciplina de Construção de Algoritmos foi ver que existem várias formas de programar, e que não existe 'forma certa' ou 'forma errada' e sim, formas mais eficazes" (A14.2).

As falas desses alunos remetem aos fatores apresentados na fundamentação teórica, tais como criatividade (Ambrósio et al., 2011; Oliveira et al., 2015; Silva et al., 2015), raciocínio lógico (Ambrósio et al., 2011; Araújo et al., 2019; Gomes, 2010; Oliveira et al., 2015; Santos et al., 2015; Santos et al., 2020; Al-Imamy et al., 2006; Bosse & Gerosa, 2017; T. Castro et al., 2011; T. H. Castro et al., 2008; Costa, 2019; T. A. N. de Oliveira & Rebouças, 2018; Ortiz-Ortiz et al., 2018; Rezende & Bispo, 2018; Skalka & Drlik, 2018) e a habilidade de resolução de problemas (Ambrósio et al., 2011; Dijkstra, 1989; Gomes, 2010; Medeiros, 2019; Medeiros et al., 2020; Oliveira et al., 2015; da Silva & Falcão, 2020; da Silva et al., 2015; Villalobos et al., 2009; (Aparicio & Costa, 2018; Bosse & Gerosa, 2017; T. Castro et al., 2011; T. H. Castro et al., 2008; Chang et al., 2000; Chuchulashvili et al., 2016; Costa, 2019; Eltegani & Butgereit, 2015; A. Gomes & Mendes, 2015; Nawahdah et al., 2015; T. A. N. de Oliveira & Rebouças, 2018; Ortiz-Ortiz et al., 2018; Rezende & Bispo, 2018; A. Santos et al., 2010, 2011, 2013; Skalka & Drlik, 2018; Souleiman, 2017).

Com os depoimentos dos alunos, percebemos que o que mais os motiva e os leva a boas expectativas no processo de aprendizagem é conhecer as possibilidades práticas que a aprendizagem de programação oferece. Na opinião dos alunos, a motivação é ver a aplicabilidade dos conceitos, ver o 'poder' da programação na resolução de problemas do dia a dia. Sobre as perspectivas futuras, os alunos apontam que com a programação, já conseguem desenvolver seus próprios programas e participar de projetos na área, além de vislumbrarem crescer dentro da área. Os alunos percebem a importância da dedicação, prática e da aprendizagem contínua para melhor aproveitamento ao longo do Curso.

4.2.7.2. Dedicação e Iniciativa. Outro ponto importante que os alunos percebem que é fundamental em seu processo de aprendizagem para obter sucesso no futuro diz respeito à categoria 'dedicação e iniciativa'. Estas estão relacionadas às tomadas de decisão feitas pelos alunos ao longo do processo de aprendizagem. Por dedicação, entendemos o desprendimento destes em função do estudo, e por iniciativa, as atitudes dos alunos em ser proativos e buscar outras formas de aprender para além daquelas vistas em sala ou orientadas pelos docentes.

Nas falas dos alunos, podemos apontar os seguintes resultados que podem ser alcançados com a dedicação aos estudos: não desistir do curso (ALU2), ultrapassar as dificuldades (ALU2; ALU3) e buscar novos conhecimentos (ALU6). Acerca da dedicação apontada pelos alunos, podemos perceber de acordo com o pensamento de Carvalho e Síveres (2013) que os estudantes demonstraram motivação, envolvimento e esforço no processo de aprendizagem, pois mesmo enfrentando dificuldades mantiveram-se persistentes para realizar as atividades e cumprir com as responsabilidades que assumiram.

Em relação à iniciativa, percebemos que em diferentes momentos do percurso acadêmico, os alunos por si próprios desenvolvem estratégias de estudo, tais como: buscar materiais de apoio (ALU2), desenvolver programas por conta própria (ALU4; ALU6), buscar apoio do professor (ALU4), aprofundar os conhecimentos fora da sala de aula (ALU4; ALU6) e estudar a Língua Inglesa (ALU4; ALU5). No que concerne ao desenvolvimento da capacidade de iniciativa, percebemos com base em Kamii (2002) que diante de algumas circunstâncias, os alunos consideraram os fatos relevantes e decidiram sozinhos a melhor maneira de agir para resolver os problemas que ocorreram durante o processo de aprendizagem. Isso sinaliza que essa experiência vivida, conforme argumenta Bruner (2009) pode ser projetada em outras situações futuras, uma vez que o processo formativo pode estruturar seu comportamento na sociedade. Sobre este aspecto, o professor PROF1 aborda que

> Se em sala de aula é cobrado para você fazer o exercício A, B, C, então você vai e além disso aí, você crie seus próprios, vá atrás de outros exercícios, D, E F, G, Z... então seja proativo nesse sentido. 'Ah, vou saber curiosidades sobre a linguagem de programação que eu utilizo', 'Ah, vou procurar novos caminhos para fazer aquelas mesmas linhas de código, sei lá, tô [Sic] com dez linhas de código, então como é que eu posso reduzir aí pra cinco...', então tem que ter esse tipo de curiosidade. (PROF1.43)

Para o PROF2, é muito bom o aluno ter iniciativa pois às vezes os alunos desenvolvem soluções que o professor não esperava e até mesmo melhores que a que o professor propôs. Assim, a iniciativa e dedicação são aspectos relevantes para a aprendizagem de programação.

4.2.7.3. Visão do Futuro. Com a aprendizagem, o aluno passa a ter novas e diferentes perspectivas para o futuro. Se antes de entrar no Curso, o aluno pouco ou nada conhecia de programação, ao longo do percusso acadêmico esse passa a ter maior compreensão do que a aprendizagem pode lhe oferecer, e projetar seu futuro e comportamentos em determinadas situações.

Sobre as perspectivas futuras, para o aluno A11, "o que mais gostei nas disciplinas de programação é que foi delas que saíram meus primeiros projetos de desenvolvimento" (A11.3). Para o aluno A25, aprender a programar "foi muito bom pois foi daí que comecei a criar meus programas, o que me instigou. O que me ajuda a continuar a aprender é meu objetivo de ser o melhor que eu puder nessa área" (A25.4).

No recorte abaixo, o aluno ALU1 aponta como perspectivas futuras o desenvolvimentos de alguns projetos que já estão em andamento, fruto da sua participação em projetos de Iniciação Científica:

> Meio que eu já tenho alguns projetos, então ou vai ser professor [nome do professor omitido] ou professor [nome do professor omitido] que irá me orientar, por que eu tenho projeto com eles dois, meu PIBIC é professor [nome do professor omitido] e eu tenho um projeto que é relacionado com o Governo do Estado que é com o professor [nome do professor omitido]. (ALU1.59)

De acordo com Pinho (2017), levando em consideração a tríade docente-discentes-curso universitário, menciona os benefícios que podem ser conseguidos através da Iniciação Científica, como: estímulo à problematização, criação de linhas de pesquisa, legitimação da produção de novos conhecimentos, o que pode ser identificado na fala do ALU1 e que coaduna com PPC (2019), que apresenta que "a participação dos alunos nas atividades de Pesquisa e Extensão desenvolvidas na área da Ciência da Computação possibilita a inserção de ideias inovadoras no mercado de trabalho, gerando assim a capacidade de alavancar e/ou transformá-lo" (p.28).

Para o aluno ALU2, algo que ele percebeu com a aprendizagem de programação é a necessidade de trabalho em equipe. Como visto no parâmetro 'Como', o aluno ALU2 afirma preferir estudar individualmente, entretanto, ele reconhece que é algo que precisa ser melhorado:

> É inclusive uma coisa que eu quero melhorar [estudar sozinho], eu disse aos meus amigos que ia tentar criar de vez em quando um repositoriozinho para a gente começar a trabalhar em equipe, por que o mercado não é você individual, é você com outras pessoas. (ALU2.46)

Ao falar sobre 'mercado de trabalho' o aluno demonstra maturidade e consciência de como poderá ser o futuro. O aluno ALU4, na mesma linha de pensamento do ALU2, fala nas perspectivas do futuro ao buscar fazer atividades que possam agregar em seu currículo.

> Eu tento aprender coisas novas e ir atrás mesmo, tipo coisas que às vezes podem ser boas para o meu currículo, ou pra uma experiência que um dia eu possa precisar disso, ou alguma coisa assim, mas que não fosse minha área assim, por que o que me instigar é estar aprendendo coisas novas, tipo, entender como que é feito um reconhecimento facial, como é que é essas coisas, por que antigamente eu era apenas um usuário comum, não era um desenvolvedor, então a gente aprende muita coisa desenvolvendo, ir vendo como é, e isso me ajuda muito a ir atrás. (ALU4.54)

É interessante verificar que o aluno consegue elencar suas capacidades e distinguir seus papeis entre usuário final e desenvolvedor. Ele ainda vê que com a aprendizagem, ele é capaz de buscar melhorias para si, e com a prática consegue aprender novos conceitos que poderão ser usados no futuro. Por fim, o professor PROF2 aconselha que é importante o aluno sonhar alto e perceber que

> aquele conhecimento de programação vai influenciar no futuro dele como profissional, né, que ele pode ser um bom programador de jogos, que ele pode ser um bom programador científico, ou que se ele se destacar como programador, ele pode galgar uma posição profissional numa empresa como Google, Microsoft, né, eu acho que é fundamental que o aluno de computação, nesse aspecto, ele sonhe alto. (PROF2.97)

A fala do professor PROF2 é motivadora e nos faz refletir que ter autoestima alta e autoconfiança permite ao estudante galgar o sucesso na área que desejar estar.

4.2.7.4. Conclusão. O parâmetro 'Quanto' nos permite 'caracterizar os fatores que podem influenciar para o aluno aprender a programar' e nos traz aspectos emergentes não identificados na fundamentação teórica, mas que podemos considerar relevantes para compreendermos como ocorre o processo de ensino e aprendizagem de programação.

Na categoria 'importância da aprendizagem de programação', fica evidente que o aluno sente-se motivado a aprender quando está fazendo algo que realmente gosta. Quando o aluno percebe a aplicabilidade da programação e os benefícios que a carreira de programador pode trazer para a sociedade, sente orgulho da escolha que fez para a sua formação. E isto nos mostra que o processo pode tornar-se menos difícil. Os dados triangulados nos mostram que a aprendizagem permite que os alunos tenham capacidade de resolver os mais diversos problemas, em diferentes contextos. Para além disso, os dados apontam para o que abordamos na fundamentação teórica de que a criatividade,

raciocínio lógico e habilidade de resolução de problemas são fatores importantes, e que fazem parte da formação do programador.

Sobre a 'dedicação e iniciativa', fator este abordado também no parâmetro 'Como', percebemos que o 'Quanto' um aluno se dedica e tem iniciativa de estudar está diretamente ligada à sua aprendizagem, e que trazem benefícios importantes para seu percurso acadêmico tais como, não desistir do curso e buscar novos conhecimentos, o que diminui as dificuldades e suas consequências. A iniciativa está ligada à proatividade, abordada no parâmetro 'What' e revelada pelos professores e pelos documentos oficiais como uma característica importante para aprender a programar. A categoria 'visão do futuro' nos mostra o que o aluno percebe acerca da aprendizagem de programação e as possibilidades que esta oferece tanto no percurso acadêmico quanto no mercado de trabalho. Podemos afirmar que, mesmo ainda estando em sala de aula, os alunos já vislumbram o futuro e buscam cada vez mais adquirir e melhorar seus conhecimentos e habilidades. É importante ainda, trazer o aspecto de autoconfiança alta e de 'sonhar alto', como fatores que motivam, incentivam e influenciam no processo de ensino e aprendizagem de programação.

4.3. Síntese Conclusiva do Capítulo

Neste capítulo apresentamos o estudo de caso realizado na nossa pesquisa, apresentado em duas seções: 'caracterização do caso' e 'percepções, vozes e palavras dos participantes do processo', desenvolvidas a partir dos procedimentos metodológicos apresentados no capítulo 3. Primeiramente, na seção 'caracterização do caso', pudemos conhecer o Curso, seus objetivos, estrutura e como as UC de programação estão inseridas no percurso acadêmico. Conhecer o contexto em que o caso de estudo está inserido, nos permite imergir na realidade de como ocorre o processo de ensino e aprendizagem de programação naquela Instituição de Ensino Superior.

Na seção 'percepção, vozes e palavras dos participantes do processo' percebemos com mais profundidade, a partir da triangulação dos dados, o que nos dizem as nossas fontes (alunos, professores e documentos oficiais do Curso), sobre 'O que' é necessário para ensinar e aprender programação, 'Quem' participa do processo, 'Onde' se estuda e busca informação, 'Quando' se estuda programação no Curso, 'Por que' o processo é complexo, 'Como' se ensina e se aprende e 'Quanto' se espera alcançar com as aprendizagens.

Esta configuração em que as reflexões dos atores do estudo de caso estão organizadas, aliada à caracterização do caso, nos permitiu apresentar as interpretações das análises de forma coerente, de maneira que pudéssemos expressar as complexidades que envolvem o processo de ensino e aprendizagem de programação. De fato, as dificuldades existem e as percepções, falas e palavras

corroboram, na maioria das vezes com o que apresentamos na fundamentação teórica desta tese. Podemos afirmar que o caso escolhido é capaz de apresentar uma circunstância representativa, possibilitando fornecer informações sobre experiências para outros, indo ao encontro com Yin (2015).

Dessa forma, este capítulo pautou-se na busca para responder à questão de partida e os objetivos desta tese, com a triangulação dos dados coletados, e ainda reforçar os conhecimentos adquiridos âmbito da fundamentação teórica, permitindo a construção de novos saberes.

5. Considerações Finais

Este capítulo apresenta nossas considerações, reflexões e conclusões acerca do trabalho desenvolvido ao longo dessa tese de doutoramento. Para chegar até aqui, tivemos que percorrer todo um trajeto, que veio desde a minha iniciação como aluna em um curso superior na área de Informática, passando pela minha experiência docente, que me trouxeram inquietações para serem exploradas enquanto pesquisadora, e que me levaram ao doutoramento em Ciências da Educação.

O propósito do nosso trabalho é responder à questão de partida: 'como ocorre o processo de ensino e aprendizagem de programação no Ensino Superior' a partir da contemplação dos objetivos traçados para o estudo. Podemos afirmar que o processo é complexo, pela natureza do que é o ensino e aprendizagem de programação, e envolve diferentes aspectos que pudemos conhecer e evidenciar nesta tese.

O capítulo está organizado da seguinte forma: 'respostas à questão de partida e aos objetivos', 'contribuições do estudo para o ensino e aprendizagem de programação', 'limitações do estudo', 'sugestões para trabalhos futuros' e 'conclusões finais'.

5.1. Respostas à Questão de Partida e aos Objetivos

Para apresentar as respostas, vamos relembrar o que apresentamos ao longo dos capítulos da tese.

No capítulo 'fundamentação teórica' abordamos conceitos importantes apresentados em duas diferentes seções 'definição e contextualização' e 'caracterização'.

Na seção 'definição e contextualização' vimos que o processo de aprendizagem de programação é tarefa considerada complexa e que coloca desafios importantes aos docentes de Informática. Compreendermos os conceitos de programação à luz de diferentes autores e que convergem para o processo de interpretação e compreensão do problema – especificação dos requisitos para resolvê-lo – implementação de um plano de resolução – implementação do algoritmo em uma linguagem de programação – execução do programa e testagem, passos esse que são cíclicos e iterativos, até que se atinja o resultado esperado. Percebemos a complexidade da programação por estar presente em diferentes áreas do conhecimento, por ter diferentes designações e níveis de formação, embora seja predominante em cursos de Informática e áreas afins. Com o que abordamos nessa seção, pudemos começar a conhecer 'como ocorre o processo de ensino e aprendizagem de programação no Ensino Superior' uma vez que aponta peculiaridades específicas para esse processo.

Na seção 'caracterização' mergulhamos no que nos fala a literatura, na busca de encontrar caminhos que nos norteassem a compreender nossa questão de partida e os objetivos. Embora a

pesquisa macro envolvesse o processo e o ensinar e aprender programação, a grande maioria dos achados eram no tocante às dificuldades enfrentadas por professores e alunos ao longo do processo. Esse fato nos trouxe evidências da complexidade do processo, especialmente pela quantidade e diversidade das dificuldades encontradas.

Com a 'caracterização' iniciamos às respostas dos objetivos elencados nesta tese, de acordo com cada uma de suas subseções.

O objetivo 'Identificar as dificuldades enfrentadas por professores e alunos durante o processo de ensino e aprendizagem, e o que estas dificuldades acarretam' é contemplado na subseção 'dificuldades enfrentadas pelo aluno durante a aprendizagem', que mostrou-nos que estas podem ser devido ao currículo do estudante (base matemática, dificuldade em interpretação de textos, dificuldades em lógica), a questões socioeconômicas (falta de recursos para estudar, ter que conciliar estudo com trabalho), ou até mesmo à estrutura do curso superior (turmas muito grandes, formação docente inadequada, turmas heterogêneas). Notamos que as dificuldades podem gerar consequências tais como retenção, desistência, frustração e desmotivação dos alunos, que podem afetar o seu processo de aprendizagem e seu percurso acadêmico.

Os objetivos 'caracterizar os fatores que podem influenciar para o aluno aprender a programar' e 'explicitar os conhecimentos e habilidades necessários para aprender a programar' são comtemplados na subseção 'condições para aprendizagem de programação', que apresenta as condições que podem auxiliar no processo de aprendizagem de programação, e que vêm desde a forma como ocorre o primeiro contato do aluno com a programação, ou seja, a abordagem pedagógica, a linguagem de programação adotada pelo professor e o apoio dado pelo docente, até a forma como o aluno organiza-se para estudar e se este está motivado para aprender. Por outro lado, um aluno que estuda de maneira superficial ou resolve poucos exercícios certamente terá sua aprendizagem afetada. Além destes fatores, destacam-se como habilidades importantes para aprender a programar a autoconfiança, autoestima, proatividade e planejamento; como conhecimento destaca-se a abstração, algoritmia, conhecimento em matemática e conhecer linguagens de programação (embora não seja necessário esse conhecimento em um primeiro momento; e como competência é interessante que o aluno seja criativo, cooperativo, saiba trabalhar em equipes e consiga ligar com regras.

O objetivo 'inventariar metodologias e ferramentas usadas para o ensino e aprendizagem de programação' é alcançado na subseção 'metodologias e ferramentas', que apresenta diferentes metodologias e ferramentas que vêm sendo desenvolvida ao longo dos últimos anos. Podemos destacar que independentemente da metodologia ou ferramenta adotada, a aprendizagem de

programação destaca-se nestas três estratégias: prática – uso de bom material de apoio – recorrer a aulas de programação que sejam confiáveis. Aliada a estas estratégias, encontramos na literatura a programação por pares, a sala de aula invertida, o uso de jogos e linguagem de programação visual, e ferramentas de apoio tais como ambiente de programação visual, de programação com blocos, plataformas de recomendação de atividades e de feedback automático. Entretanto, embora sejam fortes os esforços de pesquisadores na busca de estratégias e no desenvolvimento de ferramentas, não existem evidências da melhor metodologia ou ferramenta a ser adotada, uma vez que isto dependerá do contexto em que está se ensinando a programação.

Nesse capítulo pudemos perceber que o processo de ensino e aprendizagem de programação é complexo porque envolve diferentes aspectos e fatores que estão entrelaçados entre si, podendo auxiliar ou afetar o processo, dependendo de suas características. Para compreender estes aspectos e representá-los, concebemos categorias, baseadas no framework 5W2H, de maneira que pudéssemos organizar os dados coletados das nossas fontes de evidências nos parâmetros 'O que', 'Quem', 'Onde', 'Quando', 'Porque', 'Como' e 'Quanto'.

No capítulo 'enquadramento metodológico' apresentamos a 'abordagem metodológica' e o 'procedimento metodológico' da pesquisa desenvolvida. Na 'abordagem metodológica' apresentamos que devido à natureza complexa do objeto de estudo, para responder à questão de partida e alcançar os objetivos, desenvolvemos um estudo de caso, cujo caso está inserido no âmbito de um Curso de Ensino Superior na área de Informática no contexto brasileiro. Para tanto, usamos diferentes estratégias quantitativas e qualitativas para a obtenção dos dados a serem analisados com a triangulação, de acordo com os conceitos da abordagem metodológica mista.

Na seção 'procedimento metodológico' apresentamos todas as etapas realizadas no procedimento, que incluem a elaboração do projeto apresentado e aprovado pelo CEP-UERN, a apresentação do estudo de caso incluindo a justificativa e características do caso, a recolha de dados apontando os instrumentos de recolha adotados, o registro e tratamento de dados apresentando as ferramentas usadas e como os dados foram armazenados, e a análise e interpretação dos dados, a fim de apresentarmos as estratégias adotadas para contemplar a abordagem metodológica do nosso estudo.

Esse capítulo é, portanto, o norteador de todo o processo realizado para que pudéssemos responder à questão de partida e aos objetivos desse estudo.

No capítulo ‘estudo de caso’, organizado em duas seções: ‘caracterização do caso’ e ‘percepções, vozes e palavras dos participantes do processo’, apresentamos a caracterização do caso e os resultados da triangulação dos dados.

A seção ‘caracterização do caso’, pudemos conhecer o caso estudado, seus objetivos, estrutura e como as UC de programação estão inseridas no percurso acadêmico. Conhecer o contexto em que o caso de estudo está inserido, nos permite imergir na realidade de como ocorre o processo de ensino e aprendizagem de programação naquela Instituição, nos auxiliando na interpretação dos dados coletados.

Na seção ‘percepção, vozes e palavras dos participantes do processo’ percebemos com mais profundidade, a partir da triangulação dos dados, o que nos dizem as nossas fontes (alunos, professores e documentos oficiais do Curso), sobre ‘O que’ é necessário para ensinar e aprender programação, ‘Quem’ participa do processo, ‘Onde’ se estuda e busca informação, ‘Quando’ se estuda programação no Curso, ‘Por que’ o processo é complexo, ‘Como’ se ensina e se aprende e ‘Quanto’ se espera alcançar com as aprendizagens.

Esta configuração em que as reflexões dos atores do estudo de caso estão organizadas, aliada à caracterização do caso, nos permitiu apresentar as interpretações das análises de forma coerente, de maneira que pudéssemos expressar as complexidades que envolvem o processo de ensino e aprendizagem de programação e responder à questão de partida e contemplar nossos objetivos com mais clareza e fidedignidade, à luz da fundamentação teórica realizada nesta tese.

O parâmetro ‘O que’ nos ajuda a responder os objetivos ‘caracterizar os fatores que podem influenciar no processo de ensino e aprendizagem’ e ‘explicitar os conhecimentos e habilidades necessários para aprender a programar’. O parâmetro ‘Quem’ nos ajuda a ‘caracterizar os fatores que podem influenciar no processo de ensino e aprendizagem de programação’, tendo em vista o papel desempenhado por professores, alunos, familiares e instituições. O parâmetro ‘Onde’ nos ajuda a ‘Inventariar metodologias e ferramentas usadas para o ensino e aprendizagem de programação’ e ‘Caracterizar os fatores que podem influenciar para o aluno aprender a programar’, uma vez que busca elucidar onde o aluno estuda e busca seus recursos para a aprendizagem, mas também consegue nos dar chances de ‘identificar as dificuldades enfrentadas por professores e alunos durante o processo de ensino e aprendizagem’, quando, por exemplo, o aluno não tem acesso a ambiente adequado para estudar ou a recursos de apoio que sejam confiáveis.

O parâmetro ‘Quando’ nos ajuda a ‘caracterizar os fatores que podem influenciar para o aluno aprender a programar’ e ‘identificar as dificuldades enfrentadas por professores e alunos durante o

processo de ensino e aprendizagem, e o que estas dificuldades acarretam', uma vez que nos apresenta quando os alunos e professores ingressaram no curso, as UC cursadas e ministradas e as percepções acerca do percurso.

. O parâmetro 'Porque' nos permite claramente 'identificar as dificuldades enfrentadas por professores e alunos durante o processo de ensino e aprendizagem, e o que estas dificuldades acarretam', uma vez que nos apresenta evidências sobre as principais dificuldades vivenciadas por professores e alunos ao longo do processo de ensino e aprendizagem de programação e as consequências trazidas por estas para ambos.

A triangulação dos dados realizada no parâmetro 'Como', nos permite 'inventariar metodologias e ferramentas usadas para o ensino e aprendizagem de programação' bem como nos ajuda a 'caracterizar os fatores que podem influenciar para o aluno aprender a programar', uma vez que as diferentes fontes nos apresentam como os professores ensinam, como os alunos estudam, que ferramentas ambos utilizam e como as metodologias e ferramentas adotadas podem auxiliar no processo de ensino e aprendizagem. O parâmetro 'Quanto', por sua vez, nos permite 'caracterizar os fatores que podem influenciar para o aluno aprender a programar' e nos traz aspectos emergentes não identificados na fundamentação teórica, mas que podemos considerar relevantes para compreendermos como ocorre o processo de ensino e aprendizagem de programação.

5.2. Contribuições do Estudo para o Ensino e Aprendizagem de Programação

Com a realização deste estudo, várias das questões que nos rodeavam acerca da complexidade do processo de ensino e aprendizagem de programação começaram a ser respondidas. Embora a experiência acadêmica, enquanto alunos e professores, nos tragam suposições, a Ciência consegue evidenciá-las.

Como primeiro contributo, pudemos evidenciar as dificuldades enfrentadas por alunos e professores, clarificá-las, de forma que estes atores do processo consigam percebê-las em si e nos demais participantes, a fim de questionar, tentar sanar ou diminuí-las, para que alunos e professores possam ter experiências mais proveitosas e exitosas ao longo do processo.

Como segundo contributo, caracterizamos os fatores que ajudam o aluno a aprender a programar, que podem ser fatores pessoais ou fatores externos a este. Com esta caracterização, nosso objetivo é apresentá-los à comunidade acadêmica para que ela os reconheça e possa explorá-los, adquiri-los e melhorá-los, beneficiando todos os que participam do ensino e aprendizagem de programação.

Como terceiro contributo, explicitamos as competências e habilidades importantes e necessárias para aprender a programar. Algumas destas são abrangentes para quaisquer áreas do conhecimento, mas outras são específicas para a aprendizagem de programação. Torná-las explícitas à comunidade acadêmica permite que professores, alunos e instituições de ensino busquem estratégias que permitam ao aluno adquiri-las ou melhorá-las, e formas de mitigar as dificuldades enfrentadas pela falta desses conhecimentos e habilidades.

Como quarto contributo, inventariamos metodologias e ferramentas que podem ser usadas no processo de ensino e aprendizagem de programação. Nossa fundamentação teórica nos permitiu conhecer diferentes estratégias e ferramentas, mas que nem sempre são utilizadas no dia a dia de sala de aula. A intenção deste contributo é que professores e alunos tenham acesso a estas metodologias e ferramentas, possam conhecê-las, adotá-las e assim poder perceber melhores formas de ensinar e aprender programação.

Como quinto contributo, adotamos o *framework* 5W2H para organizar de acordo com os parâmetros 'O que', 'Quem', 'Onde', 'Quando', 'Porque', 'Como' e 'Quanto', a interpretação da triangulação dos dados advindos de nossas fontes de evidência. O intuito é que a comunidade acadêmica perceba que o ensino e aprendizagem de programação ocorre sob vários aspectos que estão interligados.

5.3. Limitações do Estudo Desenvolvido

Ao longo do desenvolvimento deste estudo, nos deparamos com diferentes dificuldades que nos revelaram algumas limitações.

Uma limitação deu-se por parte da participação dos alunos na recolha dos dados. No projeto submetido ao CEP, era previsível a realização de coletas por meio de diários de bordo (ou semanários de bordo) desenvolvidos por alunos que estavam cursando UC de programação. Apesar de todos os esforços de minha parte e dos professores das disciplinas, nenhum dos alunos aceitou contribuir, de forma que não pudemos obter essa fonte de dados para nosso estudo.

Outra limitação foi a pandemia da Covid-19 que atingiu a todos inesperadamente e que também nos levou a buscar diferentes estratégias para a realização da nossa pesquisa, em especial o estudo de caso. Assim como o diário de bordo, era previsto para este estudo a observação participante, uma técnica importante para o estudo de caso. Entretanto devido à necessidade de distanciamento social, as aulas passaram da presencialidade para o formato remoto, impedindo, que fizéssemos a observação *in loco.*

Ressalta-se que mesmo com essas limitações, a produção da tese foi bastante proveitosa, especialmente por se tratar de uma temática relevante para a área de Informática, especialmente no tocante ao ensino e aprendizagem de programação, e para mim enquanto aluna, professora e pesquisadora da área.

5.4. Sugestões para Trabalhos Futuros

Apesar de termos atingido nossos objetivos e respondido à nossa questão de partida, é importante refletirmos que ainda se há muito para pesquisar e para desenvolver, inspirados nos contributos dessa tese.

Essa investigação não pretende ser uma pesquisa finalizada, mas um desafio a outros estudiosos que pretendam aprofundar ou buscar caminhos diferentes para as inquietações existentes no ensino e aprendizagem de programação. Nesse sentido, sugerimos que outros estudos possam ser realizados, a fim de perceber:

- Os contributos que a inserção da programação na educação básica pode trazer para a Educação.
- Os contributos que a realização de um estudo de caso que contemple a observação participante pode agregar a este estudo.
- A realização de um estudo mais amplo que contemple mais de um caso.
- O desenvolvimento de estratégias que possam diminuir ou eliminar as dificuldades enfrentadas por professores e alunos no ensino e aprendizagem de programação.

5.5. Conclusões Finais

Por fim, a nossa pesquisa 'Ensino e aprendizagem de programação: estudo de caso no Ensino Superior' que está inserida no âmbito de Ciências da Educação, do Instituto de Educação da Universidade do Minho, na especialidade Tecnologia Educativa, permitiu-nos compreender como ocorre o processo de ensino e aprendizagem de programação, à luz da literatura e das percepções, vozes e palavras das nossas fontes de evidências, com o desenvolvimento da triangulação dos dados.

Permitiu-nos ainda inventariar metodologias e ferramentas usadas para o ensino e aprendizagem de programação; caracterizar os fatores que podem influenciar para o aluno aprender a programar; explicitar os conhecimentos e habilidades necessários para aprender a programar e identificar as dificuldades enfrentadas por professores e alunos durante o processo de ensino e aprendizagem, e o que estas dificuldades acarretam

Com isso, estudo identifica contributos relacionados aos objetivos elencados, de forma que os resultados obtidos podem servir de suporte para a construção de novos conhecimentos para o

desenvolvimento de estratégias que possam contribuir para o processo de ensino e aprendizagem de programação.

Referências Bibliográficas

Adán-Coello, J. M., Menezes, W. S. de, Faria, E. S. J. de, & Tobar, C. M. (2008). Conflito Sócio-cognitivo e Estilos de Aprendizagem na Formação de Grupos para o Aprendizado Colaborativo de Programação de Computadores. *Revista Brasileira de Informática na Educação*, *16*(03), Article 03. https://doi.org/10.5753/rbie.2008.16.03.%p

Aguilar, L. J. (2008). *Fundamentos de Programação-: Algoritmos, estruturas de dados e objetos*. AMGH Editora.

Al-Imamy, S., Alizadeh, J., & Nour, M. (2006). On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process. *Journal of Information Technology Education-Research*, *5*, 271–283.

Almeida, T., Netto, J. F., da Silva, R., & Custódio, T. (2017). Laboratório remoto de robótica como elemento motivador para a aprendizagem de programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, *28*(1), 665. http://dx.doi.org/10.5753/cbie.sbie.2017.665.

Alves, F. P., & Jaques, P. (2014). Um ambiente virtual com feedback personalizado para apoio a disciplinas de programação. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *3*(1), 51. http://dx.doi.org/10.5753/cbie.sbie.2014.1078.

Alves, G., Rebouças, A., & Scaico, P. (2019). Coding Dojo como Prática de Aprendizagem Colaborativa para Apoiar o Ensino Introdutório de Programação: Um Estudo de Caso. *Anais do Workshop sobre Educação em Computação (WEI)*, 276–290. https://doi.org/10.5753/wei.2019.6636

Ambrósio, A. P. L., Almeida, L. S., Macedo, J., Santos, A., & Franco, A. H. (2011). *Programaçâo de computadores: Compreender as dificuldades de aprendizagem dos alunos*. 13. Revista Galego-Portuguesa de Psicoloxía e Educacion. Vol.19, n.1, Ano 16º-2011 ISNN: 1138-1663.

Anderson, M., & Gavan, C. (2012). Engaging undergraduate programming students: Experiences using lego mindstorms NXT. *Proceedings of the 13th annual conference on Information technology education*, 139–144.

André, M. (2013). O que é um Estudo de Caso Qualitativo em Educação? *Revista da FAEEBA - Educação e Contemporaneidade*, *22*(40), 95–103. https://doi.org/10.21879/faeeba2358-0194.2013.v22.n40.p95-103

Aparicio, J. T., & Costa, C. J. (2018). A Virtual Robot Solution to support Programming Learning. *2018 13th Iberian Conference on Information Systems and Technologies (CISTI). (pp. 1-6). IEEE*.

Arantes, F. L., & Ribeiro, P. E. J. (2017). Desenvolvimento do Pensamento Computacional com Valores da Ética Hacker. *Informática na educação: teoria & prática*, *20*(2 mai/ago).

Araujo, R., Simão, A., Malucelli, A., Zorzo, A., Monteiro, J. A., & Chaimowicz, L. (2019). Referenciais de Formação para os Cursos de Pós-Graduação Stricto Sensu em Computação 2019. *Sociedade Brasileira de Computação*.

Aureliano, V. C. O., Tedesco, P. C. de A. R., & Giraffa, L. M. M. (2016). Desafios e oportunidades aos processos de ensino e de aprendizagem de programação para iniciantes. *Anais do Workshop sobre Educação em Computação (WEI)*, 71–80. https://doi.org/10.5753/wei.2016.9650

Barbosa, A. P. M., Andrade, G. E., & Hoss, D. J. (2019, junho 26). Visual Programmer – ViP. *Anais do Encontro Nacional de Computação dos Institutos Federais (ENCompIF)*. Anais do VI Encontro Nacional de Computação dos Institutos Federais. https://doi.org/10.5753/encompif.2019.6345

Barbosa, L. S., Fernandes, T. C., & Campos, A. M. (2011). Takkou: Uma ferramenta proposta ao ensino de algoritmos. *XVIII Workshop sobre Educação em Computação (WEI 2011)*.

Bardin, L. (1977). *Análise de Conteúdo*. Lisboa: edições, v. 70, 225 p..

Begel, A., & Klopfer, E. (2007). Starlogo TNG: An introduction to game development. *Journal of E-Learning*, *53*(2007), 146.

Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. *ACM inroads*, *10*(2), 30–36.

Bento, M., Lencastre, J. A., & Pereira, I. S. P. (2016). Dispositivos móveis no desenvolvimento de competências de interpretação de texto no 1. O Ciclo do Ensino Básico. *In Carvalho, A.A.A.; Cruz, S.; Marques, C. G.;*

Moura, A.; Santos, M. I., & Zagalo, N. (2016) (orgs). Atas do Encontro sobre Jogos e Mobile Learning, (pp. 620-625). Coimbra: Universidade de Coimbra, FPCE, LabTE.

Berssanette, J. H. (2016). *Ensino de programação de computadores: Uma proposta de abordagem prática baseada em Ausubel* [Tese de Mestrado]. Universidade Tecnológica Federal do Paraná.

Berssanette, J. H. (2020). *Construindo "novos" caminhos para o ensino de programação por meio do uso de metodologias ativas de aprendizagem e teoria da carga cognitiva* (1◦ ed). Produto da Tese de Doutorado (Doutorado em Ensino de Ciência e Tecnologia) - Universidade Tecnológic. https://doi.org/10.29327/517802

Bez, J. L., Tonin, N. A., & Rodegheri, P. R. (2014). URI Online Judge Academic: A tool for algorithms and programming classes. *2014 9th International Conference on Computer Science Education*, 149–152. https://doi.org/10.1109/ICCSE.2014.6926445

Biggs, J., & Tang, C. (2011). *Teaching for quality learning at university*. McGraw-hill education (UK).

Bittencourt, R. A., Rocha, A. S., Santana, B. L., Santana, C. S., Carneiro, D. A., Borges, G. A., Chalegre, H. S., Silva, J., Santos, J., Silva, L. A., & others. (2013). Aprendizagem de programaçao através de ambientes lúdicos em um curso de engenharia de computaçao: Uma primeira incursão. *XXI Workshop sobre Educação em Computação–XXXIII Congresso da Sociedade Brasileira de Computação*.

Bosse, Y., & Gerosa, M. A. (2017). Difficulties of Programming Learning from the Point of View of Students and Instructors. *IEEE Latin America Transactions*, *15*(11), 2191–2199. https://doi.org/10.1109/TLA.2017.8070426

Bruner, J. S. (2009). *The Process of Education, Revised Edition*. Harvard University Press.

Brusilovsky, P., Edwards, S., Kumar, A., Malmi, L., Benotti, L., Buck, D., Ihantola, P., Prince, R., Sirkiä, T., Sosnovsky, S., & others. (2014). Increasing adoption of smart learning content for computer science education. *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, 31–57.

Cambruzzi, E., & Souza, R. M. de. (2015). Robótica Educativa na aprendizagem de Lógica de Programação: Aplicação e análise. *Anais do Workshop de Informática na Escola*, *21*(1), 21–28. https://doi.org/10.5753/cbie.wie.2015.21

Cardoso, R., & Antonello, S. (2015). Interdisciplinaridade, programação visual e robótica educacional: Relato de experiência sobre o ensino inicial de programação. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *4*(1), 1255. https://doi.org/10.5753/cbie.wcbie.2015.1255

Carlisle, M. C., Wilson, T. A., Humphries, J. W., & Hadfield, S. M. (2005). RAPTOR: a visual programming environment for teaching algorithmic problem solving. *Acm Sigcse Bulletin*, *37*(1), 176–180.

Casarotto, R. I., Bernardi, G., Cordenonsi, A. Z., & Medina, R. D. (2018). Logirunner: Um Jogo de Tabuleiro como Ferramenta para o Auxílio do Ensino e Aprendizagem de Algoritmos e Lógica de Programação. *RENOTE*, *16*(1), Article 1. https://doi.org/10.22456/1679-1916.85998

Castro, T. H., Fuks, H., Fernandes Sposito, M. A., & de Castro Junior, A. N. (2008). The analysis of a case study for group programming learning. In Diaz, P and Ignacio, A and Mora, E (Org.), *8TH IEEE International Conference On Advanced Learning Technologies, Proceedings* (p. 850+). IEEE Comp Soc; IEEE Tech Comm Learning Technol. https://doi.org/10.1109/ICALT.2008.99

Castro, T., Robertson, D., Fuks, H., & Castro, A. (2011). Identifying the Need to Intervene: Analysis and Representation of Interaction Patterns in Group Programming Learning. In Vivacqua, AS and Gutwin, C and Borges, MRS (Org.), *Collaboration And Technology* (Vol. 6969, p. 158+).

Chan, A. T. S., Cao, J., Liu, C.-K., & Cao, W. (2003). Design and Implementation of VPL: A Virtual Programming Laboratory for Online Distance Learning. In W. Zhou, P. Nicholson, B. Corbitt, & J. Fong (Orgs.), *Advances in Web-Based Learning—ICWL 2003* (p. 509–519). Springer. https://doi.org/10.1007/978-3-540-45200-3_47

Chang, K.-E., Chiao, B.-C., Chen, S.-W., & Hsiao, R.-S. (2000). A programming learning system for beginners—A completion strategy approach. *IEEE Transactions on Education*, *43*(2), 211–220. https://doi.org/10.1109/13.848075

Chaves, J. O., Castro, A., Lima, R., Lima, marcos V., & Ferreira, K. (2013a). Um Módulo de Integração com Juízes On-line para Auxiliar Atividades de Programação. *Revista de Informática Aplicada*, *9*(1). https://doi.org/10.13037/ria.vol9n1.2750

Chaves, J. O., Castro, A., Lima, R., Lima, M. V., & Ferreira, K. (2013b). MOJO: uma ferramenta para auxiliar o professor em disciplinas de programação. *Congresso Brasileiro de Ensino Superior a Distância, Belém, PA*.

Chen, G.-D., Li, L.-Y., & Wang, C.-Y. (2012). A Community of Practice Approach to Learning Programming. *Turkish Online Journal of Educational Technology-TOJET*, *11*(2), 15–26.

Chuchulashvili, M., Goziashvili, N., Pereira, M. J., & Lopes, R. P. (2016). Micro-atividades para a aprendizagem de programação. *VII Congresso Mundial de Estilos de Aprendizagem: livro de atas*. https://bibliotecadigital.ipb.pt/handle/10198/13690

Clark, R. C., Nguyen, F., & Sweller, J. (2011). *Efficiency in learning: Evidence-based guidelines to manage cognitive load*. John Wiley & Sons.

CNE/CES. (2012). *Diretrizes Curriculares Nacionais para os cursos de graduação em Computação*. Parecer CNE/CES Nº 136/2012.

Code::Blocks. ([s.d.]). Recuperado 27 de julho de 2021, de https://www.codeblocks.org/

CodeChef | Competitive Programming | Participate & Learn | CodeChef. ([s.d.]). Recuperado 27 de julho de 2021, de https://www.codechef.com/

Cook, D. D. (2015). Flowgorithm: Principles for teaching introductory programming using flowcharts. *Proc. American Society of Engineering Education Pacific Southwest Conf.(ASEE/PSW)*, 158–167.

Cooper, S., Dann, W., & Pausch, R. (2000). *Alice: a 3-d tool for introductory programming concepts*. 8.

Correção Automática de Exercícios de Programação—Run.codes: ([s.d.]). Recuperado 28 de julho de 2021, de https://we.run.codes/

Costa, J. M. (2019). Microworlds with Different Pedagogical Approaches in Introductory Programming Learning: Effects in Programming Knowledge and Logical Reasoning. *Informatica: an International Journal of Computing and Informatics*, *43*(1), Article 1. p. 145-174. https://doi.org/10.31449/inf.v43i1.2657

Coutinho, E., Bonates, M., & Moreira, L. O. (2018). Relato sobre o Uso de uma Ferramenta de Desenvolvimento de Jogos para o Ensino Introdutório de Lógica de Programação. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *7*(1), 689. https://doi.org/10.5753/cbie.wcbie.2018.689

Creswell, J. W. (2007). *Projeto de pesquisa-: Métodos qualitativo, quantitativo e misto*. Penso Editora.

de Almeida, E. S., Costa, E. de B., Silva, K. dos S., Paes, R. de B., Almeida, A. A. M., & Braga, J. D. H. (2002). AMBAP: um ambiente de apoio ao aprendizado de programação. *Anais do XXII Congresso da Sociedade Brasileira de Computaçao*, *4*, 79–88.

de Barros Paes, R., Malaquias, R., Guimarães, M., & Almeida, H. (2013). Ferramenta para a Avaliação de Aprendizado de Alunos em Programação de Computadores. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *2*(1).

de França, R. S., & do Amaral, H. J. C. (2012). Proposta de um Jogo Eletrônico Educativo Aplicado ao Ensino da Lógica de Programação*. Conferencias LACLO, 3(1)*.

de Souza, C. M. (2009). VisuAlg-Ferramenta de apoio ao ensino de programação. *Revista Eletrônica TECCEN*, *2*(2), 01–09.

de Souza, D. M., Maldonado, J. C., & Barbosa, E. F. (2011). ProgTest: An environment for the submission and evaluation of programming assignments based on testing activities. *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE T)*, 1–10. https://doi.org/10.1109/CSEET.2011.5876088

Denning, P. J., & Martell, C. H. (2015). *Great principles of computing*. MIT Press.

Díaz-Agudo, B., González-Calero, P. A., Recio-García, J. A., & Sánchez-Ruiz-Granados, A. A. (2007). Building CBR systems with jCOLIBRI. *Science of Computer Programming*, *69*(1–3), 68–75.

Dijkstra, E. W. (1989). On the cruelty of really teaching computing science. *Communications of the ACM*, *32*(12), 1398–1404.

dos Santos Júnior, G. P., Fechine, J. M., & de Barros Costa, E. (2009). Analogus: Um ambiente para auxılio ao ensino de programaçao orientado pelo raciocınio por analogia. *Anais do Workshop sobre Educação em Computação (WEI)*, 499–508.

Eltegani, N., & Butgereit, L. (2015). Attributes of Students Engagement in Fundamental Programming Learning. In Saeed, RA and Mokhtar, RA (Org.), *2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)* (p. 101–106). Neelain Univ; IEEE Sudan Subsect.

Esteves, A., Filho, A. H., & Raabe, A. (2019). Um Plugin para Aprendizagem de Lógica e Programação no Portugol Studio Baseado em Sistemas Adaptativos com Trilhas de Aprendizagem. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *8*(1), 1442. https://doi.org/10.5753/cbie.wcbie.2019.1442

Esteves, M., Antunes, R., Fonseca, B., Morgado, L., & Martins, P. (2008). Using Second Life in Programming's Communities of Practice. In R. O. Briggs, P. Antunes, G.-J. de Vreede, & A. S. Read (Orgs.), *Groupware: Design, Implementation, and Use* (p. 99–106). Springer. https://doi.org/10.1007/978-3-540-92831-7_9

Ettles, A., Luxton-Reilly, A., & Denny, P. (2018). Common logic errors made by novice programmers. *Proceedings of the 20th Australasian Computing Education Conference*, 83–89.

Farias, F. L. de O., & Nunes, I. (2019). Aprendizagem Ativa no Ensino de Programação: Uma Revisão Sistemática da Literatura. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *8*(1), 377. https://doi.org/10.5753/cbie.wcbie.2019.377

Farrel, J. (2010). *Lógica e design de programação: Introdução*. Cengage Learning.

Figueiredo, K. (2015). Proposta de Gamificação de Disciplinas em um Curso de Sistemas de Informação. *Anais do Simpósio Brasileiro de Sistemas de Informação (SBSI)*, 611–614. https://doi.org/10.5753/sbsi.2015.5892

Force, C. T. ([s.d.]). *Computing Curricula 2020: Paradigms for Global Computing Education November 2020*.

França, R., & Tedesco, P. (2015). Caracterizando a pesquisa sobre autoavaliação na aprendizagem de programação para iniciantes. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, *26*(1), 549. https://doi.org/10.5753/cbie.sbie.2015.549

Frantz, W., & Pontes, H. (2014). Um Ambiente de Desenvolvimento Personalizável para o Ensino de Programação. *Anais do XXII Workshop sobre Educação em Computação*, Porto Alegre: Sociedade Brasileira de Computação, 2014 . p. 337-344. ISSN 2595-6175.

Franzen, E., Hemming, C., & Bercht, M. (2018). Sistema de apoio ao uso da problematização no ensino e aprendizagem de programação. *Revista Destaques Acadêmicos*, *10*(4), Article 4. https://doi.org/10.22410/issn.2176-3070.v10i4a2018.2026

Fraser, N. (2015). Ten things we've learned from Blockly. *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, 49–50. https://doi.org/10.1109/BLOCKS.2015.7369000

Freire, P. (1987). *Pedagogia do Oprimido* (17◦ ed). Editora Paz e Terra.

Freitas, R. R. (2015). *Análise e Projeto de Software*. 60.

Galdino, C., Neto, S., & Costa, E. (2015). KidCoder: Uma Proposta de Ensino de Programação de forma Lúdica. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, *26*(1), 687. https://doi.org/10.5753/cbie.sbie.2015.687

Game Making Software—Construct 3. ([s.d.]). Recuperado 27 de julho de 2021, de https://www.construct.net

GameMaker para Faculdades e Universidades. ([s.d.]). YoYo Games. Recuperado 27 de julho de 2021, de https://www.yoyogames.com/pt-BR/education

Gaudencio, M., Wanderley, L. F., Lemos, F. W., Araújo, E. C. de, Figueiredo, J. C. A., & Guerrero, D. D. S. (2013). Eu Sei o que Vocês Fizeram (Agora e) na Aula Passada: O TSTView no Acompanhamento de Exercícios de Programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, *24*(1), 204. https://doi.org/10.5753/cbie.sbie.2013.204

Gerdes, A., Heeren, B., Jeuring, J., & van Binsbergen, L. T. (2017). Ask-Elle: An adaptable programming tutor for Haskell giving automated feedback. *International Journal of Artificial Intelligence in Education*, *27*(1), 65–100. https://doi.org/10.1007/s40593-015-0080-x.

Giraffa, L. M., & Mora, M. da C. ([s.d.]). Evasão na disciplina de algoritmo e programação: um estudo a partir dos fatores intervenientes na perspectiva do aluno. *III Conferência Latino-Americana sobre el abandono en la Educacion Superior*, 10. ISBN 9788415302711.

Glinert, E. P., & Tanimoto, S. L. (1984). Pict: An interactive graphical programming environment. *Computer*, *17*(11), 7–25.

Gomes, A. de J. (2010, dezembro 15). Dificuldades de aprendizagem de programação de computadores: Contributos para a sua compreensão e resolução. [Tese de Doutoramento]*. Departamento de Engenharia Informática. Universidade de Coimbra*. https://estudogeral.sib.uc.pt/handle/10316/14586

Gomes, A. J., Henriques, J., & Mendes, A. J. (2008). Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *EFT: Educação, Formação & Tecnologias*, *1*(1), 93–103.

Gomes, A., & Mendes, A. (2015). A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. *Proceedings - Frontiers in Education Conference, FIE*, *2015-February*. https://doi.org/10.1109/FIE.2014.7044086

Gomes, A., & Mendes, A. J. (2010). Studies and Proposals about Initial Programming Learning. *2010 IEEE Frontiers In Education Conference (FIE).* DOI: 10.1109/FIE.2010.5673426.

Gomes, A., & Mendes, A. J. (2007). Learning to program-difficulties and solutions. *International Conference on Engineering Education–ICEE*, *7*.

Gomes, M. A. C. N., Soto, M. N. C., & Núñez, J. A. L. (2019). Factores que dificultam a transição dos alunos do segundo ciclo do ensino secundário para o ensino superior. *Innoeduca. International Journal of Technology and Educational Innovation*, *5*(1), 91–101.

Gomes, M., D'Emery, R., & Cysneiros, G. (2014). AAPW: Uma ferramenta para facilitar o aprendizado de programação Web. *Anais do Workshop sobre Educação em Computação (WEI)*, 269–278. https://sol.sbc.org.br/index.php/wei/article/view/10981

Grillo, M. (2008). Percursos da constituição da docência. *A docência na Educação Superior: sete olhares*, *2*, 65–79.

Guimarães, S. É. R., & Boruchovitch, E. (2004). O estilo motivacional do professor e a motivação intrínseca dos estudantes: Uma perspectiva da teoria da autodeterminação. *Psicologia: reflexão e crítica*, *17*(2), 143–150.

Hansen, S. R., Narayanan, N. H., & Schrimpsher, D. (2000). Helping learners visualize and comprehend algorithms. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, *2*(1), 10.

Hartness, K. (2004). Robocode: Using games to teach artificial intelligence. *Journal of Computing Sciences in Colleges*, *19*(4), 287–291.

Harvey, B., Garcia, D. D., Barnes, T., Titterton, N., Armendariz, D., Segars, L., Lemon, E., Morris, S., & Paley, J. (2013). SNAP! (build your own blocks) (abstract only). *Proceeding of the 44th ACM technical symposium on Computer science education*, 759. https://doi.org/10.1145/2445196.2445507

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, *13*(3), 259–290.

Hwang, W.-Y., Shadiev, R., Wang, C.-Y., & Huang, Z.-H. (2012). A Study of Cooperative Computer Programming Learning Behavior and its Influence on Learning Performance. In Lam, P (Org.), *Proceedings Of The 7th International Conference on Elearning* (p. 150–157).

Iepsen, E. F., Bercht, M., & Reategui, E. B. (2013). Avaliando a Dimensão Afetiva para Apoio ao Processo de Aprendizagem na Disciplina de Algoritmos: Um Estudo de Caso / Evaluating the Affective Dimension to Support the Learning Process in the Discipline of Algorithms: a Case Study. *Revista Latinoamericana de Tecnología Educativa - RELATEC*, *12*(2), 55–66.

Ingalls, D., Kaehler, T., Maloney, J., Wallace, S., & Kay, A. (1997). Back to the future: The story of Squeak, a practical Smalltalk written in itself. *Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, 318–326. https://doi.org/10.1145/263700.263754.

Jenkins, T. (2002). On the Difficulty of Learning to Program. *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 53–58. http://www.ics.ltsn.ac.uk//pub/conf2002/jenkins.html

Jenkins, T. (2001). The motivation of students of programming. *Proceedings of the 6th annual conference on Innovation and technology in computer science education*, 53–56. https://doi.org/10.1145/377435.377472

Jesus, E. A. de. (2010). Avaliação Empírica da Utilização de um Jogo para Auxiliar a Aprendizagem de Programação. *Tese de Mestrado. Departamento de Computação Aplicada. Universidade do Vale do Itajaí.* 152 pp..

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. Org. *Computers in Human Behavior*, *52*, 200–210.

Kamiya, R. R., & Brandão, L. O. (2009). IVProg - um sistema para introdução à Programação através de um modelo Visual na Internet. *Anais do XX Simpósio Brasileiro de Informática na Educação. Florianópolis, SC*.

Kawaguchi, S., Nishikawa, T., Sato, Y., Onuma, R., Nakayama, H., Nakamura, S., & Miyadera, Y. (2019). Development of a Training Data Creation Support Environment for Estimating Programming Learning Situations. *2019 18th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 1–6. https://doi.org/10.1109/ITHET46829.2019.8937339

Kay, A., Rose, K., Ingalls, D., Kaehler, T., Maloney, J., Wallace, S., & others. (1997). Etoys & SimStories. *ImagiLearning Internal Document*.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences*, *47*, 1991–1999.

Khouri, C. M. B., Santos, G. N. dos, & Barbosa, M. S. S. (2020). Mapeamento Sistemático em Metodologias de Ensino-aprendizagem de Programação. *Revista de Ciência da Computação*, *2*(1), 13–27. https://doi.org/10.22481/recic.v2i1.6669

Klock, A. C. T., Gasparini, I., & Pimenta, M. S. (2016). 5W2H Framework: A guide to design, develop and evaluate the user-centered gamification. *Proceedings of the 15th Brazilian Symposium on Human Factors in Computing Systems*, 1–10.

Kodu Game Lab. ([s.d.]). KoduGameLab. Recuperado 27 de julho de 2021, de http://www.kodugamelab.com/

Koliver, C., Dorneles, R. V., & Casa, M. E. (2004). Das (muitas) dúvidas e (poucas) certezas do ensino de algoritmos. *XII Workshop de Educação em Computação.* 24, Salvador: UFBA.

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ System and its Pedagogy. *Computer Science Education*, *13*(4), 249–268. https://doi.org/10.1076/csed.13.4.249.17496

Kumar, B., & Khurana, P. (2012). Gamification in education - learn computer programming with fun. *Vol. No.*, *1*, 9.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *Acm sigcse bulletin*, *37*(3), 14–18.

Law, K. M. Y., Lee, V. C. S., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, *55*(1), 218–228. https://doi.org/10.1016/j.compedu.2010.01.007

Leite, J. C. S., Yu, Y., Liu, L., Eric, S., & Mylopoulos, J. (2005). Quality-based software reuse. *International Conference on Advanced Information Systems Engineering*, 535–550.

Lemos, M. (2004). *Uma Abordagem Baseada em Padrões Elementares para Aprendizado de Programação.* [Tese de Doutoramento]. Universidade de São Paulo, São Paulo.

LightBot. ([s.d.]). Recuperado 27 de julho de 2021, de https://lightbot.com/

Lopes, M. C., & Kopsch, H. K. (2018). Furbot-web: Umas plataforma adaptativa para o ensino de programação. *Revista Tecnologias na Educação (TECEDU)*, *10*, 25.

Magnuson, B. (2010). *Building blocks for mobile games: A multiplayer framework for App inventor for Android* [Tese de Doutoramento]. Massachusetts Institute of Technology.

Manso, A., Oliveira, L., & Marques, C. (2009). Portugol IDE–Uma ferramenta para o ensino de programação. *PAEE'2009–Project Approaches in Engineering Education–Guimaraes*.

Marcolino, A., & Barbosa, E. F. (2015). Softwares Educacionais para o Ensino de Programação: Um Mapeamento Sistemático. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, *26*(1), 190. https://doi.org/10.5753/cbie.sbie.2015.190

Marques, D. L., Costa, L. F. S., de Azevedo Silva, M. A., & Rebouças, A. D. D. S. (2011). Atraindo alunos do ensino médio para a computação: Uma Experiência Prática de Introdução à Programação utilizando Jogos e Python. *Anais do Workshop de Informática na Escola*, *1*(1), 1138–1147.

McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., & Mander, K. (2005). Grand challenges in computing: Education–A summary. *The Computer Journal*, *48*(1), 42–48.

Medeiros, R. P. (2019). *Hello, world: Uma análise sobre dificuldades no ensino e na aprendizagem de introdução à programação nas universidades.* [Tese de Doutoramento]. Programa de Pós-Graduação em Ciência da Computação. Universidade de Pernamuco.

Medeiros, R. P., Falcão, T. P., & Ramalho, G. L. (2020). Ensino e Aprendizagem de Introdução à Programação no Ensino Superior Brasileiro: Revisão Sistemática da Literatura. *Anais do Workshop sobre Educação em Computação (WEI)*, 186–190. https://doi.org/10.5753/wei.2020.11155

Meirinhos, M. F. A., & Osório, A. J. (2011). *O advento da escola como organização que aprende: A relevância das TIC.* Conferência Ibérica: Inovação na Educação com TIC.

Mendes, A. J., Paquete, L., Cardoso, A., & Gomes, A. (2012). Increasing student commitment in introductory programming learning. *2012 Frontiers in Education Conference (FIE).* DOI: 10.1109/FIE.2012.6462486.

Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative research: A guide to design and implementation*. John Wiley & Sons.

Monster Coding—Fun Programming for Kids. ([s.d.]). Recuperado 27 de julho de 2021, de http://monstercoding.com/#pt

Morais, C. G. B. (2010). *Cognitio: Um processo para reuso de requisitos* [Dissertação de Mestrado].Mestrado em Ciência da Computação. Universidade do Estado do Rio Grande do Norte. Mossoró, Brasil. http://www.dominiopublico.gov.br/pesquisa/DetalheObraForm.do?select_action=&co_obra=184485

Morais, C. G. B., Mendes Neto, F. M., & Osório, A. J. M. (2020). Dificuldades e desafios do processo de aprendizagem de algoritmos e programação no ensino superior: Uma revisão sistemática de literatura. *Research, Society and Development*, *9*(10), e9429109287. https://doi.org/10.33448/rsd-v9i10.9287

Morais, C. G. B., Osório, A. J. M., & Neto, F. M. M. (2019). O processo de ensino e aprendizagem de programação no nível superior: Uma revisão sistemática de literatura. *XV Congreso Internacional Gallegoportugués de Psicopedagogía. II Congreso de la Asociación Científica Internacional de Psicopedagogía: (A Coruña, 4-6 de septiembre de 2019), 2019, ISBN 978-84-9749-726-8, págs. 3984-3996*, 3984–3996. https://dialnet.unirioja.es/servlet/articulo?codigo=7360866

Moreira, G. L., Holanda, W., Coutinho, J. C. da S., & Chagas, F. S. (2018). Desafios na aprendizagem de programação introdutória em cursos de TI da UFERSA, campus Pau dos Ferros: Um estudo exploratório. *Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA (ISSN 2526-7574)*, *2*, Article 2. https://periodicos.ufersa.edu.br/index.php/ecop/article/view/7907

Natal, M. E. C., Barbosa, B. A., Hernandes, J. C., de Sousa Much, B., Bigolin, M., da Silva, S. J. R., Silva, C. B., & de Carvalho, L. F. B. (2018). Tri-Logic: Um Ambiente Gamificado como Ferramenta de Auxílio ao ensino de aprendizagem de Lógica de Programação. *RENOTE*, *16*(2), 41–50. https://doi.org/10.22456/1679-1916.89298.

Nawahdah, M., Taji, D., & Inoue, T. (2015). Collaboration leads to success: A study of the effects of using pair-programming teaching technique on student performance in a Middle Eastern society. *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 16–22. https://doi.org/10.1109/TALE.2015.7386009

Neto, W. C. B., & Schuvartz, A. A. (2007). Ferramenta Computacional de Apoio ao Processo de Ensino-Aprendizagem dos Fundamentos de Programação de Computadores. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, *1*(1), 520–528. https://doi.org/10.5753/cbie.sbie.2007.520-528

Netto, D., Medeiros, L. M., Pontes, D. de, & Morais, E. de. (2017, julho 6). Game Logic: Um jogo para auxiliar na aprendizagem de lógica de programação. *Anais Do Workshop Sobre Educação Em Computação (WEI)*. Anais do XXV Workshop sobre Educação em Computação. https://doi.org/10.5753/wei.2017.3546

Nicácio, J. M., & Costa, F. P. D. da. (2018). Dekstra: Um Ambiente de Aprendizagem Social para iniciação à aprendizagem de programação usando esquemas de concepção e planos programação. *Informática na educação: teoria & prática*, *21*(2 Mai/Ago), Article 2 Mai/Ago. https://doi.org/10.22456/1982-1654.77611

Nishida, A., Ferreira, V., & Braga, J. C. (2017). Processs Legend Jogo de Enigmas para o Ensino de Introdução à Programação. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *6*(1), 200. https://doi.org/10.5753/cbie.wcbie.2017.200

Nobre, I. A. M., & de Menezes, C. S. (2002). Suporte à Cooperação em um Ambiente de aprendizagem para Programação (SAmbA). *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, *1*(1), 337–347.

Noschang, L. F., Pelz, F., de Jesus, E., & Raabe, A. (2014). Portugol studio: Uma ide para iniciantes em programaçao. *Anais do XXII Workshop sobre Educação em Computação*, 1–10. ISSN 2595-6175.

O'Kelly, J., & Gibson, J. P. (2006). RoboCode & problem-based learning: A non-prescriptive approach to teaching programming. *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education*, 217–221. DOI:10.1145/1140124.1140182.

Oliveira, A. S., Mota, L. da C., & Oliveira, A. A. de. (2015). Uso de ambientes virtuais de aprendizagem como suporte ao ensino de programação: Uma revisão sistemática. *Interfaces Científicas - Exatas e Tecnológicas*, *1*(3), 9–22. https://doi.org/10.17564/2359-4942.2015v1n3p9-22

Oliveira, J. P. de. (2020, março 20). *A formação de estudantes do Ensino Médio integrado no Brasil: Contributos para os estudos sobre programas de extensão em Institutos Federais* [Tese de Doutoramento]. Universidade de Évora. http://dspace.uevora.pt/rdpc/handle/10174/27838

Oliveira, T. A. N. de, & Rebouças, A. D. (2018). The Use of Pair Programming to Support Introductory Programming Teaching: A Qualitative Study. *2018 XIII Latin American Conference on Learning Technologies (LACLO)*, 65–68. https://doi.org/10.1109/LACLO.2018.00025

Oliveira, M., & Oliveira, E. (2014). Metodologia de Diagnóstico e Regulação de Componentes de Habilidades da Aprendizagem de Programação. *Anais do XXII Workshop sobre Educação em Computação*, 120–129. ISSN 2595-6175.

Ortiz-Ortiz, O., Jimenez-Murillo, J. A., & Jimenez-Hernandez, E. M. (2018). A Web Framework to Improve Computer Programming Learning. *2018 IEEE International Autumn Meeting On Power, Electronics And Computing (ROPEC).* DOI:10.1109/ROPEC.2018.8661409 .

Pacitti, T., Aktinson, C. P., & Teles, A. A. d. S. (1983). *Programação e Métodos Computacionais*. LTC.

Paillard, G. A. L., & Moreira, L. O. (2017). O impacto dos paradigmas e linguagens de programaçao no ensino intermediário da programaçao de computadores. *Revista Tecnologias na Educação*, *19*(9), 1–13.

Panegalli, F. S., Bernardi, G., & Cordenonsi, A. Z. (2019). Super Mario Logic: Um jogo sério para auxiliar no processo de ensino e aprendizagem de lógica de programação. *RENOTE*, *17*(1), 244–253. https://doi.org/10.22456/1679-1916.95788

Papert, S. (2008). Logo: Computadores e educação. São Paulo: Brasiliense, 1980. *A Máquina das Crianças. Porto Alegre-RS: Artmed*.

Paredes, Y. V., Huang, P.-K., & Hsiao, I.-H. (2019). Utilising behavioural analytics in a blended programming learning environment. *New Review of Hypermedia and Multimedia*, *25*(3), 89–111. https://doi.org/10.1080/13614568.2019.1695961

Paula, L. de, Piva Jr, D., & Freitas, R. L. (2009). A Importância da Leitura e da Abstração do Problema no processo de formação do raciocínio lógico-abstrato em alunos de Computação. *XVII Workshop sobre Educação em Computação-WEI*.

Pea, R. D., & Kurland, D. M. (1983). *On the Cognitive Prerequisites of Learning Computer Programming. Technical Report No. 18*. https://eric.ed.gov/?id=ED249931

Pedrosa, D. C. C. (2017). Autorregulação e corregulação das aprendizagens no ensino superior: Estratégias adotadas por alunos de programação de computadores. [Tese de Doutoramento]*, Didática de Ciências e Tecnologias, especialização em Didática de Informática, Universidade de Trás-os-Montes e Alto Douro*.

Pereira, R., Costa, C. J., & Aparicio, J. T. (2017). Gamification to support programming learning. *2017 12th Iberian Conference on Information Systems and Technologies (CISTI).* DOI: 10.23919/CISTI.2017.7975788.

Pereira, R. M. S. (2017). *Gamificação como solução para os problemas da aprendizagem da programação.* [Dissertação de Mestrado]. *Instituto Universitário de Lisboa.* Discponível em: https://repositorio.iscte-iul.pt/handle/10071/15264

Perkins, D. N., & Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. *Papers presented at the First Workshop On Empirical Studies Of Programmers On Empirical Studies Of Programmers*, 213–229.

Píccolo, H. L., Sena, V. de F., Nogueira, K. B., da Silva, M. O., & Maia, Y. A. (2010). Ambiente Interativo e Adaptável para ensino de Programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, *1*(1).

Pimenta, S. G., & Anastasiou, L. das G. C. (2002). *Docência no ensino superior* (Vol. 1). Editora Cortez. São Paulo.

Pinho, M. J. de. (2017). Ciência e ensino: Contribuições da iniciação científica na educação superior. *Avaliação: Revista da Avaliação da Educação Superior (Campinas)*, *22*(3), 658–675. https://doi.org/10.1590/s1414-40772017000300005

Pio, J. L., Castro, T. H. C., & Castro Júnior, A. N. (2006). A robótica móvel como instrumento de apoio à aprendizagem de computação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, *1*(1), 497–506.

Piteira, M., Costa, C. J., & Aparicio, M. (2017). A conceptual framework to implement gamification on online courses of computer programming learning: implementation. In Chova, LG and Martinez, AL and Torres, IC (Org.), *10TH International Conference of Education, Research and Innovation (ICERI2017)* (p. 7022–7031). IATED-INT Assoc Technology Education & Development.

Piva Jr, D., & Freitas, R. L. (2010). Estratégias para melhorar os processos de abstração na disciplina de Algoritmos. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, *1*(1).

PPC. (2019). *Projeto Pedagógico do Curso de Ciência da Computação*. http://fanat2.uern.br/di/index.php/ppc

Prietch, S. S., & Pazeto, T. A. (2010). Mapeamento de cursos de licenciatura em Computação seguido de proposta de padronização de matriz curricular. *XVIII Workshop de Educação em Computação (WEI 2010), Anais do XXX Congresso da Sociedade Brasileira de Computação-CSBC*, 921–930.

Programming For Kids | AgentCubes. ([s.d.]). Recuperado 28 de julho de 2021, de https://agentsheets.com/

Proulx, V. K. (2000). Programming patterns and design patterns in the introductory computer science course. *ACM Sigcse Bulletin*, *32*(1), 80–84.

PyGame. ([s.d.]). Recuperado 28 de julho de 2021, de https://www.pygame.org/news

Raabe, A. L. A., & Silva, J. da. (2005). Um ambiente para atendimento as dificuldades de aprendizagem de algoritmos. *XIII Workshop de Educação em Computação (WEI'2005). São Leopoldo, RS, Brasil*, *3*, 5.

Raabe, A., Zanchett, G., & Vahldick, A. (2015). Jogos de Programar como uma Abordagem para os Primeiros Contatos dos Estudantes com à Programação. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *4*(1), 1485.

Rapkiewicz, C. E., Falkembach, G., Seixas, L., Rosa, N. dos S., Cunha, V. V. da, & Klemann, M. (2006). Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. *RENOTE*, *4*(2), Article 2. https://doi.org/10.22456/1679-1916.14284

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & others. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67.

Rezende, C. M. C., & Bispo, E. L. (2018). Comparison between the use of pseudocode and visual programming in programming teaching: An evaluation from scratch tool [Comparação entre o Uso de Pseudocódigo e a Programação Visual no Ensino de Programação: Uma avaliação a partir da ferramenta Scratch]. *Iberian Conference on Information Systems and Technologies, CISTI*, *2018-June*, 1–5. https://doi.org/10.23919/CISTI.2018.8399305

Ribeiro, F., Merlin, B., & Fülber, H. (2019). Avaliação do impacto de ambientes gamificados no processo de ensino-aprendizagem de programação de computadores: Uma comparação entre elementos monousuário e multiusuário. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, *30*(1), 803. https://doi.org/10.5753/cbie.sbie.2019.803

Richardson, R. J. (2017). *Pesquisa social: Métodos e técnicas*. Editora Atlas. São Paulo.

RoboMind.net—Welcome to RoboMind.net, the new way to learn programming. ([s.d.]). Recuperado 27 de julho de 2021, de https://www.robomind.net/en/index.html

Rocha, P. S., Ferreira, B., Monteiro, D., Nunes, D. da S. C., & Góes, H. C. do N. (2010). Ensino e Aprendizagem de Programação: Análise da Aplicação de Proposta Metodológica Baseada no Sistema Personalizado de Ensino. *RENOTE*, *8*(3), Article 3. https://doi.org/10.22456/1679-1916.18061

Rum, S. N. M., & Ismail, M. A. (2017). Metocognitive support accelerates computer assisted learning for novice programmers. *Journal of Educational Technology & Society*, *20*(3), 170–181.

Sales, C. G., & Dantas, V. F. (2010). ProGame: Um jogo para o ensino de algoritmos e programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, *1*(1), Article 1. https://doi.org/10.5753/cbie.sbie.2010.%p

Santos, A., Gomes, A., & Mendes, A. (2011). A Class Record and Reviewing System Designed to Promote Programming Learning. *2011 Frontiers in Education Conference (FIE).* DOI: 10.1109/FIE.2011.6142966.

Santos, A., Gomes, A., & Mendes, A. (2013). A taxonomy of exercises to support individual learning paths in initial programming learning. *2013 IEEE Frontiers in Education Conference.* DOI: 10.1109/FIE.2013.6684794.

Santos, A., Gomes, A., & Mendes, A. J. (2010). Integrating New Technologies and Existing Tools to Promote Programming Learning. *ALGORITHMS*, *3*(2), 183–196. https://doi.org/10.3390/a3020183

Santos, A., Gorgônio, A., Lucena, A., & Gorgônio, F. (2015). A Importância do Fator Motivacional no Processo Ensino-Aprendizagem de Algoritmos e Lógica de Programação para Alunos Repetentes. *Anais do Workshop sobre Educação em Computação (WEI)*, 168–177. https://doi.org/10.5753/wei.2015.10233

Santos, G., Khouri, C., & Barbosa, M. (2020). Mapeamento Sistemático em Metodologias de Ensino-aprendizagem de Programação. *Revista de Ciencia Política*, *2*, 13–27. https://doi.org/10.22481/recic.v2i1.6669

Santos, R., & Menezes, C. de. (2019). Ambiente para Aprendizagem de Programação com apoio Dialogado por Assistentes Inteligentes. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, *30*(1), 1511. https://doi.org/10.5753/cbie.sbie.2019.1511

Shadiev, R., Hwang, W.-Y., Yeh, S.-C., Yang, S. J. H., Wang, J.-L., Han, L., Huang, Y.-M., & Liu, C.-J. (2013). Applying unidirectional versus reciprocal teaching strategies in web-based environment and their effects on computer programming learning. *Proceedings - 2013 IEEE 13th International Conference on Advanced Learning Technologies, ICALT 2013*, 5–9. https://doi.org/10.1109/ICALT.2013.7

Silva, B. S. da, & Trentin, M. A. S. (2016). Dificuldades no ensino-aprendizagem de programação de computadores: Contribuições para sua compreensão e resolução. *V Simpósio Nacional de Ensino de Ciência e Tecnologia, 5º edição, Universidade Tecnológica Federal do Paraná–Campus Ponta Grossa. Anais do V SINECT. UTFPR*.

Silva, E. O. da, & Falcão, T. P. (2020). O Pensamento Computacional no Ensino Superior e seu Impacto na Aprendizagem de Programação. *Anais do Workshop sobre Educação em Computação (WEI)*, 171–175. https://doi.org/10.5753/wei.2020.11152

Silva, T. R. da, Medeiros, T., Medeiros, H., Lopes, R., & Aranha, E. (2015). Ensino-aprendizagem de programação: Uma revisão sistemática da literatura. *Revista Brasileira de Informática na Educação*, *23*(01), 182. https://doi.org/10.5753/rbie.2015.23.01.182

Silva, F. N. e, & Borges, M. (2016). PBL e robótica no ensino de conceitos de Lógica de Programação. *Anais do Workshop sobre Educação em Computação (WEI)*, 298–307. https://doi.org/10.5753/wei.2016.9673

Silva, I., Silva, I. M., & Santos, M. S. (2009). Análise de problemas e soluções aplicadas ao ensino de disciplinas introdutórias de programação. *Universidade Federal Rural de Pernambuco, Recife–PE*.

Silva, J., Oliveira, F., & Martins, D. (2018). Gamificação e storytelling como estratégia motivacional no ensino de programação. *Proceedings of SBGames*.

Silveira, S. R., Pereira, A. S., Bertolini, C., Parreira, F., & Bigolin, N. (2018). Educação a Distância, Sala de Aula Invertida e Aprendizagem Baseada em Problemas: Possibilidades para o ensino de programação de computadores. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *7*(1), 1052. https://doi.org/10.5753/cbie.wcbie.2018.1052

Sirotheau, S., Brito, S. R. de, Silva, A. do S. da, Eliasquevici, M. K., Favero, E. L., & Tavares, O. de L. (2012). Aprendizagem de iniciantes em algoritmos e programação: Foco nas competências de autoavaliação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, *1*(1), Article 1. https://doi.org/10.5753/cbie.sbie.2011.%p

Sirotheau, S., Favero, E., Santos, J., & Balieiro, R. F. (2018). LabPy: Laboratório virtual de ensino em Python. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, *7*(1), 749. https://doi.org/10.5753/cbie.wcbie.2018.749

Skalka, J., & Drlik, M. (2018). Priscilla—Proposal of System Architecture for Programming Learning and Teaching Environment. *2018 IEEE 12TH INTERNATIONAL CONFERENCE ON APPLICATION OF INFORMATION AND COMMUNICATION TECHNOLOGIES (AICT)*, 414–419.

Souleiman, A. H. (2017). Orchestration and Adaptation of Learning Scenarios: Application to the Case of Programming Learning / Teaching. *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 7–11. https://doi.org/10.1109/AICCSA.2017.185

Stake, R. (2012). A arte da investigação com estudos de caso. *3ª ed. Lisboa: Fundação Calouste Gulbenkian*.

Stephan, J., Oliveira, A., & Renhe, M. C. (2020). O Uso de Jogos para Apoiar o Ensino e Aprendizagem de Programação. *Anais do Simpósio Brasileiro de Informática na Educação*, 381–390. https://doi.org/10.5753/cbie.sbie.2020.381

Tavares, P. C. (2018). *O impacto da animação e da avaliação automática na motivação para o ensino da programação*. [Tese de Doutoramento]. Escola de Engenharia. Universidade do Minho. Braga, Portugal. http://repositorium.sdum.uminho.pt/

Using, S. N. M., Ahmad, R., & Taib, S. Mohd. (2010). Ontology of programming resources for semantic searching of programming related materials on the Web. *2010 International Symposium on Information Technology*, *2*, 698–703. https://doi.org/10.1109/ITSIM.2010.5561537

Vasconcellos, I. L. B., Tamariz, A. D. R., & Batista, S. C. F. (2019). Planejamento, desenvolvimento e avaliação de um ambiente virtual de aprendizagem gamificado. *RENOTE*, *17*(1), 21–30. https://doi.org/10.22456/1679-1916.95663

Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P., & Queirós, R. (2012). A distributed system for learning programming on-line. *Computers & Education*, *58*(1), 1–10. https://doi.org/10.1016/j.compedu.2011.08.015

Villalobos, J. A., Calderon, N. A., & Jiménez, C. H. (2009). Developing programming skills by using interactive learning objects. *ACM SIGCSE Bulletin*, *41*(3), 151–155. https://doi.org/10.1145/1595496.1562927

Visual Logic. ([s.d.]). Recuperado 28 de julho de 2021, de https://www.visuallogic.org/

von Wangenheim, C. G., Nunes, V. R., & dos Santos, G. (2014). Teaching Computing with SCRATCH in Elementary Schools–A Case Study'. *Brazilian Journal of Computers in Education*, *22*(03), 115.

Vygotsky, L. S., & Cole, M. (1978). *Mind in society: Development of higher psychological processes*. Harvard University Press.

Whittall, S. J., Prashandi, W. A. C., Himasha, G. L. S., De Silva, D. I., & Suriyawansa, T. K. (2017). CodeMage: Educational programming environment for beginners. *2017 9th International Conference on Knowledge and Smart Technology (KST)*, 311–316. https://doi.org/10.1109/KST.2017.7886101

Wilson, C. (2014). Hour of code: We can solve the diversity problem in computer science. *ACM Inroads*, *5*(4), 22–22.

Wing, J. M. (2014). Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, *2014*, 26.

Wood, C., Mentzelopoulos, M., & Protopsaltis, A. (2015). EdCCDroid: An Education Pilot Prototype for Introducing Code-Combat using LUA. *Intelligent Environments (Workshops)*, 353–360.

Yin, R. K. (2015). *Estudo de Caso: Planejamento e métodos*. Bookman editora.

Zanini, A. S., & Raabe, A. L. A. (2012). Análise dos enunciados utilizados nos problemas de programação introdutória em cursos de Ciência da Computação no Brasil. *Anais do XXXII Congresso da Sociedade Brasileira de Computação, XX WEI–Workshop sobre Educação em Computação*.

APÊNDICES

Apêndice A - Parecer Consubstanciado do CEP-UERN

UERN - UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: O processo de aprendizagem de Algoritmos e Programação no nível superior de ensino.
Pesquisador: CERES GERMANNA BRAGA MORAIS
Área Temática:
Versão: 2
CAAE: 34035520.3.0000.5294
Instituição Proponente: UERN
Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 4.183.849

Apresentação do Projeto:
O projeto avaliado trata-se de uma tese de doutorado do programa de doutoramento em Ciências da Educação, do Instituto de Educação da Universidade do Minho. A aprendizagem de programação é conhecida por ser difícil para muitos alunos, tendo sido pesquisada ao longo dos anos. É necessário apoiá-los a aprender de forma eficaz, de forma que as atividades de aprendizagem possam ser adaptadas ao ritmo de aprendizagem do aluno e às suas necessidades específicas. Este projeto integra-se a esta problemática para compreender o processo de aprendizagem de programação no nível superior, buscando verificar quais são as metodologias existentes e de que forma são postas em prática, quais as principais dificuldades enfrentadas pelos alunos e quais competências necessárias aos alunos para que possam ter êxito na aprendizagem de programação. Os resultados obtidos deverão permitir construir novos conhecimentos sobre como proporcionar uma melhor experiência para a aprendizagem, especialmente com apoio das tecnologias educativas, e contribuir para a definição de uma proposta de inovação pedagógica capaz de responder às exigências curriculares atuais e ao perfil dos alunos.

Objetivo da Pesquisa:
Objetivo Primário:
A finalidade do projeto é desenvolver uma metodologia de aprendizagem de algoritmos e

Endereço: Rua Miguel Antonio da Silva Neto, s/n
Bairro: Aeroporto **CEP:** 59.607-360
UF: RN **Município:** MOSSORO
Telefone: (84)3312-7032 **E-mail:** cep@uern.br

programação, para alunos do ensino superior, com o intuito maximizar o êxito dos alunos nessas disciplinas.
Objetivo Secundário:
1. Inventariar as metodologias, ferramentas e práticas de ensino de algoritmos e programação no nível superior;
2. Caracterizar os fatores que podem influenciar para que um aluno aprenda a programar;
3. Explicitar as competências, conhecimentos e aptidões necessárias ao aluno para que este aprenda a programar;
4. Identificar as principais dificuldades enfrentadas pelos alunos durante a aprendizagem de programação, e o que estas dificuldades podem acarretar durante o percurso acadêmico;
5. Conceber uma metodologia que apoie o processo de aprendizagem de algoritmos e programação no nível superior, e desenvolver uma Investigação-Ação com a finalidade de avaliar e validar a metodologia proposta.

Avaliação dos Riscos e Benefícios:
Os riscos e benefícios inerentes à pesquisa foram avaliados. Sendo esses,
Riscos:
Para os professores participantes, existem riscos de constrangimento, e ainda alegarem falta de tempo e disponibilidade para participarem da pesquisa. Para que tais riscos sejam amenizados, as etapas e metodologias da pesquisa serão esclarecidas aos docentes. Os riscos para os alunos serão mínimos, já que a pesquisa será realizada por meio eletrônico, podendo responder às questões em horários que não esteja em aula ou no
trabalho, além disso é garantido o anonimato, não tem caráter avaliativo e os mesmos participam por adesão, e não por imposição. Durante a Intervenção-Ação pode haver risco de constrangimento por haver outro professor (no caso a pesquisadora) observando suas atitudes em sala de aula.
Benefícios:
Os benefícios da pesquisa para os docentes se concentram na possibilidade de ampliar as possibilidades metodológicas e de materiais para utilização em sala de aula, favorecendo o processo de aprendizagem dos alunos. Os benefícios para os discentes estão na possibilidade de aprender de modo mais dinâmico, tendo acesso a uma nova metodologia de aprendizagem que poderá maximizar sua aprendizagem de algoritmos e programação.

Comentários e Considerações sobre a Pesquisa:

O protocolo de pesquisa avaliado apresenta relevância e exequibilidade.

Considerações sobre os Termos de apresentação obrigatória:

Todos os termos de apresentação obrigatória encontram-se anexados.

Conclusões ou Pendências e Lista de Inadequações:

A pesquisa não apresenta óbices éticos. O presente protocolo de pesquisa encontra-se de acordo com as normativas éticas vigentes.

Considerações Finais a critério do CEP:

Considerando a Declaração de Emergência em Saúde Pública de Importância Internacional pela Organização Mundial de Saúde (OMS), em 30 de janeiro de 2020, em decorrência da Doença por Coronavírus – COVID-19 (decorrente do SARS-CoV-2, novo Coronavírus);

Considerando a forma de priorizar a saúde da comunidade com o distanciamento social, conforme determinado por cada Chefe do Executivo Estadual;

O Comitê de Ética em Pesquisa com Seres Humanos (CEP) da Universidade do Estado do Rio Grande do Norte recomenda que as particularidades relacionadas a proteção da saúde de todos os envolvidos nos protocolos de pesquisa sejam observadas e que os decretos e resoluções pertinentes a realidade de cada Instituição Proponente, bem como das instituições anuentes, sejam respeitadas. Por fim, recomendamos que caso sua pesquisa passe por alterações em decorrência dessa paralisação uma emenda deve ser enviada ao CEP para apreciação das mesmas.

Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Informações Básicas do Projeto	PB_INFORMAÇÕES_BÁSICAS_DO_PROJETO_1573219.pdf	13/07/2020 19:53:08		Aceito
Outros	Carta_CEP_CeresMorais.pdf	13/07/2020 19:49:27	CERES GERMANNA BRAGA MORAIS	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	TCLE_alunos_alterado.pdf	13/07/2020 19:47:40	CERES GERMANNA BRAGA MORAIS	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	TCLE_professores_alterado.pdf	13/07/2020 19:47:02	CERES GERMANNA BRAGA MORAIS	Aceito

Endereço: Rua Miguel Antonio da Silva Neto, s/n
Bairro: Aeroporto **CEP:** 59.607-360
UF: RN **Município:** MOSSORO
Telefone: (84)3312-7032 **E-mail:** cep@uern.br

Declaração de concordância	Inicio_pesquisaAssinado.pdf	13/07/2020 19:46:25	CERES GERMANNA BRAGA MORAIS	Aceito
Projeto Detalhado / Brochura Investigador	PROJETO_Ceres_Morais.pdf	15/06/2020 12:36:26	CERES GERMANNA BRAGA MORAIS	Aceito
Folha de Rosto	folha_de_rosto.pdf	15/06/2020 12:30:32	CERES GERMANNA BRAGA MORAIS	Aceito

Situação do Parecer:
Aprovado

Necessita Apreciação da CONEP:
Não

MOSSORO, 30 de Julho de 2020

Assinado por:
Ana Clara Soares Paiva Tôrres
(Coordenador(a))

Endereço: Rua Miguel Antonio da Silva Neto, s/n
Bairro: Aeroporto **CEP:** 59.607-360
UF: RN **Município:** MOSSORO
Telefone: (84)3312-7032 **E-mail:** cep@uern.br

Apêndice B - Convite aos Professores para Participação da Entrevista

Caro Professor,

Eu sou Ceres Germanna, aluna do Doutoramento em Ciências da Educação, na Universidade do Minho, e para minha tese estou desenvolvendo um estudo de caso sobre o "Processo de ensino e aprendizagem de programação no Ensino Superior", tendo como caso de pesquisa o nosso curso de Ciência da Computação.

Para a pesquisa, estou levantando dados tanto documentais, quanto por meio de questionários e entrevistas.

Desta forma, venho por este e-mail convidá-lo para colaborar com a pesquisa participando de uma entrevista sobre a temática. A entrevista tem uma duração estimada entre 40 e 60 minutos, já tendo sido aprovado pelo Comitê de Ética em Pesquisa da UERN, e será feita por meio da plataforma Zoom com gravação da entrevista.

A entrevista será conduzida por mim e seus dados pessoais estarão sempre em sigilo, de forma que não serão divulgadas nenhuma informação que possam te identificar.

Em anexo seguem o Termo de Consentimento Livre e Esclarecido, os termos de Autorização de uso e áudio e imagem, para que você possa verificá-los.

Sua participação é muito importante para o desenvolvimento do meu trabalho. Fico aguardando seu contato, com sua resposta em relação a aceitar ou não participar da entrevista. Caso possas, fico aguardando sua indicação para melhor dia e horário.

Desde já, agradeço a atenção e aguardo o seu retorno.

Atenciosamente,
Ceres Germanna Braga Morais
Doutoranda em Ciências da Educação
Especialidade Tecnologia Educativa
Instituto de Educação – Universidade do Minho

Apêndice C - Convite aos Alunos para Participação no Questionário

Caro (a) aluno (a),

Eu sou Ceres Germanna, aluna do Doutoramento em Ciências da Educação, na Universidade do Minho, e este é um convite para você participar da pesquisa "O processo de ensino e aprendizagem de programação no Ensino Superior" coordenada por mim, e que faz parte do meu projeto de tese de doutoramento.

Este questionário segue todas as recomendações das resoluções 466/12 e 510/16 do Conselho Nacional de Saúde e suas complementares, assegurando a você sigilo e segurança nos seus dados, os quais serão acessados apenas por mim.
Sua participação é voluntária, o que significa que você poderá desistir a qualquer momento, retirando seu consentimento sem que isso lhe traga nenhum prejuízo ou penalidade.

O link que dá acesso ao questionário é o seguinte: https://forms.gle/LMeWXDr8NSvZrKEF6

Sua dedicação e tempo para participar desta pesquisa (cerca de 10 minutos) são fundamentais para o sucesso dela. Caso você não consiga acessar o link, ou tenha alguma dúvida no preenchimento é só entrar em contato comigo por este e-mail.

Desde já, agradeço sua participação!

Atenciosamente,
Ceres Germanna Braga Morais
Doutoranda em Ciências da Educação
Especialidade Tecnologia Educativa
Instituto de Educação – Universidade do Minho

Apêndice D - Convite aos Alunos para Participação na Entrevista

Caro(a) aluno(a),

Eu sou Ceres Germanna, aluna do Doutoramento em Ciências da Educação, na Universidade do Minho, e este é um convite para você participar da pesquisa "O processo de ensino e aprendizagem de programação no Ensino Superior" coordenada por mim, e que faz parte do meu projeto de tese de doutoramento.

Estou entrando em contato com você pois, para minha pesquisa, preciso realizar entrevistas com alguns alunos do nosso curso de Computação, no tocante à aprendizagem de programação, para que eu possa perceber aspectos tais como as dificuldades enfrentadas, conhecimentos necessários para aprender a programar e quais os principais recursos que vocês, alunos, utilizam para aprender a programar.

A entrevista tem duração de cerca de 30 minutos, e será feita por meio da plataforma zoom com gravação da entrevista. No entanto, seus dados pessoais estarão sempre em sigilo, de forma que não serão divulgados nenhuma informação que possa te identificar.

Em anexo seguem o Termo de Consentimento de Livre Esclarecimento, os termos de autorização de uso de áudio e de imagem, para que você possa verificá-los.

Gostaria de ter sua participação na entrevista. Fico aguardando seu contato, com sua resposta em relação a aceitar ou não participar. Caso possas, fico aguardando sua indicação para melhor dia e horário.

Desde já, agradeço a atenção e aguardo o seu retorno.

Atenciosamente,
Ceres Germanna Braga Morais
Doutoranda em Ciências da Educação
Especialidade Tecnologia Educativa
Instituto de Educação – Universidade do Minho

Apêndice E - Termo de Consentimento Livre e Esclarecido – Professores

Governo do Estado do Rio Grande do Norte
Secretaria de Estado da Educação e da Cultura - SEEC
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN
Campus Central
Curso de Ciência da Computação – Departamento de Informática

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO – TCLE

Esclarecimentos

Este é um convite para você participar da pesquisa **"O processo de aprendizagem de Algoritmos e Programação no nível superior de ensino"** coordenada pelo (a) **Prof. Ceres Germanna Braga Morais** e que segue as recomendações das resoluções 466/12 e 510/16 do Conselho Nacional de Saúde e suas complementares. Sua participação é voluntária, o que significa que você poderá desistir a qualquer momento, retirando seu consentimento sem que isso lhe traga nenhum prejuízo ou penalidade.

Essa pesquisa tem como objetivo geral: "Desenvolver uma metodologia de aprendizagem de algoritmos e programação, para alunos do ensino superior, com o intuito maximizar o êxito dos alunos nessas disciplinas.". E como objetivos específicos: Inventariar as metodologias, ferramentas e práticas de ensino de algoritmos e programação no nível superior; caracterizar os fatores que podem influenciar para que um aluno aprenda a programar; explicitar as competências, conhecimentos e aptidões necessárias ao aluno para que este aprenda a programar; identificar as principais dificuldades enfrentadas pelos alunos durante a aprendizagem de programação, e o que estas dificuldades podem acarretar durante o percurso acadêmico.

O benefício desta pesquisa é a possibilidade de Conceber uma metodologia que apoie o processo de aprendizagem de algoritmos e programação no nível superior, de forma a diminuir a retenção e evasão do aluno, de forma a contribuir com o sucesso destes nestas disciplinas.

Os riscos mínimos que o participante da pesquisa estará exposto são de constrangimento, uma vez que o professor estará apresentando sua opinião em relação ao processo de aprendizagem de algoritmos e programação vivenciado po seus alunos. Esses riscos serão minimizados mediante: Garantia do anonimato/privacidade do participante na pesquisa, onde não será preciso colocar o nome do mesmo; Para manter o sigilo e o respeito ao participante da pesquisa, apenas a pesquisadora responsável Ceres Germanna Braga Morais, aplicará os instrumentos de recolha de dados, e somente a pesquisadora responsável poderá manusear e guardar estes dados; Sigilo das informações por ocasião da publicação dos resultados, visto que não será divulgado dado que identifique o participante; Garantia que o participante se sinta a vontade para responder aos questionários e Anuência das Instituições de ensino para a realização da pesquisa.

Todas as informações obtidas serão sigilosas. Seu nome não será identificado em nenhum momento. Os dados coletados permanecerão guardados pelo período de 5 anos, na forma de arquivos físicos sob a responsabilidade da pesquisadora responsável no Departamento de Informática, a fim de garantir a confidencialidade, a privacidade e a segurança das informações coletadas, e a divulgação dos resultados será feita de forma a não identificar os participantes e o responsável.

Você ficará com uma via original deste TCLE e toda a dúvida que você tiver a respeito desta pesquisa, poderá perguntar diretamente para a pesquisadora Ceres Germanna Braga Morais da Universidade do Estado do Rio Grande do Norte/RN, Campus Central, no endereço

Rua Professor Antônio Campos, s/nº, BR 110 Km 48, Bairro Costa e Silva, CEP 59.600-000, Mossoró/RN. Tel.(84) 3315-2225.

Dúvidas a respeito da ética desta pesquisa poderão ser questionadas ao **Comitê de Ética em Pesquisa (CEP-UERN) –** Faculdade de Medicina da UERN - Rua Miguel Antonio da Silva Neto s/n - Aeroporto
Home page: http://www.uern.br - e-mail: cep@uern.br – CEP: 59607-360 - Mossoró –RN Tel: (84) 3312-7032.

Se para o participante houver gasto de qualquer natureza, em virtude da sua participação nesse estudo, é garantido o direito a indenização (Res. 466/12 II.7) – cobertura material para reparar dano – e/ou ressarcimento (Res. 466/12 II.21) – compensação material, exclusivamente de despesas do participante e seus acompanhantes, quando necessário, tais como transporte e alimentação – sob a responsabilidade do (a)pesquisador(a) Ceres Germanna Braga Morais.

Não será efetuada nenhuma forma de gratificação por sua participação. Os dados coletados farão parte do nosso trabalho, podendo ser divulgados em eventos científicos e publicados em revistas nacionais ou internacionais. O pesquisador estará à disposição para qualquer esclarecimento durante todo o processo de desenvolvimento deste estudo. Após todas essas informações, agradeço antecipadamente sua atenção e colaboração.

Consentimento Livre

Concordo em participar desta pesquisa **"O processo de aprendizagem de Algoritmos e Programação no nível superior de ensino".** Declarando, para os devidos fins, que fui devidamente esclarecido quanto aos objetivos da pesquisa, aos procedimentos aos quais meu/ minha filho (a) será submetido (a) e dos possíveis riscos que possam advir de tal participação. Foram garantidos a mim esclarecimentos que venham a solicitar durante a pesquisa e o direito de desistir da participação em qualquer momento, sem que minha desistência implique em qualquer prejuízo a minha pessoa ou a minha família. Autorizo assim, a publicação dos dados da pesquisa, a qual me garante o anonimato e o sigilo dos dados referentes à minha identificação.

Cidade, _____/_____/_____.

_____
Assinatura do Pesquisador

_____
Assinatura do Participante

Prof Ceres Germanna Braga Morais (Pesquisadora Responsável) - Doutoranda em Ciências da Educação, Especialidade Tecnologia Educativa, do Instituto de Educação, da Universidade do Minho/Portugal e professora do Curso de Ciência da Computação, da Universidade do Estado do Rio Grande do Norte – UERN, Campus Central, no endereço Professor Antônio Campos, n. s/nº , Costa e Silva, CEP 59600-000– Mossoró – RN. Tel.(84) 3315-2225

Comitê de Ética em Pesquisa (CEP-UERN) - Faculdade de Medicina da UERN - Rua Miguel Antonio da Silva Neto s/n - Aeroporto
Home page: http://www.uern.br - e-mail: cep@uern.br – CEP: 59607-360 - Mossoró –RN Tel: (84) 3312-7032.

Apêndice F - Termo de Consentimento Livre e Esclarecido – Alunos

Governo do Estado do Rio Grande do Norte
Secretaria de Estado da Educação e da Cultura - SEEC
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN
Campus Central
Curso de Ciência da Computação – Departamento de Informática

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO – TCLE

Esclarecimentos

Este é um convite para você participar da pesquisa **"O processo de aprendizagem de Algoritmos e Programação no nível superior de ensino"** coordenada pelo (a) **Prof. Ceres Germanna Braga Morais** e que segue as recomendações das resoluções 466/12 e 510/16 do Conselho Nacional de Saúde e suas complementares. Sua participação é voluntária, o que significa que você poderá desistir a qualquer momento, retirando seu consentimento sem que isso lhe traga nenhum prejuízo ou penalidade.

Essa pesquisa tem como objetivo geral: "Desenvolver uma metodologia de aprendizagem de algoritmos e programação, para alunos do ensino superior, com o intuito maximizar o êxito dos alunos nessas disciplinas.". E como objetivos específicos: Inventariar as metodologias, ferramentas e práticas de ensino de algoritmos e programação no nível superior; caracterizar os fatores que podem influenciar para que um aluno aprenda a programar; explicitar as competências, conhecimentos e aptidões necessárias ao aluno para que este aprenda a programar; identificar as principais dificuldades enfrentadas pelos alunos durante a aprendizagem de programação, e o que estas dificuldades podem acarretar durante o percurso acadêmico.

O benefício desta pesquisa é a possibilidade de Conceber uma metodologia que apoie o processo de aprendizagem de algoritmos e programação no nível superior, de forma a diminuir a retenção e evasão do aluno, de forma a contribuir com o sucesso destes nestas disciplinas.

Os riscos mínimos que o participante da pesquisa estará exposto são de constrangimento, uma vez que o aluno estará apresentando sua opinião em relação ao processo de aprendizagem de algoritmos e programação vivenciado por ele. Esses riscos serão minimizados mediante: Garantia do anonimato/privacidade do participante na pesquisa, onde não será preciso colocar o nome do mesmo; Para manter o sigilo e o respeito ao participante da pesquisa, apenas a pesquisadora responsável Ceres Germanna Braga Morais, aplicará os instrumentos de recolha de dados, e somente a pesquisadora responsável poderá manusear e guardar estes dados; Sigilo das informações por ocasião da publicação dos resultados, visto que não será divulgado dado que identifique o participante; Garantia que o participante se sinta a vontade para responder aos questionários e Anuência das Instituições de ensino para a realização da pesquisa.

Todas as informações obtidas serão sigilosas. Seu nome não será identificado em nenhum momento. Os dados coletados permanecerão guardados pelo período de 5 anos, na forma de arquivos físicos sob a responsabilidade da pesquisadora responsável no Departamento de Informática, a fim de garantir a confidencialidade, a privacidade e a segurança das informações coletadas, e a divulgação dos resultados será feita de forma a não identificar os participantes e o responsável.

Você ficará com uma via original deste TCLE e toda a dúvida que você tiver a respeito desta pesquisa, poderá perguntar diretamente para a pesquisadora Ceres Germanna Braga Morais da Universidade do Estado do Rio Grande do Norte/RN, Campus Central, no endereço

Rua Professor Antônio Campos, s/nº, BR 110 Km 48, Bairro Costa e Silva, CEP 59.600-000, Mossoró/RN. Tel.(84) 3315-2225.

Dúvidas a respeito da ética desta pesquisa poderão ser questionadas ao **Comitê de Ética em Pesquisa (CEP-UERN) –** Faculdade de Medicina da UERN - Rua Miguel Antonio da Silva Neto s/n - Aeroporto
Home page: http://www.uern.br - e-mail: cep@uern.br – CEP: 59607-360 - Mossoró –RN Tel: (84) 3312-7032.

Se para o participante houver gasto de qualquer natureza, em virtude da sua participação nesse estudo, é garantido o direito a indenização (Res. 466/12 II.7) – cobertura material para reparar dano – e/ou ressarcimento (Res. 466/12 II.21) – compensação material, exclusivamente de despesas do participante e seus acompanhantes, quando necessário, tais como transporte e alimentação – sob a responsabilidade do (a)pesquisador(a) Ceres Germanna Braga Morais.

Não será efetuada nenhuma forma de gratificação por sua participação. Os dados coletados farão parte do nosso trabalho, podendo ser divulgados em eventos científicos e publicados em revistas nacionais ou internacionais. O pesquisador estará à disposição para qualquer esclarecimento durante todo o processo de desenvolvimento deste estudo. Após todas essas informações, agradeço antecipadamente sua atenção e colaboração.

Consentimento Livre

Concordo em participar desta pesquisa **"O processo de aprendizagem de Algoritmos e Programação no nível superior de ensino".** Declarando, para os devidos fins, que fui devidamente esclarecido quanto aos objetivos da pesquisa, aos procedimentos aos quais meu/ minha filho (a) será submetido (a) e dos possíveis riscos que possam advir de tal participação. Foram garantidos a mim esclarecimentos que venham a solicitar durante a pesquisa e o direito de desistir da participação em qualquer momento, sem que minha desistência implique em qualquer prejuízo a minha pessoa ou a minha família. Autorizo assim, a publicação dos dados da pesquisa, a qual me garante o anonimato e o sigilo dos dados referentes à minha identificação.

Cidade, _____/_____/_____.

_____
Assinatura do Pesquisador

_____
Assinatura do Participante

Prof Ceres Germanna Braga Morais (Pesquisadora Responsável) – Doutoranda em Ciências da Educação, Especialidade Tecnologia Educativa, do Instituto de Educação, da Universidade do Minho/Portugal e professora do Curso de Ciência da Computação, da Universidade do Estado do Rio Grande do Norte – UERN, Campus Central, no endereço Professor Antônio Campos, n. s/nº , Costa e Silva, CEP 59600-000– Mossoró – RN. Tel.(84) 3315-2225

Comitê de Ética em Pesquisa (CEP-UERN) - Faculdade de Medicina da UERN - Rua Miguel Antonio da Silva Neto s/n - Aeroporto
Home page: http://www.uern.br - e-mail: cep@uern.br – CEP: 59607-360 - Mossoró –RN Tel: (84) 3312-7032.

Apêndice G - Termo de Autorização para uso de Áudio

TERMO DE AUTORIZAÇÃO PARA USO DE ÁUDIO

Eu _____(*PARTICIPANTE DA PESQUISA*), depois de conhecer e entender os objetivos, procedimentos metodológicos, riscos e benefícios da pesquisa, bem como de estar ciente da necessidade da gravação de áudio produzido por mim, especificados no Termo de Consentimento Livre e Esclarecido (TCLE), AUTORIZO, através do presente termo, os pesquisadores **Ceres Germanna Braga Morais e António José Meneses Osório** do projeto de pesquisa intitulado "**O processo de aprendizagem de Algoritmos e Programação no nível superior de ensino**" a realizar captação de áudios que se façam necessários sem quaisquer ônus financeiros a nenhuma das partes.

Ao mesmo tempo, libero a utilização destes áudios (suas respectivas cópias) para fins científicos e de estudos (livros, artigos, monografias, TCC's, dissertações ou teses, além de slides e transparências), em favor dos pesquisadores da pesquisa, acima especificados, obedecendo ao que está previsto nas Leis que resguardam os direitos das crianças e adolescentes (ECA, Lei N.º 8.069/ 1990), dos idosos (Lei N.º 10.741/2003) e das pessoas com deficiência (Decreto Nº 3.298/1999, alterado pelo Decreto Nº 5.296/2004).

Mossoró - RN, __ de _____ de 2020

_____

Assinatura do participante da pesquisa

_____

Assinatura do pesquisador responsável

IMPRESSÃO DATILOSCÓPICA

__(ESTE DOCUMENTO DEVERÁ SER ELABORADO EM DUAS VIAS; DAS QUAIS UMA VIA DEVERÁ FICAR COM O PARTICIPANTE DA PESQUISA E A OUTRA COM O PESQUISADOR RESPONSÁVEL)__

Apêndice H - Termo de Autorização de uso de Imagem

TERMO DE AUTORIZAÇÃO PARA USO DE IMAGEM

Eu _____ (*PARTICIPANTE DA PESQUISA*),depois de conhecer e entender os objetivos, procedimentos metodológicos, riscos e benefícios da pesquisa, bem como de estar ciente da necessidade do uso de minha imagem, especificados no Termo de Consentimento Livre e Esclarecido (TCLE), AUTORIZO, através do presente termo, os pesquisadores **Ceres Germanna Braga Morais e António José de Meneses Osório** do projeto de pesquisa intitulado **"O processo de aprendizagem de Algoritmos e Programação no nível superior de ensino"** a realizar captação de imagens que se façam necessárias sem quaisquer ônus financeiros a nenhuma das partes.

Ao mesmo tempo, libero a utilização destas imagens (suas respectivas cópias) para fins científicos e de estudos (livros, artigos, monografias, TCC's, dissertações ou teses, além de slides e transparências), em favor dos pesquisadores da pesquisa, acima especificados, obedecendo ao que está previsto nas Leis que resguardam os direitos das crianças e adolescentes (ECA, Lei N.° 8.069/ 1990), dos idosos (Lei N.° 10.741/2003) e das pessoas com deficiência (Decreto N° 3.298/1999, alterado pelo Decreto N° 5.296/2004).

Mossoró - RN, __ de _____ de 2020

_____

Assinatura do participante da pesquisa

_____

Assinatura do pesquisador responsável

IMPRESSÃO DATILOSCÓPICA

__(ESTE DOCUMENTO DEVERÁ SER ELABORADO EM DUAS VIAS; DAS QUAIS UMA VIA DEVERÁ FICAR COM O PARTICIPANTE DA PESQUISA E A OUTRA COM O PESQUISADOR RESPONSÁVEL)__

Apêndice I - Questionário Apresentado aos Alunos

O processo de ensino e aprendizagem de programação no Ensino Superior

1. Prezado(a)

Este é um convite para você participar da pesquisa "O processo de aprendizagem de Algoritmos e Programação no nível superior de ensino" coordenada por Ceres Germanna Braga Morais, aluna do Doutoramente em Ciências da Educação, na Universidade do Minho, e que segue as recomendações das resoluções 466/12 e 510/16 do Conselho Nacional de Saúde e suas complementares.
Sua participação é voluntária, o que significa que você poderá desistir a qualquer momento, retirando seu consentimento sem que isso lhe traga nenhum prejuízo ou penalidade.

O Link abaixo te dá acesso ao Termo de Consentimento Livre e Esclarecido desta pesquisa, o qual deve ser lido antes de você iniciar sua participação:

https://drive.google.com/file/d/1SUQ1yY_3ga1M6BGWXMKcTzkRE1UdjTol/view?usp=sharing

*Obrigatório

1. Se você concordar participar, de forma voluntária, deverá marcar a opção abaixo, correspondentes à assinatura do Termo de Consentimento Livre e Esclarecido. Sua dedicação e tempo em participar desta pesquisa são fundamentais para o sucesso dela. Agradeço sua participação! *

Marque todas que se aplicam.

- [ ] Aceita participar da pesquisa

Sobre você

2. Você já cursou ou está cursando alguma disciplina de algoritmos ou programação? *

Marcar apenas uma oval.

- ◯ Sim *Pular para a pergunta 3*
- ◯ Não *Pular para a seção 12 (Obrigada pela participação!)*

Sobre você

3. Qual a sua idade? *

Digite sua idade

4. Você está em que período do curso? *

Marcar apenas uma oval.

- ◯ Segundo Período
- ◯ Terceiro Período
- ◯ Quarto Período
- ◯ Quinto Período
- ◯ Sexto Período
- ◯ Sétimo Período
- ◯ Oitavo Período

Sobre seu percurso acadêmico nas disciplinas de Algoritmos e Programação

5. Quais destas disciplinas de algoritmos e/ou programação você já cursou? *

Marque todas que se aplicam.

- ☐ Construção de Algoritmos
- ☐ Estrutura de Dados
- ☐ Programação Estruturada
- ☐ Programação Orientada a Objetos
- ☐ Programação Avançada

6. Das disciplinas listadas acima, você já desistiu de alguma? *

Marcar apenas uma oval.

- ◯ Sim *Pular para a pergunta 7*
- ◯ Não *Pular para a pergunta 8*

Sobre seu percurso acadêmico nas disciplinas de Algoritmos e Programação

7. De qual(ais) disciplina(s) você desistiu?

Marque todas que se aplicam.

- [ ] Construção de Algoritmos
- [ ] Estrutura de Dados
- [ ] Programação Estruturada
- [ ] Programação Orientada a Objetos
- [ ] Programação Avançada

Sobre seu percurso acadêmico nas disciplinas de Algoritmos e Programação

Construção de Algoritmos
Estrutura de Dados
Programação Estruturada
Programação Orientada a Objetos
Programação Avançada

8. Das disciplinas listadas acima, você já reprovou alguma? *

Marcar apenas uma oval.

- ◯ Sim *Pular para a pergunta 9*
- ◯ Não *Pular para a pergunta 10*

Sobre seu percurso acadêmico nas disciplinas de Algoritmos e Programação

9. Qual(ais) disciplina(s) você reprovou? *

Marque todas que se aplicam.

- [ ] Construção de Algoritmos
- [ ] Estrutura de Dados
- [ ] Programação Estruturada
- [ ] Programação Orientada a Objetos
- [ ] Programação Avançada

Sobre os fatores que te influenciam no aprendizado de algoritmos e programação

Para as afirmações a seguir, inidique se "discorda totalmente", "discorda em parte", "concorda em parte" ou "concorda totalmente", em relação a influência destes no seu processo de aprendizagem em disciplinas de algoritmos e/ou programação.

10. A linguagem de programação utilizada pelo professor me ajudou a aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

11. Me sentir motivado e ser motivado pelos professores e colegas me ajudou a aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

12. A minha experiência com programação me ajudou a aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

13. O tempo que eu possuo disponível para estudar me ajudou a aprender a programar *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

14. A forma que eu estudo (a metodologia que utilizo) me ajudou a aprender a programar *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

15. O apoio que o professor me dá em sala de aula foi importante para que eu aprendesse a programar *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

16. O feedback fornecido pelo professor fora de sala de aula me ajudou a aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

17. A minha compreensão em relação aos conceitos básicos de algoritmos e programação me ajudou a aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

Sobre as competências e conhecimentos necessários para aprender algoritmos e programação

Para as afirmações a seguir, inidique se "discorda totalmente", "discorda em parte", "concorda em parte" ou "concorda totalmente", em relação às competências e conhecimentos que você considera necessárias para a aprendizagem de algoritmos e programação

18. Precisei ter habilidade em resolver problemas para aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

19. Precisei ter a compreensão da sintaxe e da estrutura do programa para aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

20. Foi necessário que eu tivesse pensamento lógico para aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

21. Foi necessário que eu tivesse pensamento computacional para aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

22. Precisei ter autonomia para aprender a programar *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

23. Precisei ter conhecimento em ferramentas (IDEs) para aprender a programar *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

24. Foi necessário que eu fosse proativo para aprender a programar *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

25. Tive que ter a habilidade de planejamento para aprender a programar *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

26. Precisei ter a habilidade de leitura de código para aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

27. Precisei ter domínio na língua inglesa para aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

28. Foi necessário que eu tivesse conhecimento em matemática para aprender a programar *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

Sobre as dificuldades enfrentadas no processo de aprendizagem de algoritmos e programação

Para as afirmações a seguir, responda se "discorda totalmente", "discorda em parte", "concorda em parte" ou "concorda totalmente", em relação aos fatores que dificultam ou dificultaram o processo de aprendizagem de algoritmos e programação.

29. Não compreender um conceito-chave dificulta a aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

30. Precisar de mais tempo para compreender um determinado assunto dificulta o processo de aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

31. A dissociação entre os exercícios vistos em sala de aula e o mundo real dificulta a aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

32. A linguagem de programação usada pelo professor pode dificultar a aprendizagem de programação *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

33. O aluno pode ter dificuldade em aprender a programar por não possuir experiência em programação *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

34. Não conseguir interpretar o problema a ser resolvido dificulta a aprendizagem de programação *

Marcar apenas uma oval.

- Discordo totalmente
- Discordo em parte
- Concordo em parte
- Concordo totalmente

35. Uma turma com muitos alunos dificulta o processo de aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

36. Uma turma com alunos de diferentes perfis e níveis de conhecimentos em relação à programação dificulta o processo de aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

37. Ter pouco conhecimento em matemática dificulta a aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

38. A natureza dos problemas computacionais e como resolvê-los algoritmicamente dificultam a aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

39. A autoconfiança baixa do aluno pode prejudicar o processo de aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

40. Um aluno desmotivado tem dificuldade no processo de aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

41. Não ter conhecimento em na Língua Inglesa dificulta o processo de aprendizagem de programação *

Marcar apenas uma oval.

- ◯ Discordo totalmente
- ◯ Discordo em parte
- ◯ Concordo em parte
- ◯ Concordo totalmente

Se possível, conte-nos um pouco da sua experiência nas disciplinas de algoritmos e programação!

42. Fale um pouco do que você mais gostou nas disciplinas de algoritmos e programação, quais suas maiores dificuldades, por que desistiu da disciplina, o que mais te ajudou a aprender a programar...

Obrigada pela participação!

Apêndice J - Guião da Entrevista Semiestruturada Aplicada aos Alunos

Roteiro da entrevista semiestruturada para os alunos
Pesquisadora responsável: Ceres Germanna Braga Morais

Tema: "PROCESSO DE APRENDIZAGEM DE PROGRAMAÇÃO NO ENSINO SUPERIOR"

Quebra-gelo: Apresentar-me, apresentar o objetivo da entrevista, perguntar nome do aluno, a idade, quando entrou no curso, o período em que está matriculado e as UC de programação que está cursando ou que já cursou.

Perguntas pré-estabelecidas:

Objetivos	Pergunta
Objetivo 1: Inventariar metodologias, ferramentas e estratégias de ensino e aprendizagem de programação	1.1. Nas disciplinas de algoritmos e programação, como você costuma estudar? 1.2. Durante as disciplinas de algoritmos e programação, quais ferramentas você utilizou? 1.2.1. Você já as conhecia ou foram indicadas pelos professores das disciplinas? 1.2.2. O uso dessas ferramentas te ajudou a aprender a programar?
Objetivo 2: Explicitar as competências, conhecimentos e aptidões necessárias para aprender a programar	2.1. Na sua opinião, o que o aluno precisa saber para aprender a programar? 2.2. Para aprender a programar que habilidades e competências você considera importante ter?
Objetivo 3: Caracterizar os fatores que podem influenciar para o aluno aprender a programar	3.1. Durante as disciplinas, na sua opinião, que fatores auxiliaram para que você aprendesse melhor os conceitos de algoritmos e programação? 3.2. Você possuía algum conhecimento prévio em programação? Se sim, isso te ajudou nas disciplinas cursada? 3.3. Na sua opinião, quais atitudes dos professores e dos colegas influenciam ou podem afetar no seu processo de aprendizagem?
Objetivo 4: Identificar as dificuldades enfrentadas pelos alunos durante a aprendizagem de programação, e o que estas dificuldades acarretam	4.1. Você sentiu dificuldade em aprender algoritmos e programação? Se sim, na sua opinião, quais foram as maiores dificuldades que você sentiu ao cursar essas disciplinas? 4.1.1. O que essas dificuldades te causaram durante a disciplina? 4.2. Você já pensou em desistir ou desistiu de alguma disciplina de programação? 4.2.1. Se sim, em quais? 4.2.2. Na sua opinião, o que te levou a essa opção? 4.3. Você reprovou em alguma disciplina de algoritmos ou programação? 4.3.1. Se sim, em quais? 4.3.2. Na sua opinião, quais os principais motivos que te levaram à reprovação?

Observação: Outras perguntas poderão surgir no momento da entrevista.

Apêndice K - Guião da entrevista Semiestruturada Aplicada aos Professores

Roteiro da entrevista semiestruturada para os professores
Pesquisadora responsável: Ceres Germanna Braga Morais

Tema: "PROCESSO DE ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO NO ENSINO SUPERIOR"

Quebra-gelo: Apresentar-me, apresentar o objetivo da entrevista, perguntar nome do professor, tempo de docência de disciplinas de programação e quais disciplinas de programação ministra.

Perguntas pré-estabelecidas:

Objetivos	Pergunta
Objetivo 1: Inventariar metodologias, ferramentas e estratégias de ensino e aprendizagem de programação	1.1. Que estratégias e metodologias de ensino você utiliza para ensinar programação? 1.2. Nas disciplinas de programação você sugere aos alunos algum método de estudo? 1.3. Durante as disciplinas de programação, quais ferramentas você utiliza e indica aos alunos? 1.4. Há alguma metodologia, estratégia e/ou ferramenta você gostaria de utilizar em sala de aula para melhorar a aprendizagem dos alunos em programação? Se sim, Qual (is))?
Objetivo 2: Explicitar as competências, conhecimentos e aptidões necessárias para aprender a programar	2.1. Na sua opinião, que competências os alunos devem ter (ou adquirir) para aprender a programar? 2.2. Na sua opinião, que conhecimentos os alunos devem ter (ou adquirir) para aprender a programar? 2.3. E que que aptidões são necessárias para que eles aprendam a programar?
Objetivo 3: Caracterizar os fatores que podem influenciar para o aluno aprender a programar	3.1. Na sua opinião, quais atitudes dos alunos, frente a disciplina de programação, podem ajudá-lo na aprendizagem? 3.1.1. E quais podem prejudicá-lo? 3.2. Na sua opinião, que atitudes externas aos alunos, podem influenciar o processo de aprendizagem de programação dos alunos? 3.2.1. E quais podem prejudicá-las?
Objetivo 4: Identificar as dificuldades enfrentadas pelos alunos durante a aprendizagem de programação, e o que estas dificuldades acarretam	4.1. Na sua opinião, quais as maiores dificuldades apresentadas pelos alunos para aprender algoritmos e programação? 4.2. O que estas dificuldades enfrentadas pelo aluno geralmente acarretam ou podem acarretar seu percurso acadêmico?

Observação: Outras perguntas poderão surgir no momento da entrevista.

Apêndice L – Guião de Orientação para Análise Documental

Roteiro para análise documental

Pesquisadora responsável: Ceres Germanna Braga Morais

Tema: Processo de ensino e aprendizagem de programação no Ensino Superior

Objetivos	**Perguntas**
Inventariar metodologias e ferramentas usadas para o ensino e aprendizagem de programação	1. O documento aborda estratégias de ensino e aprendizagem de programação? Se sim, quais? 2. O documento elenca ferramentas para o ensino e aprendizagem de programação? Se sim, quais?
Caracterizar os fatores que podem influenciar para o aluno aprender a programar	1. O documento apresenta fatores que auxiliam o processo de ensino e aprendizagem? 2. O documento apresenta a estrutura do curso e se este auxilia no processo?
Explicitar as competências, conhecimentos e aptidões necessárias para aprender a programar	1. O documento aborda os conhecimentos necessários para aprender a programar? Se sim, quais? 2. O documento traz as habilidades importantes para aprender a programar? Se sim, quais?
Identificar as dificuldades enfrentadas pelos alunos e professores durante o processo de aprendizagem, e o que estas dificuldades acarretam ao aluno em seu percurso acadêmico.	1. O documento explicita dificuldades que podem ser enfrentadas no processo de ensino e aprendizagem? Se sim, Quais? 1.2. O documento explicita as consequências das dificuldades? Se sim, quais?

A análise documental deve ainda:

- Contemplar o tipo do documento analisado
- O ano de publicação (se aplicável)
- Informações que caracterizem o caso (Unidades curriculares, tempo de curso, estrutura do curso)

Apêndice M - Guia de orientação para codificação e transcrição das entrevistas

Guia de orientação para codificação e transcrição de entrevistas[22]

Pesquisadora responsável: Ceres Germanna Braga Morais

1. Cada áudio ou vídeo está identificado com um código do instrumento de recolha e uma numeração seguido da identificação do projeto, data de registro e código da pessoa observada, conforme descrição abaixo.

Código do instrumento	Descrição
EN	Entrevista
ALU	Aluno
PROF	Professor
CM	Entrevistadora investigadora

Exemplos:

EN1 – ALU1 – 10/07/2020 – significa a primeira entrevista com o aluno 1, ocorrida no dia 10 de julho de 2020.

2. No início do documento da transcrição o código acima deverá estar indicado e destacado em negrito. Cada código será usado apenas uma vez, pois as atividades são únicas.

Exemplo de uma entrevista transcrita:

EN1 – ALU1- 10/07/2020

CM: *Olá, boa tarde!*

ALU1: *Boa tarde, professora.*

[pausa breve]

CM: Podemos começar a entrevista?

ALU1: *Sim, professora.*

3. Tudo que constar na gravação do áudio ou vídeo deverá ser descrito/escrito (risos, risadas, pausas e outros sons) da seguinte forma:

Sons: descrever entre parênteses retos. Ex.: [gargalhadas dos alunos]; [risos dos alunos].

Pausas, desenvolvimento de atividades grupais, falas inaudíveis: descrever entre colchetes. Ex.: [inaudível], [pausa], [alunos desenvolvem atividade em grupo/individualmente].

Observações:

[22] Apêndice elaborado a partir de: Oliveira, J. P. (2020). A formação de estudantes do Ensino Médio Integrado no Brasil: contributos para os estudos sobre programas de extensão em Institutos Federais (Tese de doutoramento). Évora: Universidade de Évora. (apêndice 7, p. 480). Disponível em http://dspace.uevora.pt/rdpc/bitstream/10174/27838/3/Doutoramento-Ciencias_da_Educacao-Joao_Paulo_de_Oliveira-2vol_Apendices.pdf acessado em 04 de janeiro de 2021.

Para pausas longas e atividades longas sem diálogo, informar o tempo ou a indicação longa ou breve. Ex.: [pausa longa. 30min].

Para frases ou palavras não compreensíveis no áudio/vídeo escrever inaudível entre colchetes. Ex.: Vocês precisam prestar mais [inaudível] aula.

4. Salvar as cópias das transcrições em disco local e no e-mail tese.ceres@gmail.com à medida que for concluindo as transcrições.

Coimbra, 04 de janeiro de 2021.

Apêndice N - Unidades de Registro dos Questionários com os Alunos de acordo com os Parâmetros e Categorias

Parâmetro	Categoria	Unidade de registro
O que	Background	(A23.1) Gostei bastante, já tinha um conhecimento em programação antes de cursar as disciplinas por isso foi mais fácil assimilar o conteúdo.
	Conhecimentos e habilidades	(A14.1) O fato de ter cursado a disciplina de Lógica um semestre antes de cursar a disciplina de Construção de Algoritmos ajudou bastante no processo de absorção e assimilação do conteúdo
	Fatores que auxiliam	
Quem	Papel do docente	(A02.1) com o auxílio de um dos professores, que foi muito importante, consegui superar as dificuldades das matérias [UC] (A10.1) O que mais ajudou a programar foi a maneira de ensinar do professor que foi super atencioso e acompanhava o passo a passo ajudando a superar as dificuldades e motivando sempre a fazer novos exercícios para melhor fixar o aprendizado (A29.1) mas com o auxílio do professor que sempre esteve pronto para tirar as dúvidas e com a prática diária é possível compreender.
	Papel dos colegas	
	Papel de outros	
Onde	Onde estuda	
	Onde busca informação	
Quando	Entrada no curso	
	Período regular	
	UC de programação	
Porque	Dificuldades enfrentadas	(A02.2) Companheiros de turma também tinham o costume de mostrar o quanto seus êxitos superavam os fracassos de outros (que me incluía, também, na lista dos fracassos), isso era muito desmotivador. (A02.3) devido a problemas pessoais acabei desenvolvendo uma 'baixa estima' sobre o quando eu poderia aprender e fazer [...] por isso acabava não comparecendo às aulas ou até trancando (A05.1) Certo que programação não é somente sintaxe porém para alguém como eu que nunca havia tido contato com programação foi desafiador a linguagem C (A10.2) A minha maior dificuldade em programação é transformar um problema real em um código e poder resolvê-lo (A11.1) Minhas maiores dificuldades aconteceram no início da faculdade por nunca ter visto nada relacionado a programação (A12.1) sou oriundo de escola pública, onde o ensino possui uma deficiência crônica e sair de um Ensino Médio com várias lacunas e ter que se adequar a uma nova realidade, demanda um tempo (A12.2) Em um primeiro momento as maiores dificuldades estiveram relacionadas ao fato de ser o primeiro contato com algoritmos, lógica e o pensamento matemático para resolução de problemas computacionais. (A15.1) Infelizmente cada aluno apresenta condições e níveis bastante variados fazendo com que o professor tivesse que ir e voltar durante a aula para englobar a todos. Esse processo fazia com que alguns se perdessem e até mesmo se sentissem confusos e desmotivados ao aprendizado. (A25.1) o que me dá mais dificuldade é a sintaxe das linguagens, as vezes são muito confusas (A29.2) tive um pouco de dificuldade na abstração de alguns conceitos para passa-los para o código, como por exemplo nas Árvores Binárias de Busca. (A30.1) Me sentia um pouco envergonhado [por ter dificuldades] pois tinha bastante gente na turma que tinha uma facilidade de entendimento bem maior que o meu. (A30.2) Quando aconteceu de eu ter perdido o foco em apenas uma aula, virou uma bola de neve, e eu não conseguia acompanhar o conteúdo
	Consequências das dificuldades	(A30.3) reprovei no primeiro período, pois eu me sentia desmotivado, sem foco, sem vontade
Como	Metodologia docente	(A03.1) o professor não conseguiu apresentar um uso real daquela linguagem, ter que aprender algo que você sabe que nunca vai usar é desmotivador (A05.2) Estudei a matéria de construção de algoritmos duas vezes. Reprovei na primeira e justifico meu fracasso com a linguagem que foi abordada que foi a C. O professor sempre

		era dedicado em repassar o conhecimento. Porém a linguagem em si se tornou um problema. (A05.3) a 2ª vez [que cursou a UC] foi com um outro professor que abordou outra linguagem [de programação] nela aprendi muito e comecei a desenvolver raciocínio lógico nas questões (A08.1) os professores sempre demonstraram clareza e facilidade para repassar o conteúdo, as linguagens de programação apresentadas também facilitaram bastante (A11.2) o que mais me ajudou a aprender a programar foi a didática utilizada pelos professores aliada à paciência que tive para aprender com os erros (A19.1) O que eu mais gosto nas disciplinas de programação são os projetos finais que os professores geralmente passam no final das disciplinas, pois envolvem todo conhecimento adquirido ao decorrer do semestre, e se torna um desafio muito instigante de se resolver e desenvolver. (A22.1) A parte da programação é relativamente simples, no entanto ao manifestar algoritmicamente a prática, revela ao aluno que programar é um pouco abstrato e que requer treino e persistência (A25.2) Gostei muito da linguagem Python, que foi a utilizada pelo professor, e comecei a me dedicar nela. (A26.1) a abordagem do professor era interessante, felizmente não desisti de nenhuma disciplina e fui aprovado nas que paguei
	Metodologia discente	(A26.2) A leitura e prática são essenciais para o aprendizado (A29.3) só com a prática diária é possível compreender [os assuntos]
	Ferramentas que utiliza	
Quanto	Importância do aprendizado	(A03.2) é ver como o conteúdo da aula poderia ser usado para resolver um problema (A04.1) Ter mais de uma visão sobre o mesmo problema para resolver, e ter mais tempo e saber que tipo de área na programação que gosto ajudou a ter vontade de trabalhar em novos projetos (A10.3) A forma de interpretar os problemas reais e conseguir desenvolver uma ferramenta que resolva esse problema é muito gratificante (A11.3) o que mais gostei nas disciplinas de programação é que foi delas que saíram meus primeiros projetos de desenvolvimento. (A14.2) O que mais gostei de ver ao cursar a disciplina de Construção de Algoritmos foi ver que existem várias formas de programar, e que não existe 'forma certa' ou 'forma errada' e sim, formas mais eficazes
	Dedicação e iniciativa	(A25.3) Mas, percebi que outros professores usam outras linguagens, então decidi aprender C++ em concomitância a Python
	Visão do futuro	(A25.4) foi muito bom pois foi daí que comecei a criar meus programas, o que me instigou. O que me ajuda a continuar a aprender é meu objetivo de ser o melhor que eu puder nessa área

Apêndice O - Unidades de Registro das Entrevistas com os Alunos de acordo com os Parâmetros e Categorias

Parâmetro	Categorias	Unidades de registro
O que	Background	(ALU1.1) "Quando eu entrei eu tinha uma noção básica do que eram as coisas. Mas é claro que aquela noção básica que eu tinha lá quando entrei no primeiro período não era nada que fosse a realidade dos períodos que vinham depois [risos]. Mas eu nunca tinha tido nenhuma experiência com programação mesmo, mas antes de entrar eu já fazia alguns sitezinhos em HTML, essas coisas, eu sempre gostei dessas coisas. HTML, CSS, essas coisas, mas em relação à programação mesmo eu nunca tinha mexido não antes de entrar no curso." (ALU2.1) "Na verdade de conhecimento eu não tinha nenhum, quando eu entrei na [nome da Instituição omitida] assim de programação, nada, nada, nada... eu tinha visto Pascal... eu acho que a maioria dos alunos inclusive, entram assim meio sem saber né programação [...] mas eu tinha visto alguma coisa de Pascal já antes mas eu não tinha muita noção. [...] por que, por exemplo, eu fazia escola particular, eu não tinha noção do que era Ciência da Computação" (ALU3.1) "por que assim, o meu conhecimento em matemática era muito pouco, eu fui aluno de escola pública e eu tive um período no meu segundo grau que a gente quase não tinha professor, passei muito tempo sem professor de matemática, e quando voltava fazia assim um trabalho uma coisa assim, para cumprir a carga-horária só que a gente não via o conteúdo..." (ALU3.2) "Quando, logo quando eu comecei o curso que eu via meus colegas sempre falando de programação, que passavam a noite programando, e não sei o que... eu ficava "meu Deus do céu, o que é isso?", eles falam tanto de programação e eu aqui sentado sem saber nem o que eles estão falando, aí eu dizia "quando chegar a minha vez, vamos ver o que é que vai ser né?" (ALU3.3) "Pronto, eu não conhecia nenhuma... essas ferramentas eu não conhecia nenhuma." (ALU4.1) "tipo eu não tinha muito conhecimento em C++ aí eu comecei a ver o que era" (ALU4.2) "Bom, é.. antes do curso, antes de passar no SISU, eu não tinha nada assim, eu sabia que existia as coisas, mas eu não sabia o que era uma linguagem, por que que tinha, como era feito, essas coisas..." (ALU4.3) "tanto que eu entrei no curso teve aquele negócio com o Scratch, com o cachorrinho, que é de Python que tem, mas que é outra ferramenta mas que eu não conhecia, eu conheci outras coisas que foram ajudando, mas antes do curso "entre aspas" eu quis um pouco entender o que era a lógica, entender algumas coisas, apesar de que não ficou claro mas quando eu entrei no curso algumas coisas eu já tinha tipo feito" (ALU5.1) "Eu posso citar o [nome da Instituição omitida]? Por que antes da [nome da Instituição omitida] eu tinha feito um curso técnico do [nome da Instituição omitida], mas antes do [nome da Instituição omitida] eu não tinha noção nenhuma em programação, eu cheguei nesse mundo da informática totalmente alheia, meu primeiro contato foi com o [nome da Instituição omitida] e a partir do [nome da Instituição omitida] foi que eu tive essa experiência" (ALU5.2) "Sim, por que eu não vim totalmente leiga, quando eu entrei na [nome da Instituição omitida] e os professores começaram a falar, eu começava a relembrar coisas que eu já tinha visto, e isso foi bom, por que eu pude rever conceitos, aprender coisas novas, por que o curso técnico são só dois anos então é uma coisa bem rápida, e você vê um mundo de tudo, em dois anos" (ALU6.1) "Não, antes do curso eu tive contato com programação uma vez, que o meu primo ele fazia o curso de ciência da computação, então eu já vi ele programando algumas vezes e pelo que eu me lembro, faz anos, faz uns cinco anos eu acho, só um Hello World na tela e acho que não passou disso na época"
	Conhecimentos e habilidades	(ALU1.2) "Em questão de programação, acho que não. A parte do Inglês talvez, por que você tem uma facilidade para encontrar mais coisa na internet." (ALU1.3) "Mas em relação à matéria em si, à programação em si, eu não vejo tanta diferença assim, você ter o Inglês ou não." (ALU1.4) "E em relação à matemática, eu também não percebi muita diferença não. A matemática eu acho que é mais para a questão de computação gráfica, onde a gente vê mais um pouco essas coisas. Mas em programação, o que a gente vê em relação à matemática é mais em alguns algoritmos que a gente implementa, mas não é nada que seja tão importante assim." (ALU3.4) "aí eu aprendi a separar meu tempo" (ALU4.4) "aí a partir daí eu comecei a ter mais ou menos um pouco da lógica, e a partir daí

		foi meio que moldando minha cabeça a pensar como estruturar um código, ou seja, o que é que eu vou precisar, como vai ser o algoritmo daquele até chegar o resultado de um problema, como solucionar, aí foi tendo também outros tipos de conhecimento, como estrutura de dados, como o que é uma árvore, uma pilha, uma fila, foi melhorando como estruturar alguns tipos de problemas, estruturar a lógica é importante para resolver aquele problema, como declarar uma variável, alguma coisa do tipo, coisas simples assim." (ALU4.5) "Em relação ao inglês, muita documentação, essas coisas está em inglês. [...] então é necessário, é muito necessário na área ter o inglês". (ALU4.6) "entender o processo, entender até como você melhorar seu código, você deixar uma coisa mais organizada, tipo isso são problemas que a gente não entende, eu pelo menos não entendo, mas tem que melhorar, tipo, entender outras coisas, entender como deixar um programa menos complexo, mais rápido". (ALU5.3) "por que algumas palavras [em Inglês] eu ainda consigo entender, [...]quando são esses erros mais fáceis, que tem uma tradução mais simples, eu até consigo resolver".
	Fatores que auxiliam	(ALU1.5) "Você tem facilidade para procurar, geralmente as informações em inglês na Internet são muito mais completas do que em português." (ALU1.6) "A gente começa com C né? Ou Python, como agora, mas como eu já tinha, por exemplo, eu já tinha uma base na minha cabeça de como funciona um algoritmo mesmo sem nunca ter implementado algo em C, ou em outra linguagem, mas como eu já via muita coisa na Internet, já via o pessoal fazendo, eu tinha uma noção. Claro que não é a mesma coisa que você fazer, né? Mas eu acho que ajudou sim. Apesar que a gente vê pouca coisa de HTML e CSS no curso, mas ajuda sim." (ALU2.2) "Sim! Só para você ter uma noção em questão de mercado, por exemplo, você não tem muita noção por que você está aqui em Mossoró, mas você entrando em um grupo, já tem pessoas com mais experiência, outros professores, eu acho que já dá uma noção para você." (ALU2.3) "Não, acho que não, por que essas ferramentas são mais para auxílio mesmo, questão de atrapalhar, eu acho que na verdade ensinaram mais, por que por exemplo, o visual studio tem muitas extensões então se você for procurar, tem infinitas assim, para melhorar sua agilidade seu desenvolvimento do programa, então acho que na verdade auxiliaram." (ALU2.4) "Eu acho assim, que se eu tivesse estudado antes, com certeza isso teria facilitado, por que eu não perderia tempo aprendendo coisas que eu já sabia. [...] Eu acho que influencia sim, se você tem um conhecimento prévio , se você já entra lá sabendo, acho que com certeza já dá uma ajuda muito boa." (ALU3.5) "Mas é por que lá, o ensino médio dele lá já está sendo as disciplinas na prática já, e eu nunca tinha visto isso..." (ALU3.6) "e ele [o filho] é mais novo com certeza a facilidade de entender e de aprender é mais rápido, né?" (ALU3.7) "eu acho que influencia sim, se você tem um conhecimento prévio , se você já entra lá sabendo, acho que com certeza já dá uma ajuda muito boa." (ALU3.8) "Eu digo a ele: "a diferença de nós dois é que você é só isso aí" (ALU3.9) "Tanto pelo fato como eu já tinha visto um pouco da disciplina, quanto pelo fato de perceber que eu tinha feito daquele jeito e não deu certo" (ALU3.10) ""esse tempo aqui é para eu estudar" aí quando chegava naquela hora, aí eu isolava o mundo, aí foi quando eu comecei a entender direitinho aí deu certo, consegui o êxito e fui aprovado." (ALU4.7) "Assim, em termo de primeiramente em algoritmos e acho que em ICC, que foi Introdução à Ciência da Computação, com PROF7 eu tive uma base, teve um pouco de portugol, um pouco de algoritmos, como fazer um fluxograma que tipo já ajuda a ter uma base de conhecimento" (ALU4.8) "aí teve algum tipo de projeto que foi de final, que foi para mexer com uma biblioteca de reconhecimento facial, que eu esqueci o nome, mas foi em C++, mas era de visualização, isso me ajudou muito, tipo, a ter uma coisa mais prática mesmo de como programar e reconhecer uma imagem... e logo após teve POO com Java que foi mais ainda entender um pouco de polimorfismo, essas coisas, que ajudou." (ALU4.9) "Ahan, ajuda no sentido de tipo entender, em ICC, por que está no curso, às vezes a pessoa não tem um entendimento muito de lógica tipo, ah como fazer um bolo, por exemplo que a pessoa pensa é aquilo, mas tipo não sabe colocar passo a passo, ou seja, se atropelar os passos às vezes interrompe alguma coisa que era importante e que a pessoa não ligava em fazer aquilo. Isso então ajuda a em cada passo estruturar o código" (ALU4.10) "então esse processo como foi feito ajudou por que meio que estruturou uma coisa mais globalmente, de como é feito, para depois ir desestruturando. Tipo, como é uma memória, como é aquilo, bits, essas coisas..."

		(ALU4.11) “Apesar de ter sido uma coisa básica, uma programação básica, antes do curso, aí depois começou a vir cálculo, por que não era programação mesmo no primeiro período, aí depois é que a gente começou a ter o embasamento, mas ajudou bastante sim em lógica, em algoritmos, essas coisas, deu para fundamentar um pouco.” (ALU4.12) “apesar de tipo eu achar que não foi tão utilizada de certa forma até agora, mas acho que ajuda muito você ver a matemática” (ALU5.4) “então quando eu cheguei na [nome da Instituição omitida] eu vi conceitos que eu já tinha visto no [nome da Instituição omitida] e vi outras coisas que eu ainda não tinha visto, então foi um mar de experiências, digamos assim, foi uma outra vivência e eu complementei com algumas coisas que eu já trazia na bagagem.” (ALU6.2) “[...] ao meu ver programação é uma coisa que você tem que gostar, entende” (ALU6.3) “acho que livros, materiais na internet, cursos que eu fiz bastante, me ajudaram a ser um programador melhor, digamos assim.” (ALU6.4) “de outra visão que o professor tinha falado, ele tinha a visão dele e assim vai complementando o nosso conhecimento.”
Parâmetro	**Categorias**	**Unidades de registro**
Quem	Papel do docente	(ALU.7) “aí eu fiquei até o final, mas eu só passei mesmo por que PROF1 foi um anjo com a gente...” (ALU1.8) “[O professor] Foi um pai com a gente [risos]” (ALU1.9) “É foi, foi essa daí. Aí depois eu paguei com PROF4. Aí deu certo.” (ALU2.5) “claro o professor tirando dúvidas e tal” (ALU2.6) “O professor fala: “olha quando você estiver pesquisando, estudando alguma linguagem, um framework, alguma coisa, olha a documentação oficial que lá foram as pessoas que criaram aquela linguagem, aquele framework que fizeram aquela documentação. Então é provável que vocês sanem as suas dúvidas lá.” (ALU2.7) “Em algoritmos tudo o que eu usava basicamente foi PROF5 quem indicou [risos], tudo, tudo, tudo...” (ALU2.8) “Com certeza, eu acho que isso é importante, que todos os professores eles davam: “olha eu uso esta ferramenta, mas não é obrigado você usar ela, é gosto, você pode usar essa daqui...” (ALU2.9) “mas com certeza também ao corpo docente do curso, que é muito bom, auxilia muito” (ALU2.10) “Eu acho que é mais a metodologia e o incentivo. Por que por exemplo, nas disciplinas de programação, por exemplo, professores são muito, como eu posso dizer, amigos.... Você a qualquer momento pode conversar com eles, são aqueles nerds que nem você que tá ali e sabe de tudo, ... eu acho que a proximidade que existe na [nome da Instituição omitida] entre professor e aluno, eu acho que incentiva muito. Não é aquela negócio deu a aula, vai embora, tchau, não o professor continua lá, fica tirando dúvida, incentivando, “ah, você está com algum problema deixe eu saber qual é?!” “professor, to meio perdido, o que é que eu vou fazer da minha vida?”... ai ele vai lá e da uma mão a você, então eu acho que essa conversa assim, é sempre muito bom.” (ALU3.11) “Assim... pronto, o mais importante foi a ajuda do professor no caso” (ALU3.12) “É diferente de você perguntar ao professor, né? Professor: assim, assim... Às vezes é uma besteirinha, né? Uma coisa assim que a gente não percebe, aí, “não, é só isso assim...” e me tirava uma trave grande da vista. Foi muito confortante, muito prazeroso aprender do jeito que a metodologia que eles ensinam.” (ALU3.13) “Ele dizia ‘a ideia é muito boa, agora vamos transformar, vamos fazer o código, vamos programar’. [Quando dizia que não sabia] aí ele dizia ‘mas aí você tem que ir procurando devagarinho, faça do seu jeito, aí eu vou olhar, se não der certo a gente vai tentar achar uma solução para isso’”” (ALU3.14) “então o professor disse: “de jeito nenhum, não senhor, a gente só consegue aprender fazendo, fazendo e fazendo. Se você desistir na primeira tentativa, nunca vai saber se tem capacidade para isso” (ALU3.15) “O que eu sempre falo com os meninos é que os professores sempre ajudam muito, a gente às vezes está com a cabeça pesada, mas quando chegava para assistir aula, via a empolgação do professor, às vezes eu estava lá calado no canto, e vinha e “cadê, você fez?”, e eu “não professor”, “ah, vamos lá, faça aqui você”, aí ele me mostrava onde é que estava e eu digo, então vamos pra frente! [risos].” (ALU3.16) “O que eu acho interessante do PROF1 é que ele fica preocupado, se você está fazendo realmente, se você está aprendendo” (ALU4.13) “E eu acho que é muito necessário o professor instigar o aluno a aprender inglês, tipo não sabe, mas vai procurar essa palavra, o que significa, essas coisas... ajuda.” (ALU4.14) “Acho que o papel do professor foi muito fundamental no sentido de “ah, é normal ter essa dificuldade” “ah, você não sabe, pesquise, vá atras”

		(ALU4.15) “Também até mesmo acho que os professores, não sei se foi pegar leve, mas eu acho que eles viram uma dificuldade no aluno e tentam ajudar, a melhorar, mesmo que o aluno não saiba mesmo, por que tem coisas que é erro de sintaxe ou um pouquinho de lógica, mas o aluno pode ir melhorando...” (ALU4.16) “É, normalmente sempre tem algum retorno tipo “ah, pode tentar fazer isso, ou por que você fez isso, pode fazer de outro jeito...” (ALU5.5) “Acho que a maioria da turma já gostava do professor e começou a se empolgar com a matéria por que tinha um objetivo” (ALU5.6) “Têm professores que eles têm uma didática melhor, que eles trazem exemplos, trazem coisas novas, que estimulam a turma, estimula a você, te dá um ânimo, te dá um gás, e a forma como ele explica, os exemplos que ele traz, a animação dele por si só, você tem um gás maravilhoso para continuar, por mais que seja uma matéria super difícil, mas a gente vai, por que vai com a empolgação dele, a motivação dele.” (ALU6.5) “Mas acho que atrapalhou não, por que ele sugere uma ferramenta que ele gosta, mas se você não gostar você pode trocar, então é muito subjetivo.”
	Papel dos colegas	(ALU2.11) “E eu acho assim que o que mais me ajudou mesmo foi o contato com meus colegas” (ALU2.12) “e quando eu tenho alguma dúvida eu converso com eles, ah, você já programou isso aqui, você já viu isso? Me ajude aí, eu não sei...” (ALU2.13) “e aos meus amigos também aquele negócio, amizade influencia muito né? Então se seus amigos não estão querendo, não estão querendo estudar, isso influencia muito, talvez,... ainda bem que, pelo menos o meu grupo de amigos que assim eu converso mais, eles são também muito estudiosos, gostam muito de estudar.. também gostam de desenvolver, né? O foco deles é esse, então a gente sempre conversa sobre isso, e eu acho que isso colabora muito” (ALU3.17) “mas aí os meus colegas começaram “não homem, o que é isso? Ninguém é, eu lembro muito a frase de [fala o nome de um colega] que sempre diz assim: ninguém é velho o bastante que nunca possa aprender, você vai desistir assim?”, aí eu digo, é, vamos tentar né? Aí pronto e estou até aqui.” (ALU3.18) “que é o pessoal que já tinha bagagem, então lá quando eu estava com dúvida, cedo da noite, eu me pegava com eles.” (ALU5.7) “[trecho omitido pelo entrevistado], por que a nossa sala é bem unida, a gente sempre troca experiências, o que a gente não sabe monta grupo de estudos e vai estudando, o que eu não sei os meninos sabem aí a gente vai trocando ideias, e me ajudam muito, [trecho omitido pelo entrevistado]”
	Papel de outros	(ALU2.14) “Não, [nos grupos e fóruns] é mais dúvida pontual mesmo.” (ALU2.15) “Inclusive o PET tem até o DI nas escolas, né que para falar do curso... Eu acho que é muito importante” (ALU2.16) “eu vim conhecer [o curso de Ciência da Computação] por causa de um amigo meu, que inclusive estuda comigo hoje, no ensino médio, aí dali eu comecei a focar: vou fazer computação, mas eu nem tinha noção do curso em si.” (ALU3.19) “mas aí tanto em casa, a família quanto os meninos lá diziam: “calma, paciência, todo mundo tem um jeito de aprender, cada um tem a sua forma de entender a situação, se todo mundo fosse igual...” (ALU3.20) “As vezes, professora, eu estou conversando com ele [o filho], estou com uma dificuldade, aí ele está vendo algumas disciplinas lá, que consegue me ajudar, aí eu digo “venha aqui, me ajuda aqui”... ai ele diz “está vendo aí, pai, o senhor bacharel e eu ainda sou um aluno de informática” [risos]” (ALU5.8) “então foi aí [nome da Instituição omitida] que eu decidi fazer computação, principalmente a parte de web eu sou apaixonada.” (ALU5.9) “mas eu sou apaixonada por desenvolvimento web, assim, eu não sabia o que fazer antes do IF, tipo tinha vários cursos e eu fiquei: “eu não gosto disso, eu não gosto disso... foi meio que por eliminação” e aí quando eu entrei no [nome da Instituição omitida] e eu comecei a entrar nesse mundo da informática e ver as possibilidades, aí eu comecei a me apaixonar” (ALU5.10) “o Departamento em geral, são bem parceiros, são uma família mesmo, eu acho muito legal a convivência dos departamentos principalmente o Departamento de informática, os laboratórios, são muito unidos, uma família praticamente.” (ALU5.11) “e aí quando foi no segundo semestre eu procurei o auxílio da [nome da Instituição omitida], com a psicopedagoga, e foi muito importante, ela me ajudou bastante, me ajudou nos cronogramas, e com o acompanhamento dela, montando rotinas e tudo, eu consegui dar a volta por cima, consegui dar uma aliviada, e consegui a respirar, e aí minhas notas começaram a melhorar e eu fui me animando com o curso, daí eu falei: “agora vai, foco, e vamos para frente que agora vai dar certo”. E depois desse apoio eu não pensei mais

		em desistir." (ALU6.5) "mas eu sempre tive afinidade por ele [um primo] também ser [da área] de computação, eu acho que isso também me motiva."
Parâmetro	**Categorias**	**Unidades de registro**
Onde	Onde busca informação	(ALU1.10) "é...[pausa] então as duas últimas, em todas as matérias de programação, na verdade em todas as matérias do curso, a gente usa, pelo menos eu usei muito pouco a biblioteca, muito pouco mesmo. Se eu peguei uns 3, dois ou três livros, no máximo. Eu sou muito mais de um PDF, um buscar na internet" (ALU1.11) "então a gente tem que ir atrás, de outras coisas, de internet, essas coisas, exemplos, com certeza" (ALU2.17) "mas eu acho que melhor assim para o aprendizado eram as documentações oficiais, que é uma coisa inclusive que eu acho que os alunos não dão muita atenção." (ALU2.18) "ou então, eu gostava muito de participar em grupos no Telegram, por exemplo, comunidades, grupos de Python, de alunos de outros cursos de Ciência da Computação, então eu acho a questão da comunidade em si é mais, é o que mais ajuda o aluno..." (ALU2.19) "nesse sentido. É claro, tinha vezes que ia pesquisar no Google né? No stackoverflow da vida, [risos]" (ALU2.20) "aí se eles também não sabem, eu recorro às comunidades, que estão mais avançados." (ALU3.21) "A internet está aí para ajudar" (ALU4.17) "normalmente eu pesquiso ou no algum Git, ou GitHub, GitLab alguns códigos, Stackeoverflow, tipo internet mesmo, pesquisar como é que é feito aquilo, sempre tem um material na internet com isso" (ALU4.18) "tipo, tem no stackeoverflow, ah não conseguiu fazer isso, vê lá como os caras fizeram, entende como é que foi feito" (ALU5.12) "alguma coisa que eu tenha mais dificuldade eu vou falando com os meninos"
Parâmetro	**Categorias**	**Unidades de Registro**
Quando	Entrada no curso	(ALU1.12) "Entrei no curso em 2016" (ALU2.21) "Por que eu entrei no curso em 2018. E infelizmente na época a gente estava em greve" (ALU3.22) "Ingressei na [nome da Instituição omitida] em 2015" (ALU4.19) "era para ser no meio do ano, de 2018, aliás no início, só que estava tendo uma greve, então eu só entrei no final do ano, mais ou menos, acho que em agosto." (ALU5.13) "Aí em 2016 fui pra [nome da Instituição omitida] fazer Computação" (ALU6.7)"Entrei em 2018 no curso"
	Período regular – na entrevista	(ALU1.13) "Ah, não eu tô trocando, eu tô no sétimo" (ALU2.22) "Eu estou no quinto, não começou ainda, mas estou no quinto" (ALU3.23) "Pronto, regular eu estou no sétimo, isso". (ALU4.20) "Vou pro quinto período quando as aulas voltarem" (ALU5.14) "Já estou no sétimo [período]" (ALU6.8) "No quinto período"
	UC de programação	(ALU1.16) "É...[pausa], Construção de Algoritmos, [breve pausa, ALU1 pensa um pouco], Programação estruturada, e..." (ALU1.17) "Estrutura de dados!" (ALU1.18) "Também tem programação avançada" (ALU1.19) "POO, programação orientada a objetos" (ALU2.24) "Bom, já paguei algoritmos" (ALU2.25) "estrutura de dados e programação estruturada" (ALU2.26) "e POO também" (ALU3.25) "Pronto, eu já cumpri algoritmos, né? Construção de algoritmos, estrutura de dados e programação estruturada. A próxima seria agora programação avançada" (ALU4.22) "As disciplinas foram algoritmos, programação estruturada, POO, e estrutura de dados." (ALU5.16) "Pronto, de quando eu entrei na [nome da Instituição omitida] até agora eu cursei algoritmos no segundo, na sequencia programação em C no terceiro período, programação em Java no quarto e no quinto, e também vi programação com Python, no segundo, que eu me recordo aqui, foram basicamente esses que eu vi até agora" (ALU6.10) "Construção de algoritmos, programação estruturada, estrutura de dados, e Programação Orientada a Objetos, e vou fazer programação avançada agora"
Parâmetro	**Categorias**	**Unidades de registro**
Por que	Dificuldades enfrentadas	(ALU1.20) "Já a parte de orientação a objetos e avançada já é algo que a maioria dos alunos, inclusive eu, tem mais dificuldades. [...] Além de exigir mais do aluno exige mais do professor também" (ALU1.21) "pelo menos eu acho a nossa biblioteca lá, pelo menos depois da reforma, piorou,

		não sei. Tem muita coisa antiga, muita coisa que eu não achava tão didática assim..." (ALU1.22) "Mesmo quando a pessoa tem alguma base, tem uma dificuldadezinha sim, mas acho que foi mais ponteiros, não tive muita dificuldade." (ALU1.23) Já em avançada e POO, eu acho que a maior dificuldade foi o que eu já falei aqui, que você passa dois semestres vendo uma linguagem, e aí quando chega no outro semestre...[pausa reflexiva]" (ALU1.24) "Objetos, orientação a objetos.. você... é uma matéria que, é uma matéria que pede muito de você para você aprender muita coisa que você nunca tinha visto ainda. E eu acho que tanto avançada quanto POO eu tive bastante dificuldade, não só eu como a turma toda, né?" (ALU1.25) "pelo que eu me lembro eu não estava conseguindo acompanhar muito as aulas do professor [que deu a matéria]. [...] eu fiquei meio desmotivado mas não foi nem a matéria em si, foi o modo como foi dada... Eu acho que sim, a metodologia foi o que mais pesou" (ALU2.27) "C, em estruturada foi C++ e em estrutura de dados foi C, inclusive foi uma matéria que eu tive muita dificuldade por que, como eu vinha do Python, é bem diferente né? O C++ o C." (ALU2.28) "Só que o C++ eu conseguia desenrolar, por que era mais estruturado, já tinha orientação a objetos... o C, nossa Senhora, eu perdi muito tempo tentando entender a sintaxe do que resolver os problemas que o professor passava. Foi bem mais complicado..." (ALU2.29) "eu acho que a maior [dificuldade] que eu tive assim, foi a questão de debugar o código: "Ah, como é que eu vou resolver esse bug aqui? Será que tem uma maneira de debugar? Eu começo linha por linha, eu debugo essa função primeiro?" ficava meio perdido nisso, sabe? Como interpretar por exemplo, aquele erro que apareceu na tela, então as vezes eu ficava assim meio perdido nessa questão de debugar o código." (ALU2.30) "por que eu vejo muitos alunos que "ah, tô querendo desistir do curso, e tal", mas não é nem questão do curso em si, é geralmente ou por que está tendo problema familiar ou não sabe ainda por exemplo a área que quer dentro do curso" (ALU3.26) "por que só a teoria eu ficava "voando" sem compreender mais ou menos o tipo de assunto." (ALU3.27) "mas... aí realmente eu digo que vou ter muita dificuldade, por que eu vou estar do lado de uma turma que já sabe o que está fazendo e eu sozinho sem saber eu vou ter que..." (ALU3.28) "eu imaginava assim: se for começar do jeito de onde o pessoal já sabe, eu vou ficar perdido, eu não vou aprender nada..." (ALU3.29) "Depois da meia noite era mais difícil por que a galera às vezes estava ocupada, jogando, aí eu não queria muito ser aquela pessoa insistente, de estar incomodando" (ALU3.30) "Pronto, a minha maior dificuldade, eu senti dificuldade, e a maior o meu maior problema às vezes ainda até hoje, por que assim, o professor sempre fala que a gente, o bom é pegar um problema nosso, do dia a dia, e tentar achar uma solução para ele. E como eu não tinha nada de programação, nada, eu ficava assim... eu colocava no papel a ideia, mas não sabia como desenvolver essa programação... às vezes eu chegava para ele: "professor, esse assim, assim" (ALU3.31) "Isso! Essa é a maior dificuldade [transformar a solução em algoritmo]. Cada vez que eu conseguia resolver, e conseguia transformar o código, Vixe Maria, parecia que eu tinha ganhado na loteria [risos]" (ALU3.32) "eu não, eu tenho que trabalhar, eu tenho que te sustentar [o filho], as contas para pagar, tenho um monte de coisa na cabeça, são várias situações"..." (ALU3.33) "Então a gente deu uma travada e eu via os meninos com dificuldade, e então eu dizia assim "poxa, eu não consigo ajudar em uma hora dessa, era para mim entender e chegar junto, né?" (ALU3.34) "que teve uma fase que a gente tinha que fazer um aplicativo no final, em programação orientada a objetos, que a gente tinha que desenvolver o aplicativo lá e para mim não deu, e o pessoal também não conseguiu" (ALU3.35) "Foi mais externo. Que eu estava com uma situação um pouco complicada, aí eu não estava conseguindo conciliar as duas coisas, então, no caso interferiu bastante." (ALU3.36) "Eu acho que sim, por que eu acho que se eu tivesse me esforçado um pouquinho mais e tivesse tentado mais, eu tinha conseguido, mas é por que eu não estava conseguindo conciliar o tempo e, por que eu coloquei muito tempo para a situação lá [externa] e deixei o estudo de lado, aí quando foi na hora já estava nos prazos limitados, aí não tinha mais como fazer, não tinha como reverter" (ALU4.23) "Mas o curso realmente, pegar desde lógica o que é uma condição, claro que às vezes tipo não é uma coisa tão fácil de entender" (ALU4.24) "Assim, primeiramente estava tendo até que muitas [dificuldades], digamos assim, até hoje eu acho que ainda tenho uma deficiência em algumas coisas por que falta

		um pouco de experiência, ou seja, às vezes eu tenho dificuldade, eu tenho realmente que pesquisar, acho que tipo é até normal de todo programador" (ALU4.25) "por exemplo eu tenho muita dificuldade em entender hardware, [...] tipo essas coisas mais básicas, com C, com ponteiros, ajudou alguns exemplos dos professores, a entender como era feito essa busca de ponteiro, aquele ponteiro aponta para aquele espaço de memória, esse pra esse, essas coisas, mas isso foi meio que ajudando a entender como é "por baixo dos panos".. tive dificuldade, ainda tenho, mas eu tive várias outras também que, como organização de código, tipo algumas coisas de lógica mesmo às vezes eu tenho esse problema, então eu não sei exatamente dizer qual, mas eu sei que eu tenho vários tipo, mas no início era mais isso de entender realmente como as coisas são" (ALU4.26) "ou até mesmo em relação à IDE, coisa que é mais natural daquela área. Tipo, desenvolvimento de mobile, Web, data science... Uma coisa mais específica, talvez, eu não tenho tanta experiência então a dificuldade que eu tenho é isso tipo, de entender das áreas... entender cada coisa específico." (ALU4.27) "Outras dificuldades foram em relação à matemática por que realmente algumas coisas empencaram, tipo bloqueava, por que tem alguns problemas, por exemplo, Báskara, coisas que às vezes vai muito para matemática e a pessoa não lembra, ou tem que pesquisar, algumas coisas assim, gera dificuldade de desenvolver um código, quando tem que resolver um problema matemático em si, mas realmente tem essa dificuldade também." (ALU4.28) "então hoje em dia, eu ainda tenho essa dificuldade do inglês [...] isso era uma dificuldade mais atrás, mais hoje em dia eu consigo meio que me virar mais com isso." (ALU5.17) "Isso, teve um professor que na programação Java ele indicou IntelliJ, acho que é esse o nome, ele começou usando esse, mas nos laboratórios que a gente usava, os computadores disponibilizados para a gente só tinha o NetBeans, então teve um momento que a gente não estava conseguindo acompanhar muito bem por que ele indicava coisas que no IntelliJ eram ícones diferentes, e aí a gente perdia muito tempo tentando encontrar aquela função no NetBeans" (ALU5.18) "Sim, eu não sei qual é o meu problema, mas um dia eu descubro [risos]... eu não funciono bem sob pressão, então o que é que acontece? Eu consigo fazer as coisas, mas no meu tempo, por exemplo, algumas coisas eu consigo pensar mais rápido, eu consigo obter um conhecimento melhor, ter um aprendizado melhor, mas certas coisas, tipo, digamos assim, na programação estruturas de controle, as estruturas de controle eu tive uma dificuldade por que quando eram coisas simples de fazer eu conseguia fazer de boa, mas teve um certo momento que quando os problemas começaram a ficar maiores, eu me perdia no contexto e não conseguia de jeito nenhum ir pra frente e resolver o problema, então eu acho que minha maior dificuldade é essa, tipo o tempo que é disponibilizado para mim e o meu nervosismo, eu acho que isso tem um impacto muito grande até hoje, né? Sobre a minha vida, digamos assim... eu funciono muito bem , mas quando eu não sou pressionada..." (ALU5.19) "Em relação ao inglês, eu ainda continuo muito leiga, quanto a essa parte, mas aí quando eu faço alguma coisa e o programa me retorna aquele resultado do erro em inglês, eu não consigo ler muita coisa, mas como é... às vezes é um contexto grande que ele retorna, assim "o erro tal, na linha tal, o erro é esse, esse, e esse" algumas palavras eu não consigo entender" (ALU5.20) "por que eu estava no [nome da Instituição omitida], no estágio e tinha acabado de entrar na [nome da Instituição omitida], então eram dois mundos diferentes, sistemas de ensino diferentes, digamos assim, horários diferentes, eu estava muito acostumada com o horário da tarde e da noite, eu funcionava super bem, e quando eu cheguei na [nome da Instituição omitida] foi uma mudança total por que eu tive que mudar de horário, tive que me acostumar com o novo horário, com a nova rotina, então isso foi bem desgastante, então eu lembro que eu falei assim, "meu Deus, eu não vou conseguir dar conta, por que são horários diferentes, são rotinas diferentes", e a faculdade é uma coisa totalmente diferente." (ALU5.21) "e eu ficava: " pra que é que eu vou estudar isso se eu não sei nem onde é que eu vou usar, e aí eu comecei a me decepcionar com a matéria" (ALU5.22) "e ele não tinha uma didática muito boa" (ALU6.11) "Eu acho que, me atrapalhou de alguma forma por que como o PyCharm é pesado, é uma IDE muito pesada, acabou que meu computador deu problema uma vez mas eu acho que foi mais pelo computador, entendeu? Não foi pela ferramenta, foi mais questão de hardware mesmo [risos]." (ALU6.12) "O que eu mais sentia dificuldade em programar era mais quando envolvia questões com aspecto matemático. Por exemplo: faça uma função que use Fibonacci para fazer alguma coisa, então eu tinha que ter um conhecimento prévio em matemático que às vezes eu até tinha visto, mas não me lembrava, então isso tomava mais tempo para eu aprender a função matemática, pra ir depois escrever o código novamente.

		(ALU6.13) “Também sentia dificuldade em algumas bibliotecas que faltavam documentação, sabe? Eu não conseguia encontrar muito inglês internet, mas eu acho que basicamente foram só essas duas.”
	Consequências das dificuldades	(ALU1.26) “No caso do C, que é estrutura de dados que eu aprendi em C, na verdade me motivou mais a tentar entender mesmo ponteiros, por que era algo muito difícil para um aluno do segundo período, mas no segundo período a gente ainda está naquela de ainda está no pique para tentar entender as coisas, então me motivou mais.” (ALU1.27) “Mas em questão relacionada à Java, à POO e essas coisas, eu confesso que até hoje tenho algum trauma de Java [risos], foi o contrário... hoje em dia praticamente eu não mexo com Java, é uma linguagem que realmente eu não sou muito familiarizado, eu acho que boa parte disso foi das dificuldades que eu tive nas matérias de POO, eu acho que com certeza isso.” (ALU1.28) “Desistir não, mas, programação eu não lembro se foi avançada, ou se foi POO, mas eu e quase a turma toda ficou na 4ª prova, precisando de muito.” (ALU1.29) “E muita gente desistiu” (ALU1.30) “Sim, aí eu fiquei desmotivado, e até faltei muita aula. Aí eu acho que basicamente foi isso. Aí eu reprovei... eu reprovei estruturada.” (ALU2.31) “mas tem gente que fica meio perdido, e pensa em desistir do curso, mas nunca passou pela minha cabeça não.” (ALU3.37) “Na primeira semana do cursou deu vontade de desistir.” (ALU3.38) “Por que éee, quando o pessoal começou a falar que seria matemática pura e aplicada, e que seria muito difícil, então eu “aqui não é minha área”. (ALU3.39) “Muitas das vezes eu dizia assim: “não, eu estou no canto errado, não é possível... que o pessoal que já tinha bagagem, que já tinha mexido com programação, resolvia tão rápido e eu ficava lá, como se diz, martelando o dedo, eita, é por que eu estou no canto errado, isso aqui não é para mim” (ALU3.40) “Aí foi onde eu disse: não dessa vez, essa disciplina veio mostrar que o meu limite foi aqui” (ALU3.41) “Pronto, logo no início eu reprovei lógica. O pessoal dizia assim “não reprove lógica” aí eu fui, consegui passar essa fase, aí eu reprovei em programação, programação orientada a objetos” (ALU4.29) “Assim, acho que talvez um aproveitamento delas teria sido melhor se eu não tivesse essas dificuldades, mas de certa forma a dificuldade gera o melhoramento, tipo, do profissional, do aluno, então acho que também é necessário de certa forma, mas se eu não tivesse essa barreira, eu teria ido mais além, teria um melhor aproveitamento por exemplo, ou seja, tipo, eu poderia ter ido procurar mais documentação em inglês se eu soubesse mais, ou como perguntar tipo, tem coisas que às vezes você não sabe, como ah o que é que é isso, tipo, achar aquela resposta, isso ajuda muito, e também a matemática às vezes” (ALU4.30) “às vezes dava aquela queda de cabelo [risos] de você ficar endoidando” (ALU5.23) “já quando eu sou pressionada eu já tive várias experiências que não saíram muito bem por causa do meu nervosismo, da minha ansiedade...” (ALU5.24) “Fiquei muito nervosa, ansiosa, gelada, e isso me atrapalhou muito” (ALU5.25) “eu pensei em desistir logo quando eu entrei no curso, no primeiro semestre” (ALU5.26) “o meu rendimento na [nome da Instituição omitida] as minhas notas do primeiro semestre da [nome da Instituição omitida] foram muito, muito baixas, eu fiquei chocada, por que eu nunca tinha tirado nota tão baixa na minha vida, e aquilo me afetou emocionalmente, e eu fiquei muito triste, por que não estava acostumada, né? Ter que mudar toda uma rotina, então isso me afetou bastante. Então eu pensei realmente em desistir, e então eu terminei o primeiro semestre me arrastando, e aí eu tranquei antes de iniciar o segundo semestre, e então eu resolvi parar e dar um tempo” (ALU5.27) “Vamos respirar, vamos focar e vamos pensar direito... por que na hora do nervoso a gente faz muita bobagem, então eu achei melhor trancar e aí pensar... então eu tranquei, respirei” (ALU5.28) “porque eu desisti uma matéria, eu comecei uma matéria e tranquei ela por que eu não conseguia associar aquilo que o professor estava falando ao nosso cotidiano. e aí eu tranquei, deu tempo nem eu fazer a primeira prova, e eu tranquei já, por que eu não estava conseguindo associar nada a nada, e eu falei, eu não vou continuar nessa matéria por que eu não tô entendendo, e então eu tranquei a matéria”
Parâmetro	**Categorias**	**Unidades de registro**
Como	Como estuda	(ALU1.31) “É... [breve pausa] acho que só o fato de você revisar e ir fazendo as atividades conforme a matéria vai avançando eu não acho muito problema nessas duas matérias.” (ALU1.32) “muito apesar que ainda tem slide o aluno não pode focar muito no slide, por que se focar só no slide em programação avançada ou POO, não vai para canto nenhum”

		(ALU1.33) “Então eu acho que o que ajudou mais na parte de POO e programação avançada foi fora das aulas ir treinando. Além do que o professor passa na sala.” (ALU1.34) “Então, antes de começar o próprio semestre para pagar essas matérias a gente tem que dar uma revisada pelo menos o básico já que sabe que vai ser uma linguagem de programação que a gente nunca usou.” (ALU1.35) “Então acho que você compreender o que a matéria vai dar, os requisitos que a matéria vai dar é o mais essencial.” (ALU1.36) “Acho que essa resposta [de como estuda] vai depender muito da matéria.” (ALU1.37) “eu acho muito difícil estudar sozinho, principalmente por que como eu falei quando você chega nessas disciplinas de POO e avançada, você vê muita coisa nova que você nunca viu, e toda aquela base que você teve em construção de algoritmos e estrutura de dados ela meio que como o próprio nome diz ela é uma base, você fica perdido basicamente, quando você começa as primeiras aulas de avançada e POO e acho que são essas duas matérias que eu mais me aliei a grupos” (ALU1.38) “Mas em relação à construção de algoritmos e estrutura de dados, foi mais sozinho mesmo. Só revisando os slides, e tal, como eu falei essas duas matérias elas são mais voltadas a teóricas, a colocar na sua cabeça como funciona a base de tudo.” (ALU2.32) “Aí se não tivesse alguma coisa lá, eu recorria: “professor isso aqui, como é que faz?”” (ALU2.33) “Bom, na disciplina de algoritmos eu tento pegar o que o professor passa em sala, e procurar algum problema real assim para tentar resolver. Por exemplo, Python: que foi no meu caso com algoritmos, Python ele é muito bom para automação, então por exemplo, fazer um algoritmo que automatiza alguma coisa para o seu dia, por exemplo tudo o que você baixar, na sua pasta de downloads, você faz um algoritmo que pegue aquele programa e organize dentro de uma pasta de acordo com a extensão dele, se é MP3 vai para música, se é MP4 vai para vídeo, e assim por diante, então tentar colocar em prática mesmo aquilo que o professor passa.” (ALU2.34) “depois que você pega a lógica e começa a usar, eu acho que problemas reais são melhores, assim...” (ALU2.35) “Eu particularmente eu gosto mais de estudar só” (ALU3.42) “por que mais que eu, que eu trabalho a noite e tenho uns horários livres, por mais que eu assistisse vídeo aula, eu ficava assistindo vídeo aula, mas não era do mesmo jeito de que ter uma pessoa explicando, às vezes a noite você está assistindo uma vídeo aula, a gente não entende uma coisa, volta, volta, volta, mas não consegue.” (ALU3.43) “Como o período que eu tinha mais tempo era a noite, então cedo da noite eu me juntava com os colegas, com os meninos lá” (ALU4.31) “É assim, tipo... quando o professor passa algum assunto normalmente eu tento entender como é que ele fez” (ALU4.32) “e também pronto às vezes quando o professor disponibiliza algum tipo de problema ou algum código feito eu tento entender como é que é feito aquela estrutura, e também tentar resolver outros problemas.” (ALU4.33) “essas coisas, ajudam um pouco a ter essa lógica. Tipo, praticar para desenvolver melhor, então geralmente eu faço isso: resolver o que o professor passa, pesquisar fora.” (ALU4.34) “Assim, acho que tem que ter um pouquinho dos dois, por que ajuda tipo, grupo eu gosto por exemplo, eu não tenho alguma coisa, está faltando, tipo eu não tenho aquele conhecimento, e alguém sabe, eu tento perguntar “ei, você já fez isso” ou alguma coisa assim, e a pessoa tipo dá a experiência dela, então dá uma ajuda, tipo, digamos assim a ter um conhecimento a mais. Mas também eu acho que a prática individual ajuda por que às vezes também você tem vergonha, não sabe como perguntar, ou alguma coisa do tipo, e isso ajuda ah, como fazer isso?, parece ser besta mas é necessário. Então eu acho que um pouquinho dos dois é bom, um meio termo.” (ALU4.35) “tenta fazer, tipo prática, eu tento sempre melhorar, tentar praticar, pelo menos entender um pouco daquilo que o professor passa, pelo menos nas que já passaram”. (ALU5.29) “Eu particularmente acho que os exercícios me ajudam muito, quando eu começo a estudar alguma matéria eu gosto muito de fazer resumos, durante a aula ou durante os momentos que o professor está lá ensinando, e eu gosto muito de fazer os meus próprios resumos, e isso me ajuda bastante também, é uma metodologia que serve muito para mim, eu não funciono muito com PDF essas coisas, por que eu digo aos meninos la da sala: “eu sou meio que das antigas”, o meu método de aprender mesmo, é escrevendo é lendo, é fazendo os meus resumos, então acho que os meus resumos, aliado com várias práticas de exercício acho que isso me ajuda muito, a fixar os conteúdos e a fixar a matéria de uma melhor forma.” (ALU5.30) “Vão dos níveis fáceis até os mais difíceis e a gente vai fazendo conforme a gente vai desenvolvendo.”

		(ALU5.31) “me ajudou por que como eu falei anteriormente, meu aprendizado é dessa forma: teoria, exercício, e eu vou praticando.” (ALU6.14) “Eu costumo assim, prezo muito prestar muita atenção na aula de programação e aí eu reviso o conteúdo, aproveito que eu já tenho prestado muita atenção, reviso o conteúdo, e aí eu vou praticar da minha maneira, vamos supor, vou fazendo uns projetinhos mesmo individuais, vamos ver se eu acerto um número aleatório que a máquina gerou... aí eu fico treinando desse jeito e funciona pelo menos para mim, colocando a mão na prática, né?”
	Que ferramentas usa	(ALU1.39) “a gente usou [o URI] somente nas duas primeiras, estrutura de dados e tal. Achei pra disciplina um site muito bom” (ALU1.40) “Éee... No começo, acho que nas quatro de programação a gente usava [as ferramentas], eu usava mais o indicado pelos professores. Por que tem aquela questão principalmente em POO e avançada que as provas são no computador la da [nome da Instituição omitida] né?” (ALU1.41) “então tanto avançada quanto POO foi no Eclipse, eu não lembro se foi no Eclipse, ou se foi no [breve pausa] esqueci no nome agora... mas eu acho que foi no Eclipse” (ALU1.42) “Netbeans, eu acho... É, foi o Eclipse ou o Netbeans... Já em estrutura de dados e construção de algoritmos a gente usava o Dev, o Dev C++ né, que é mais pra C, essas coisas...” (ALU1.43) “Hoje em dia eu uso muito o Athon, para usar com frameworks.. por que hoje em dia os projetos que eu faço a maioria envolve framework, essas coisas, e Dev C++ é mais para C, essas coisas” (ALU1.44) “Sim e é muito acadêmico... nem tem versão para Linux, nem nada eu acho. Para algumas [inaudível] do Linux. Tanto que eu lembro que nos laboratórios a gente usa o Geany que vem nativo no Debian. Mas enfim hoje em dia eu uso mais o SublameTex que eu acho que é o principal”. (ALU1.45) “Mas eu gosto de eu mesmo fazer a estrutura pronta. Eu não gosto muito daqueles IDEs que você já começa a escrever e ele já completa tudo, por que eu fico meio perdido. Eu gosto muito de usar o SublimeText por que ele funciona muito bem com o framework o jungle, angle, essas coisas.” (ALU1.46) “Ah sim, ajudou sim.. é por que assim, na época, na época pelo menos pra Java eu acho o Eclipse bom. Acho que é um dos melhores mesmo, mas é por que eu não sou muito ligado em Java mesmo, pelo menos hoje em dia. Mas na época das matérias eu acho que ajudou muito sim. Java é uma programação, é uma linguagem que você tem que... ela ajuda muito, mas você tem que fazer muito também, mas ela ajuda muito, o próprio IDE, o IDE para Java ele tem que ter todas essas funcionalidades para ajudar.” (ALU2.36) “Por que assim, o contato que a gente tinha com resolução de problemas mais nessas matérias, pelo menos no meu caso, era o URI, aquele de juízes online de questões.” (ALU2.37) “E eu acho que ele é muito importante sim, para o aluno ter uma noção de, ée, fazer um algoritmo, resolver um problema, mas eu acho que depois de um tempo começa a ficar maçante, e aquilo, ele é bem acadêmico, não é algo que você vai usar no seu dia a dia...” (ALU2.38) “É exatamente, inclusive uma coisa muito importante que eu esqueci de te dizer que foi ter contato com o sistema Linux, eu acho que conta muito, tem gente que não acha tão importante e tal, mas eu acho que você usar aquele sistema ali já é mais fácil que você usar o Windows, eu acho. Por exemplo, terminal esse tipo de coisa, eu usava muito para estudar, editor de texto eu usava o ATHOM, hoje eu uso o VSCode que eu acho que é o mais famoso que tem hoje no mercado, e acho que de IDE, de ferramentas acho que só isso... Tem um programa que eu acabei esquecendo o nome agora...” (ALU2.39) “Em POO eu usei basicamente só o Eclipse, que é a IDE para desenvolver Java. PROF1 recomendou outras, como NetBeans se eu não me engano, mas a que eu mais gostei assim mesmo foi o Eclipse.” (ALU3.44) “no caso o NetBeans” (ALU3.45) “E pronto, algoritmos era o [nome do professor omitido], ele deixava a gente livre, dizia, “pronto, tem o Visual Studio, tem o NetBeans...” para as ferramentas e ele dizia assim que podia usar a linguagem, estava livre.” (ALU3.46) “Aí pronto, eu usei muito o Visual Studio, o NetBeans também, e outras ferramentas também que sempre tinham.” (ALU4.36) “pronto PROF1 passava o URI” (ALU4.37) “ eu entrei no curso teve aquele negócio com o Scratch,” (ALU4.38) “Assim, no início eu não tinha muito meu pessoal, eu não tinha muito essa desenvoltura por que eu não tinha nenhuma opinião formada, aí normalmente o que eu usava era tipo o PyCharm era tudo o que o professor realmente usava por que eu já estava meio que familiarizado com aquilo no curso. Então eu usava esse PyCharm, usava o Android

		Studio, Andriod Studio eu acho que eu só usei tipo uma vez, para Java NetBeans, o Eclipse.. depende muito do que o professor passa," (ALU4.39) "Ou às vezes dependendo se for só, digamos, um editor de texto, às vezes eu uso o VisualStudio Code, por que tipo não precisa exatamente rodar, às vezes tipo, depende da linguagem, eu posso fazer isso por terminal, mas se for tipo usar IDE ajuda, no início eu usava muito PyCharm, por causa de Python, ou Eclipse, por que eu gostava, e era bom por que já ajuda a fazer aquele processo. (ALU4.40) "Aí eu não sei, eu escolheria mais o Visual Studio Code como editor de texto para fazer as coisas por que eu gosto de web então ele ajuda muito com Java Script, e-node, mexer com essa parte, mas se fosse tipo algo com Java, Python, eu talvez escolheria outro como o PyCharm, ou não sei... ou Eclipse, não sei... alguma coisa focada naquele desenvolvimento. Se fosse Android, tipo Android Studio. " (ALU5.32) "Mas eu acho muito útil, foi muito legal eu não conhecia essa plataforma, mas é uma forma muito legal e eu acho que ajuda bastante" (ALU5.33) "Em programação Java a gente utilizou o NetBeans, em Python a gente usou o terminal mesmo dele que é o Idle, em C a gente usou o DEV C++, e em algoritmos a gente usou o PyCharm" (ALU5.34) "mas como a gente é de computação o professor indicou que a gente usasse o terminal mesmo do Python para a gente não ficar muito apegado às IDEs, para a gente usar realmente um terminal, saber como utilizar um terminal na verdade." (ALU5.35) "a gente usou o DEV C++, mas aí eu encontrei outra ferramenta, é que um colega meu me indicou, ele disse que era mais rápido e mais fácil de usar que o DEV C++, e aí eu instalei para ter uma experiência, e aí eu fiz a comparação dos dois, e aí optei por um que eu achei que fosse melhor para o momento da aprendizagem, que era uma coisa.. que era o Falcon, que ele compila os códigos em C++, e aí o professor indicou o DEV C++, eu usei o DEV C++, e aí eu tive uma experiência com o Falcon, eu achei mais rápido, melhor de usar coisa simples né? Algumas bibliotecas ele dava erro então, eu tinha que recorrer para o DEV C++ quando ele não compilava, ficava pulando, intercalando, mas tirando isso ele era bonzinho." (ALU6.15) "Na de algoritmos eu usei Pycharm, que eu vi Pyhton, o PyCharm e o Athon, que na verdade não é uma IDE é um editor de texto mas é muito bom. Também o terminal do Linux eu usei bastante e na programação estruturada, eu utilizei o Athon também, o Athon e o terminal, no caso para C++. E em programação orientada a objetos e utilizei o Eclipse, que foi para Java." (ALU6.16) "Eu sempre pergunto para o professor qual ferramenta ele acha melhor, e eu testo essa ferramenta e se eu não conseguir configurar ela do jeito que eu gosto de programar, eu tento outra, então, acho que PyCharm [nome do professor omitido] me indicou e eu gostei, eu configurei ela do jeito que eu queria. (ALU6.17) e eu utilizava também o terminal por que eu achava mais prático né, que o usar o terminal da própria IDE, mas foi meio que um complemento um do outro. (ALU6.18) O Athon por que como ele é um editor de texto, ele não tem terminal na ferramenta, eu escrevia o código e eu conseguia compilar pelo terminal, então para mim era muito prático né? Você, ele também tinha alguns plugins que você conseguia conectar com o GitHub, com o GitLab, eu usei também o GitHub em todas as disciplinas para colocar os meus códigos. (ALU6.19) E o Eclipse também, por que eu estava usando o Intelligey que é da mesma empresa que faz o PyCharm, e eu já estava me desvencilhando do PyCharm por que eu estava usando mais o terminal, então eu acabei utilizando o Eclipse, que não tem terminal, até tem.. mas ele consegue compilar o código Java bem mais prático do que eu conseguiria pelo terminal, então eu escolho a ferramenta mais pela praticidade que eu tenho, entendeu?" (ALU6.20) "E cursos também que eu faço, que tem muito exercícios, né? Cursos fora da graduação, eu gosto muito da Udemy, aquela plataforma de cursos online, que são bem baratos os cursos, alguns gratuitos, para estudar melhor para a prova."
	Como o professor ensina	(ALU1.47) "Acho que, éh.. nesse grupo dessas matérias de programação eu separo em dois: por que na parte do início da programação, na parte de estrutura de dados e construção de algoritmos eu acho que é algo muito mais didático assim, é algo muito mais voltado para entender a lógica da programação." (ALU1.48) "por que é uma parte que é muito mais prática, é muito mais você tem que sair um pouco da base dos slides e pegar no pesado mesmo," (ALU1.49) "E, não é só slide, não é só.. tem que ter muito projeto, ter muito, ée [breve pausa] muito prazo pra cumprir" (ALU1.50) "Por que por exemplo em programação avançada, nas primeiras aulas tem o PGCC, né?" (ALU1.51) "o que vai apresentar na matéria toda e muitas vezes professor vai introduzindo

		algumas coisas aos poucos," (ALU1.52) "e os próprios professores sabem disso até por que essas matérias tem muito projeto em grupo, justamente para isso." (ALU1.53) "E assim, em POO e em avançada, essas demais, essas disciplinas mais avançadas de programação, é muitos slides que os professores passam são, é só focados só naqueles slides você não vai conseguir muita coisa... por que é... muitas vezes as provas nem são baseadas nos slides..." (ALU1.54) "Eu paguei POO com P1 , e os slides de P1 eles são, tipo, 50% do que ele fala e do que ele pede para a gente fazer" (ALU1.55) "E o professor, ele ensina tudo baseado naquela IDE" (ALU1.56) "Passou um projeto lá para a quarta prova para poder ajudar, e eu acabei passando, né? Eu achava que eu não ia passar." (ALU1.57) "Eu tava achando muito didático demais, sabe? Era só slide, slide, slide..." (ALU1.58) "Isso! Muito teórico. É, no caso... o pessoal até dormia... mas eu acho que basicamente foi isso" (ALU2.40) "por exemplo um algoritmo de automatização, o PROF5 mostrou também algoritmos de sorteio, bubble sort, quick sort, esses algoritmos já bem antigos, mas assim, que eu achei melhor que só ficar naquelas questões do URI, toda vida aquele negócio." (ALU3.47) "mas os professores tanto de algoritmos quanto de programação estruturada, trabalhavam na prática, por que davam o conteúdo mas levavam a gente na prática, e mostravam como fazer, vendo a disciplina na prática mesmo aí foi que me ajudou muito" (ALU3.48) "mas como eu falei, a metodologia de tanto do professor [nome do professor omitido] quanto do professor [nome do professor omitido] me ajudou bastante, por que foi pra prática né? Ele foi, pegou lá desde o início e foi dizendo: "olhe, tem que fazer assim... vamos pra base, vamos pegar de baixo, ninguém nasce sabendo"" (ALU3.49) "Pronto, uma coisa que o PROF1 falou é que programação é você, na prática, você aprende na prática, vai tentando. Aí ele passava uns exemplos na sala e deixava o resto para você fazer em casa, né, pra tentar. E sempre colocava exercícios na plataforma para a gente ir tentando, e incentivava não pegar só os exemplos que tinha na plataforma, mas pesquisar mais por que como hoje tudo é mais fácil, né? " (ALU3.50) "O professor deu algoritmos em Python, mas ele dizia assim: "a gente vai dar em Python aqui por que fica mais, como é que ele diz, ele tem uma facilidade maior para o aluno, mas a língua que você escolher, se você preferir, não professor eu sei que você está dando em Python aqui mas eu tô vendo em C aqui também... eu lhe ajudo do mesmo jeito, não tem problema. Essa aqui eu estou escolhendo essa mas não vamos ficar preso a uma linguagem só" (ALU3.51) "Pronto, é... eu tive uma assim... em programação estruturada a gente fez a primeira prova e na segunda, o projeto final a gente tinha que fazer um projeto, e a gente fez o projeto lá, era em equipe, e teve uma hora que o professor pediu para transformar, a gente tinha feito o código lá, eu não estou lembrando a linguagem agora, e o professor pediu para mudar essa linguagem do aplicativo, e eu não estava entendendo e os meninos também não estavam, né?" (ALU3.52) "todo início de aula ele dá uma revisada no que ele falou antes, ele pergunta, ele vai dando dose, ele dá um assunto e vai perguntando a turma se realmente entendeu, se é isso mesmo, eu acho muito interessante o modo dele dar aula." (ALU4.41) "que também foi importante, e logo após uma coisa mais a fundo mesmo em programação, foi com PROF4, com algoritmos, com Python." (ALU4.42) "Que ajuda mais a ter uma coisa mais construtiva, mais para entender mesmo como é que é a prática daquilo e me ajudou muito por que era uma linguagem mais simples até certo ponto, por que não tem tanto aquele negócio de C, de ponto e vírgula, int, long... era uma coisa mais dinâmica que ajudou a entender o que é um processo, o que é um for, o que é um while, essas coisas, como estruturar" (ALU4.43) "Eu acho que até aí acho que foi tudo que teve durante até agora no curso, essa forma de programação mesmo." (ALU4.44) "Aí pelo menos com Python, eu também gostei por que meio que me ajudou a pensar mais também na lógica mais, e não ligar muito, ligar no sentido daquela coisa mais baixo nível, como em C, colocar um ponteiro, e a parte de memória, você não liga muito para isso, e eu pelo menos gostei de como foi feito por que meio que antes a pessoa pensa primeiro na lógica em termos de Python, já que é dinâmico, ele faz isso só, para depois ter conhecimento mais de C, em estrutura de dados, essas coisas, para entender como é que funciona aquilo, tipo "por baixo dos panos" (ALU4.45) "Até alguns professores passam tipo, algum exercício" (ALU5.36) "Bom, os meus professores de programação, eles utilizam muito a plataforma URI, eles colocam tipo, eles dão toda a teoria e eles colocam exercícios para a gente na

		plataforma URI, e aí são diversificados, né? " (ALU5.37) "Teve uma matéria, que acho que foi estrutura de dados e programação estruturada, os professores estavam usando a mesma linguagem, que era linguagem C, e basicamente um complementava o assunto do outro, então acho que isso foi bem enriquecedor para mim, por que cada um tem sua forma de ensinar, né? Mas acho que os dois juntos ensinando numa mesma linguagem, e mostrando coisas diferentes de como a gente podia usar, eu acho que isso foi muito útil" (ALU5.38) "eu acho que assim, eles são ótimos professores, são bem didáticos, foram professores excelentes para mim, e a forma como eles ensinaram, como eles traziam novos assuntos, novas coisas de programação acho que isso foi bem interessante, e os dois assim em um mesmo semestre, eles conseguiram abrir um leque assim bem grande" (ALU5.39) "mas eu acho que foi bem enriquecedor a forma que eles ensinam, tem uns professores que eles gostam de trazer curiosidades, dizer por que você está estudando aquela matéria, qual o intuito, onde você pode usar aquilo... associando a teoria à prática, eu acho que isso para mim faz muita diferença" (ALU5.40) "mas aí eu paguei a matéria com outro professor, e foi uma experiência totalmente diferente, por que esse professor trazia, dizia tipo você vai usar isso aqui nesse contexto, nesse contexto, e trazia exemplos..." (ALU5.41) "tipo por que tem matérias que você entra assim, não tem muito ânimo do professor e você não encontra funcionalidades, digamos assim, naquilo que ele está falando, você não consegue ver onde vai aplicar, e a matéria acaba ficando chata, ficando monótona"
Parâmetro	**Categorias**	**Unidades de registro**
Quanto	Importância da aprendizagem de programação	(ALU2.41) "Por que basicamente o que motiva estar no curso é a programação" (ALU3.53) "O que eu acho bastante interessante em programar, não sei nem se é realmente isso, é em ajudar, por que quando a gente está programando, você pega uma situação, você resolve, e deixa de uma forma simples para os outros conseguirem obter uma resposta de um jeito mais rápido, é muito fascinante, né? Saber que você conseguiu fazer com que aquela pessoa lá consiga resolver um problema dela de um jeito mais simples. E participar desse grupo "seleto" de pessoas que conseguem resolver problemas para os outros, é muito interessante." (ALU4.46) "por que tem coisas que só com o código mesmo dá para estruturar, resolver aquele problema, aí em programação estruturada também foi melhorando" (ALU4.47) "Não, não, a programação sempre me incentivou a aprender, sempre gostei de desenvolver um código" (ALU4.48) "mas é assim mesmo, mas o importante é tentar, aprender uma coisa nova, sempre é bom" (ALU5.42) "vi um mesclado de tudo e acabei me apaixonando pela área" (ALU5.43) "mas quando ele retorna um erro simples, que aí eu já estou acostumada, né? Treinando, aí a gente já vai conhecendo os erros, aí quando ele retorna um erro que você já viu, e já sabe o que é então..." (ALU5.44) "por que você não fica presa a só uma forma de programar, você fica vendo as diferentes formas na programação, então isso foi muito relevante foi bem interessante" (ALU5.45) "uma experiência nova, e eu acho que isso ajudou bastante por que faz a gente ver que você pode programar de qualquer forma, de várias formas, e certamente você vai escolher a que melhor se adapta à você, digamos assim..." (ALU5.46) "foi bem legal por que eu comecei a me empolgar com a matéria e disse: isso é muito legal!"
	Dedicação e iniciativa	(ALU2.42) "aí eu disse ah, então eu vou dar uma olhada em alguma coisa aqui, que eu sei que tem programação, e a área que eu mais quero assim, que eu tinha uma noção. Só que infelizmente eu fui estudar Pascal, que não serve para muita coisa [risos]." (ALU2.43) "Nunca, nunca [desistir]!" (ALU2.44) "Nem pra quarta, ainda bem... assim eu fiquei perigando numa disciplina de teoria da computação, foi quase, mas deu para passar." (ALU2.45) "Bom, acho que o esforço próprio de você estar ali estudando, direto, recorrendo aos professores" (ALU3.54) "Na prática mesmo, na marra." (ALU3.55) "aí eu ia para a marra mesmo, mas tá dando certo, né?" (ALU3.56) "mas aí quando foi no semestre seguinte, eu consegui e já paguei a disciplina." (ALU3.57) "eu puxei muito tempo para um lado , agora eu tenho que balancear e ver que, como eu digo: "eu comecei o curso, eu tenho que concluir..." e quando eu iniciei eu já sabia da dificuldade então eu tenho que ter o tempo para o curso né? [...] Por que a gente sabe que a gente tem a nossa vida pessoal mas aí quando você entra num projeto, você tem que concluir ele" (ALU4.49) "O resto foi mais fora parte, por mim mesmo, programando em programação

		web, essas coisas fora parte. “ (ALU4.50) “Aí durante esse período [de greve] eu pensei “ah, eu tenho que ver um pouco o que é essa área, tipo entender o que é que tem no curso, tipo o que é uma lógica, o que é programação, tipo uma base, tipo, como fazer isso”.. aí até mesmo eu pesquisei uns cursos, fiz um curso acho que com uma biblioteca para desenhar, e entender como fazer um círculo e como mexer tipo com a lógica” (ALU4.51) “então desde que começou o curso eu tentei focar um pouco mais no inglês” (ALU4.52) “mas conforme o tempo vai passando, eu meio que estou querendo ganhar tempo para aprender, ler documentação em inglês” (ALU4.53) “[se o aluno] não fez direito, chegou perto, mostra ao professor ele pode dar uma dica, alguma coisa assim.” (ALU5.47) “eu até tenho treinado e estudado em uma plataforma, que ensina inglês, mas aí são coisas básicas” (ALU5.48) “então o que eu faço: eu copio e coloco no tradutor, para saber o que ele está dizendo” (ALU5.49) “mas quando é um erro com uma tradução bem grande, aí eu recorro ao tradutor” (ALU6.21) “então no meu ver o que me ajudou muito foi o meu “fora da sala de aula”, o que eu fazia fora da sala de aula, me ajudou mais” (ALU6.22) “Então eu acho que o que eu fazia fora da sala de aula me ajudou muito a ter mais uma, a saber programação melhor mais a fundo” (ALU6.23) “mas desde o primeiro semestre eu comecei a fazer alguns códigos, mesmo por conta própria e eu gosto de programar, então isso não foi um impeditivo por eu gostar, sabe? E ter essa força de vontade de buscar mais conhecimento em programação. Assim com a motivação as coisas fluem de um jeito diferente, ameniza as dificuldades.”
	Perspectivas para o futuro	(ALU1.59) “Meio que eu já tenho alguns projetos, então ou vai ser professor [nome do professor omitido] ou professor [nome do professor omitido] que irá me orientar, por que eu tenho projeto com eles dois, meu PIBIC é professor [nome do professor omitido] e eu tenho um projeto que é relacionado com o Governo do estado que é com o professor [nome do professor omitido]” (ALU2.46) “é inclusive uma coisa que eu quero melhorar, eu disse aos meus amigos que ia tentar criar de vez em quando um repositoriozinho para a gente começar a trabalhar em equipe, por que o mercado não é você individual, é você com outras pessoas” (ALU2.47) “Por que eu quero ser desenvolvedor, então eu nunca pensei não [em desistir]. Até por que se não é esse curso, eu não sei o que eu ia estar fazendo , por que eu só sei fazer isso, basicamente” (ALU2.48) “eu fui decidir praticamente agora no quarto período, a parte de web que é a que me interessa mais” (ALU3.58) “por que eu quando eu fui fazer o curso, eu já tinha feito mais ou menos assim: eu já, eh, a minha tendência era para a parte física, de hardware, para manutenção de computador” (ALU4.54) “eu tento aprender coisas novas, e ir atrás mesmo, tipo coisas que às vezes podem ser boas para o meu currículo, ou pra uma experiência que um dia eu possa precisar disso, ou alguma coisa assim, mas que não fosse minha área assim, por que o que me instigar é estar aprendendo coisas novas, tipo, entender como que é feito um reconhecimento facial, como é que é essas coisas, por que antigamente eu era apenas um usuário comum, não era um desenvolvedor, então a gente aprende muita coisa desenvolvendo, ir vendo como é, e isso me ajuda muito a ir atrás.”

Apêndice P - Unidades de Registro das Entrevistas com os Professores de acordo com os Parâmetros e Categorias

Parâmetro	Categoria	Unidades de Registro
O que	Conhecimentos e habilidades	(PROF1.1) Quando você programa qualquer coisa, tem que saber por que você está programando, por que você está fazendo aquele *insert*, por que está colocando aquela biblioteca lá, por que está declarando aquela variável como um inteiro, um float, coisa desse tipo aí, então tem sempre que está prestando atenção no que está fazendo, por menor que seja o seu exemplo, mas você tem que saber, você tem que dominar o por quê daquilo ali. (PROF1.2) Programação não é decorar comandos, de forma alguma, o nome já está dizendo: "linguagem de programação". Do mesmo jeito que quem sai daqui vai pra França, por exemplo, tem que dominar a língua se quiser viver bem lá né, se quiser interagir com a população e tal, então se você quiser realmente entender o que você está fazendo, você tem que aprender a língua que você está utilizando. Se é C++, então aprenda a falar C++ e a escrever C++. (PROF1.3) Então, durante o percurso o que é essencial é matemática, evidentemente, lógica, tanto a lógica matemática por assim dizer como o raciocínio lógico, e isso tudo evidentemente faz parte da construção de algoritmos, então o aluno tem que ter uma mínima noção de algoritmos, né? Tem que saber a estrutura para poder conseguir lidar com a linguagem de programação escolhida. (PROF1.4) Eu falo isso aí, pode até parecer óbvio, mas tem gente que aprende a programar e vai direto para uma linguagem de programação sem ter passado pelas etapas de algoritmos, você que é professora e foi de algoritmos sabe o que é isso, que lá a gente vê todo o alicerce do que vai ser a programação no futuro. (PROF1.5) Conhecimento de matemática, lógica, ter esse poder de abstração. O poder de abstração não é algo que você vai conseguir lendo, não. Você consegue treinando, (PROF1.6) Olha, é... como você mesma citou aqui o que eu tinha falado antes, essa aptidão de se trabalhar em grupo realmente é algo que no mundo da informática ela é necessária, e ao mesmo tempo negligenciada, todo mundo tem aquela ideia de que o informático, o cientista da computação, ou atividades correlatas, eles são solitários, cada um se fecha no seu mundinho, e vai trabalhar sozinho, que prefere trabalhar sozinho. Mas não é isso não, tem que ter essa mente aberta para poder receber críticas, fazer sua própria crítica, tem que ter aquela humildade de reconhecer que está errado, e de refazer o que tem que ser feito. Aptidão... além dessa daí acho que, não além disso você perceber o problema e abstrair ele, é uma aptidão, não vou dizer "ah você tem que ser hábil em línguas, línguas estrangeiras" apesar da maioria das programações, das linguagens de programação terem em sua estrutura palavras-chave em inglês e tudo, mas é algo que não é necessário você "ah, vou mergulhar no inglês para poder entender", não. Tem aquela palavra ali, *new*, *new* é *new*, não precisa dizer que é novo. (PROF1.7) Aptidão a gente saber trabalhar em grupo, essa é uma das maiores. (PROF1.8) Bom, atenção. Atenção é primordial. Todo mundo sabe que tem, tem aluno que consegue aprender mais lendo, outro aprendendo mais escutando, mas enfim toda forma tem que ter atenção, atenção no que está fazendo. É... deixa-me ver... tem um agora na minha cabeça, mas já me escapuliu. O quê que precisa, né... bom (breve pausa) persistência, programação não é algo que do dia para a noite você vai estar um super programador, não! Tem que persistir, você tem que ser curioso, você tem que fazer seu programa lá, botar para rodar, deu defeito? Vai lá nos logzinhos, mas mensagens ver onde é que está o erro, procura entender o por quê daquele erro, e aí é nesse sentido aí, você tem que ser persistente, tem que ser curioso, e apesar do nome proativo ser um nome que bota muito medo né, hoje em dia, "ah, eu vou ter que ter proatividade para poder me sobressair", mas proatividade é só você pensar no que é que você pode fazer a mais do que é cobrado em sala de aula. (PROF1.9) Então você ser atencioso, você ser curioso... (PROF1.10) Persistente, pronto: é o tripé para você se dar bem, atenção, curiosidade e persistência. (PROF2.1) E aí é o seguinte: sobre codificação tem gente que diz: "professor, é importante eu saber todos os comandos da linguagem?" aí eu digo: "não! Não é importante não por que assim, eu quando eu não sei um comando, não lembro um comando de C++, de java, eu vou na Internet e pesquiso". (PROF2.2) Mas eu digo é o meu jeito de ser, mas eu sugiro a eles que o mais importante de programar, que você sente que está aprendendo a programar é quando você enfrenta o desafio de resolver o problema, e consegue resolver

		(PROF2.3) Eu coloco pros alunos três principais aspectos e coloco uma determinada hierarquia. Eu acho que em primeiro lugar tem que ter a questão da lógica, eles têm que entender o problema, e quais os passos para resolver. Isso aí é a questão de você organizar o raciocínio lógico de atacar um determinado problema. (PROF2.4)Segundo, ele precisa ter uma formação matemática, necessária para, se for o caso, o problema requerer alguma coisa de matemática ele ter a condição de pelo menos identificar "ah, isso aqui requer o conceito de função, ou então, ah isso aqui requer o conceito de geração randômica, ou precisa você usar uma combinação linear que da álgebra resolve o problema" então essa questão aí da fundamentação matemática é importante. (PROF2.5) E por último eu coloco a questão do conhecimento para codificar, né, e eu até digo para os alunos: "é possível ter um programador só com essas duas primeiras e que não saiba codificar uma linha de código, mas ele pode codificar um algoritmo complexo, altamente complexo, com pseudocódigo e passe para alguém que codifique e isso é programação" eles ficam meio assim (risos), mas eu coloco nessa hierarquia: primeiro ter um bom raciocínio lógico, a capacidade de analisar o problema e de estabelecer os passos para resolver; depois se precisar de conhecimento matemático ter a condição no mínimo de identificar o que precisa, por que você não precisa ter tudo na cabeça, "ah, eu preciso aqui do conceito de um determinado processo estocástico, eu consigo identificar que é isso, eu não lembro agora como é o processo, mas eu vou lá na Internet, no livro e vou ver o conceito e trago pra implementar"; e por último a ideia de codificação, né, de saber codificar, de ter o conhecimento de uma linguagem para poder codificar, basicamente são esses aspectos. (PROF2.6) É hoje em dia tá cada vez mais comum a ideia de programação coletiva né. (PROF2.7) então eu acho que é preciso o aluno ter uma boa característica que é necessária no ser humano para tudo né, que é essa capacidade de ter empatia, de trabalhar com o outro de maneira tranquila, sem criar problemas, ter maturidade para isso, outra aptidão também importante é o aluno para aprender programação ter tranquilidade, né? Não se afobar, perceber que quando cansou não bater a cabeça, procurar fazer outra coisa ter aquela condição de analisar, de fazer uma autoanálise e dizer: "não hoje eu não vou conseguir produzir mais nada! Vou fazer outra coisa, depois eu volto", que às vezes, comigo aconteceu muitas vezes, durante o mestrado, doutorado, pós-doutorado... tô lá, tô batendo, tô tentando, não consigo. (PROF3.1) Na minha opinião, primeiro: ele precisa ter uma base matemática muito forte, por que a matemática ajuda a você a raciocinar algoritmicamente. Não é que algoritmo é matemática mas a forma de raciocínio é muito próxima pra você desenvolver, por que os problemas matemáticos você segue regras, a gente segue regras pra resolver, e a gente aplica algoritmos pra solucionar eles, mesmo quando a gente está resolvendo manualmente. Então se você tem uma base matemática forte, você está acostumado a pensar algoritmicamente mesmo que você não saiba que é algoritmicamente. Então uma base matemática sim. A lógica de computação, a lógica computacional, a lógica de Boole essa que se estuda em disciplinas de lógica, eu acho que é importante se ter algum domínio, mas não acho essencial, pra mim o essencial é matemática, mas principalmente, é o modelo de pensar diferente. (PROF3.2) Então eu acho assim, o que eles precisam ter: matemática e uma forma de pensar algorítmica. Se eles tiverem essas duas coisas, eles conseguem. (PROF3.3) Se todos os alunos já chegassem sabendo pensar algoritmicamente na disciplina, nenhum teria dificuldade, por que o conteúdo é simples: é aplicar o que a gente faz no nosso dia a dia. O conteúdo de algoritmo é: eu vou fazer o que? Eu vou repetir ações, eu vou colocar condições nas minhas ações, pra saber se eu faço ou se eu não faço, a gente faz isso toda hora, o dia todo, em qualquer coisa que a gente faz, né, então o conteúdo dele não tem dificuldade, o problema é a forma de pensar, isso é o que eu acho da dificuldade. (PROF4.1) Eu acho que uma das coisas que o aluno tem que ter é uma capacidade de moldar a forma de pensar. Por que as pessoas pensam num programa todo, né? E não conseguem dividir ele em partes, né? Não conseguem pensar num ponto específico. Programar na verdade, você não faz um programa como um todo, ne, você faz, se você for até olhar a questão de algoritmos, se você faz passo a passo ne, e as pessoas elas não, as pessoas têm dificuldade em conseguir quebrar um problema em vários outros menores, até chegar num nível que possa ser codificado, então é complicado você conseguir quebrar esse todo em passos que você consiga codificar (PROF4.2) Mas que conhecimentos, habilidades a pessoa precisa ter... eu não acredito que assim exista algum talento, ou alguma coisa que seja bem específica, tipo "eu tenho um talento natural, eu sou uma pessoa que consegue por exemplo,

		uma pessoa que gosta de ler, uma pessoa que gosta de resolver problemas, uma pessoa que gosta de jogar xadrez..." não, você vê bons programadores independentemente destas características, é claro que pessoas que por exemplo conseguem jogar um jogo como xadrez ou algum outro jogo que você tem que pensar nas possibilidades e tal, o que é que vai acontecer depois, as consequências, tudo, é claro que elas vão ter uma facilidade maior do que uma pessoa que não joga, mas isso não significa que uma pessoa que não joga xadrez, ou que não joga nada no computador, ou que não consegue gostar de matemática, não significa que ela não vai conseguir programar, não significa que ela não vá programar bem. (PROF4.3) Na verdade, são talentos diferentes que podem te levar a ser um bom programador, ou uma boa programadora. (PROF4.4) Mas eu acho que não tem um talento natural, lógica ou nada desse tipo. A gente pode ter programador com os mais diferentes tipos de talentos. (PROF4.5) E até por que hoje não é só a questão intelectual que conta, né hoje a gente tem muitos casos de muitos alunos em que a continuidade ou não no curso não tem nada a ver com a condição intelectual, (PROF4.6) E você tem que ser uma pessoa também meio que autodidata né, tem que ser uma pessoa proativa, por que você não vai conseguir todo conhecimento só em sala de aula, só nos exemplos que o professor deu, (PROF5.1) Pronto... eu acho que a principal competência que alguém tem de entender para aprender a programar é a capacidade de estruturar a resolução de problemas. (PROF5.2) Então essa capacidade de expressão matemática, essa expressão lógica, de análise simbólica e de interpretação de texto, eu acho que é a principal competência que o aluno precisa ter: é conseguir ler e decompor o problema. (PROF5.3) Então "digite alguma coisa"... então a gente vai fazer um input, eles sempre querem fazer um output primeiro, ou seja, e aí é por isso que eu considero que há a necessidade de entender o computador. (PROF5.4) E entender o ambiente de execução de programação, entendendo a interação em um Shell, por exemplo, em um bash da vida, (PROF5.5) primeiro, ele tem que ter uma capacidade de resolução de problemas, separar o que é entrada do problema, decompor em variáveis, e sistematizar o que é uma saída que o problema está especificando e implementar um processo de resolução. Segunda coisa: ele tem que saber, no âmbito de programação em si, instrumentalizar o computador.
	Fatores que auxiliam	(PROF1.11) se você começa a ver exemplos de problemas e tenta passar esses problemas para uma formulação lógica matemática ou para um modelo, uma maquete, um desenho, qualquer coisa, então você já está exercitando esse poder de abstração, e isso é uma competência muito boa para quem está programando. (PROF1.12) É, você tem que estar confortável, então o que é externo que gera desconforto, vai te atrapalhar. (PROF2.8) é claro que professor não só pega o filé (risos) é muito bom quando eu pego um grupo de aluno empolgados, aqueles que têm facilidade já com esses pré-requisitos, (PROF2.9) Esse é o termo, o aluno não se afobar. (PROF2.10) Eu acho que, independentemente do conteúdo, é claro que programação tem uma complexidade maior, e por isso requer um fator maior disso que eu vou dizer, eu não acho que um aluno ele não consegue aprender sem estar concentrado na aula, né, sem estar envolvido, sem estar... (PROF2.11) mas assim eu acho que precisa foco, concentração, precisa estar ligado ali, né, precisa estar bem, se o aluno não estiver bem de saúde ele não vai aprender, precisa estar bem psicologicamente, fisicamente, mas precisa estar focado no que o professor está fazendo, está falando. (PROF2.12) Normalmente um aluno que tem muita facilidade, às vezes nunca programou, mas tem facilidade em lógica, em programação, ele é um aluno intelectualmente vamos dizer assim mais robusto. (PROF2.13) Mas assim, há alunos que tiveram, eu tenho um caso, vários casos de alunos que foram meus alunos de matemática no ensino médio, e aqueles alunos que eram extraordinários em matemática no ensino médio, são pessoas que chegaram em computação e deslancharam, né? (PROF3.4) Então eu acho que dedicação é o, seria a ação que eles podem fazer pra aprender algoritmos. (PROF3.5) Tem uma coisa por trás de tudo que eu não sei lhe dizer o que é, mas eu acho que é mais aptidão, tem uns que vão chegar e se fizer uma hora por semana de algoritmos, eles vão aprender algoritmos, tem uns que vão passar sete dias por

		semana estudando duas horas e vão ter dificuldades de aprender, mas aí eu tô dizendo aptidão por que tem gente que tem aptidão pra isso, mas tem outros que existem alguns problemas por trás disso... quem tem aptidão aprende fácil, mas existem problemas que atrapalham, eu acho que você vai perguntar sobre isso, vou deixar pra frente (risos). (PROF3.6) Bom vamos lá, primeiro vou falar dois que eu vejo que são importantes hoje em dia: primeiro é a estrutura universitária. Se a Universidade além de ter a disciplina, o conteúdo, a disciplina conteudista que passa o conteúdo, tem também que dar a estrutura para que o aluno se integre e ele consiga praticar dentro da Universidade. A gente consegue, eu creio que a gente tem estrutura pra isso, a [nome da Instituição omitida] por mais que ela não seja a melhor estrutura do mundo, mas eu acho que ela tem estrutura e dá a possibilidade ao aluno que quiser e puder ficar lá e executar, e treinar, e fazer, e estudar, não só algoritmos, mas qualquer disciplina. Então, primeiro é a estrutura, o ambiente universitário é importante. (PROF3.7) E o terceiro fator também de ambiente é o ambiente familiar. Hoje a gente vive num mundo muito problemático, então muitas vezes o ambiente familiar é importante pro aprendizado de algoritmos mas pro aprendizado geral, então não é só de algoritmos, mas pro aprendizado em geral. Ele precisa ter uma boa base estruturada da família pra que ele possa chegar na Universidade, pra que ele possa ir pra Universidade, não adianta nada a Universidade ter um ambiente, ter a estrutura, e o cara não puder por que ele tem que ficar em casa, por que ele tem que dar conta da família, tem que trabalhar pra ajudar no salário do pai, e alguma coisa desse tipo, então ele tem que ter uma boa estrutura familiar pra que ele possa chegar e trabalhar da melhor maneira possível as disciplinas, e a gente vive hoje num mundo em que problemas psicológicos, problemas familiares, muitas vezes hoje se sobressaem sobre as demais atividades e nas atitudes das pessoas. (PROF3.8) E assim, se a gente for falar do nosso curso, a gente tem um ambiente que oferece tudo isso, a gente tem professores comprometidos, tem professores que interagem com os alunos, que estão lá conversando, que estão orientando, a gente fornece a estrutura deles, (PROF3.9) o que eu percebo é que os alunos que se dão bem, eles partem pra cima, vão terminam o curso, terminam as disciplinas no período correto (PROF4.7) É, primeiro ele tem que ter uma força de vontade, uma persistência, uma perseverança, por que programar não é algo natural. (PROF4.8) mas eu acho que dessas duas são as mais importantes a questão da perseverança, de não desistir por que por que você tem um processo e às vezes a curva de aprendizagem é rápida e às vezes não, às vezes é lenta, então você tem que ter essa perseverança, e também ser proativo pra não ficar só esperando o conteúdo ou não se contentar com aquele conteúdo que é passado só na aula, na sala de aula e nos exercícios, mas ficar procurando mais coisas. (PROF4.9) E de alguma forma você tem que ser, são duas coisas que são totalmente opostas, ne? Que é você ser disciplinado, por que você quando programa você tem uma forma de fazer e você tem que seguir aquela forma de fazer, e você pode ter milhões de ideias, mas se não fizer de uma forma você não consegue, e ao mesmo tempo você tem que ser criativo, que os problemas sempre vão surgir e pra cada problema pode ter uma solução ou soluções diferentes, então você tem que ter sua parcela de criatividade. São duas coisas totalmente opostas mas você também tem que ter né? (PROF4.10) A disciplina de conseguir seguir uma forma de implementar, por exemplo, a sintaxe da linguagem: você tem que seguir, a sintaxe da linguagem, e tem que ter a disciplina de ficar "ah, vou sempre fazer dessa forma, vou criar um if vou sempre usar chave que é para não me esquecer" então você tem que ter uma certa disciplina pra seguir todos aqueles padrões de projeto, aquelas recomendações que o pessoal usa, que funcionam né, queira ou não queira, se funciona então você tem que ter uma certa disciplina pra seguir alguns passos, mas ao mesmo tempo você tem que ter a criatividade de propor soluções ou achar uma solução para os problemas que sempre vão existir. (PROF4.11) é o tempo que você tem disponível, é, os recursos que você tem disponível, e o conhecimento. Né, ou seja, se você tem um grande conhecimento, você pode fazer em pouco tempo, com poucos recursos, né? (PROF4.12) tem a questão dos recursos, como eu falei, que um, um, se ele tiver uma boa máquina, uma conexão à Internet boa, se tiver bons livros, né? Você tiver acesso a material, então uma pessoa dessas tem tudo na mão, né? (PROF4.13) Ou seja, só não se desenvolve, digamos assim, se não quiser. Então é

		interessante, e a questão do tempo, né? A pessoa tem que, se a pessoa tiver tempo para praticar, tempo pra estudar, é.. é uma coisa muito boa, porque, assim é um, normalmente programação exige dedicação, né? Exige um tempo maior pra você pensar, pra você implementar, testar e tudo, (PROF4.14) então se, se, e ai no caso, os alunos, aí falando mais uma coisa mais questão mais social digamos assim, às vezes o, se o aluno tiver condições de permanecer na Universidade, né? Ficar lá no laboratório, né, passar uma tarde lá, (PROF4.15) E eu acho que é isso, né? Assim, então não existem fatores, não existem características muito fixas que a gente pode dizer: ah você vai se dar bem ou não vai se dar bem, ou você vai se dar mal por causa disso, ou se deu bem por causa disso, então quanto mais a gente tiver esse contato, mas tiver a questão de capacitação, a questão de metodologias, isso vai ajudar os alunos a conseguirem aprender mais, na área de programação. (PROF5.6) Então quando a gente está na presencialidade, a gente tem ferramentas muito mais definidas, interfaces bem claras, a própria estrutura... expressando melhor nós temos a padronização das ferramentas e que isso acaba auxiliando na padronização dos processos. O que também possibilita que quem se desvia desse padrão, a gente consegue identificar com mais clareza, os níveis de dificuldades dos objetivos educacionais traçados para aquele momento daquela experiencia. (PROF5.7) o quanto antes os nossos alunos eles tiverem contato com a instrumentalização prática e os seus conteúdos melhor.
Parâmetro	**Categorias**	**Unidades de Registro**
Quem	Papel do docente	(PROF1.13) Por exemplo, não é por que você chegou na Universidade "ah eu tive uma base matemática muito deficiente, e tal..". Não, não é por isso que você vai se dar mal. Reforce a sua base. Então, acredite em si próprio, por que afinal de contas se você não acreditar em si próprio, ninguém vai fazer isso por você, né? (PROF2.14) e eu digo isso sempre para os meus alunos: "olha, se você não souber resolver um problema, você não sabe programar para que o computador resolva". (PROF2.15) eu costumo brincar com os alunos dizendo isso (risos), né porque você raciocina pouco e escreve muito. (PROF2.16) Dá mais trabalho pro professor, do que você chegar e fazer assim: "vamos fazer uma calculadora, ta aqui!". Ele escreve, tem aqui o cabeçalho que eu mandei... aí o menino até faz e sai animado com a calculadora. Mas se no outro dia você chegar e pedir para ele fazer outra coisa ele tá perdido. (PROF2.17) E faço o maior alarde pra fugirem de códigos prontos. Nada de procurar código na Internet, que isso vicia e faz com que você fique com preguiça de pensar. "pegue o problema, tente escrever numa linguagem que você já conhece a codificação e resolva. É claro que nem sempre o programa roda de primeira, né? Se rodar é por que o nível do problema está fácil para você procure outros mais difíceis" (risos), então é assim. (PROF2.18) mas as vezes não tem e o grande desafio do professor é com aquele que não tem, aí às vezes eu passo listas diferenciadas de revisão, aí eu digo "pronto". (PROF2.19) Agora a gente cursou uma disciplina, ministrei essa disciplina de metaheuristica e tinha uns alunos aí eu passei uma lista, alguns entregaram, outros estavam com dificuldade, aí eu fui passei uma outra revisão, mandei em particular para um grupo de alunos, aí eu disso: "olhe, esqueçam essa revisão, faça essa", aí eles fizeram, aí eu "agora peguem a outra" aí começou a fazer uma espécie de degrau, né, para ir facilitando. (PROF2.20) no começo essa ferramenta não funciona, por que aí eles vão, primeiro: ou não vão me dar atenção e eu vou ficar muito preocupado com o que eles estão fazendo, ou vão se voltar só pra mim e não vão fazer o que deveriam. Então é uma ferramenta que você deve usar em um determinado momento que você sentir que a turma pegou ritmo, né? (PROF2.21) Para parar e tentar, buscar ajuda, às vezes é preciso, às vezes você não vai conseguir resolver sozinho "ah, tenho um colega que tá trabalhando esse mesmo problema" e eu estimulo muito, quando eu tô dando essas disciplinas eu digo: "olhe, enganchou, teve um problema, procure o colega lá, não é pra você copiar o código dele mas é pra discutir, e tentar resolver, buscar ajuda". (PROF2.22) E aí, eu sei que você não perguntou esse outro lado, mas eu vou aproveitar e dizer, aí como professor eu preciso fazer com que o aluno se interesse pela minha aula. Minha aula tem que estar minimamente organizada pra pender o aluno, né? Tem que estar minimamente interessante para que ele tenha esse foco, senão, se eu não tiver organizado a aula que dê uma sequência lógica, uma compreensão o entendimento que está sendo construído o conhecimento para o

		aluno, vai contribuir para que ele se disperse com certeza. Então essa parte de planejar é importante. (PROF2.23) Pronto, [nome do aluno omitido], primeira aula que eu dei de lógica pra [nome do aluno omitido], eu passei uma atividade que ele veio me mostrar no caderno, ele escrevia normal no caderno e pelas bordas laterais do caderno, por baixo, por todo canto. Aí eu olhei e disse: "[Nome do aluno omitido], eu vou monitorar seu caderno durante essa disciplina todinha. Se você se organizar, você tá vendo essas duas linhas laterais aqui? Essa de cima e essa de baixo? Isso aqui delimita a área de escrita do caderno. Se você se organizar pra fazer seu caderno vir bem arrumadinho, com as atividades, com tudo nessa faixa aqui que é de escrita, eu vou lhe dar meio ponto por unidade de lógica, por que isso aqui tem a ver com a organização do raciocínio lógico. Se você conseguir fazer isso, no final do semestre me mostre o caderno que, se couber, pode ser que você tire 10, mas se couber eu te dou meio ponto na disciplina, na prova". Aí ele fazia isso, e vinha todo orgulhoso: "professor, olha aqui". Na primeira vez o negócio todo organizado, e a letra dele era esquisita, né? Aí quando foi na segunda, ele disse "professor eu tô querendo te dizer um negócio, mas eu tô com vergonha" aí eu: "que foi?" (risos) "eu comprei um caderno de caligrafia" (risos). (PROF2.24) é importante que o professor saiba identificar isso no aluno pra não se fazer com que se torne ainda mais forte essa sensação, então você precisa ter o jogo de cintura pra tratar com esses alunos de maneira que ele consiga se sobressair e se motivar. Por que imagina você cursar a disciplina duas, três vezes, na terceira vez, se o professor tiver uma palavra meio infalsa em relação ao aluno, é sujeito o aluno pirar, né, surtar, ou se chatear, desistir logo, então é preciso... eu quando percebo que o aluno é repetente numa disciplina, eu já fico meio cuidadoso com o aluno, eu não trato ele de maneira diferente do ponto de vista de fazer é, de avaliar, mas eu tenho um tato na forma de tratar pra que o aluno não se sinta inferior ou não se sinta que não tem capacidade, então isso pode no decorrer do tempo fazer com que ele se sinta incapaz. (PROF2.25) Ou uma palavra mau dita de um professor, ou o fato dele ter acumulado reprovações, pode fazer com que ele tenha um complexo de inferioridade, se sinta menosprezado... (PROF3.10) eu não limito eles, (PROF3.11) mas por exemplo, quando eu passo trabalho, ou quando eu faço as avaliações, eu permito que eles me entreguem na linguagem de programação que eles queiram entregar os trabalhos. Por que o intuito é esse: aprender a programar, não aprender uma linguagem. (PROF3.12) E eu digo que reproduzindo o que o professor faz eles nunca vão aprender. (PROF3.13) Tanto que é uma coisa que como eu jogo muito aberto sempre com as minhas turmas, eu digo muito "gente, é a consciência de vocês, é o que vocês vão precisar, vocês vão trabalhar isso, vocês estão aprendendo aqui pra precisar disso pro resto da vida de vocês, então não adianta vocês se enganarem e passar nas disciplinas dizendo que sabem, mas uns têm essa consciência e outros não têm... (PROF3.14) Ele tá se enganando eu digo muito: eles nunca me enganam, eles enganam a eles mesmo porque eu sei quem fez e quem não fez. E é injusto você cobrar só por prova, porque na prova tem aluno que trava e não consegue, e você coloca um exemplo um pouco mais complexo que ele tem uma forma de raciocínio um pouco mais lenta que os outros, e ele até vai conseguir fazer, mas na hora da prova, no tempo de prova ele não consegue, e aí você fica injusto cobrar, tem que ter trabalhos, tem que ter atividades, aí complica um pouco. (PROF3.15) Então isso é uma atitude que eu não tem perigo de eu não repetir isso duas, três vezes a cada semestre que eu leciono: "Não ache que vocês estão aprendendo comigo fazendo, vocês têm que aprender com vocês resolvendo, por aqui eu tô desempenhando o meu raciocínio, e não é um raciocínio único, cada um pode ter um raciocínio diferente, você tem que desenvolver o seu para resolver os mesmos problemas que eu resolvi usando esse aqui". (PROF3.16) A gente vive num mundo hoje muito competitivo, e que às vezes as pessoas adquirem um conhecimento e eles não querem repassar pros outros por que acham que você vai ser concorrente, então eu acho que desenvolver um melhor ambiente universitário académico onde eles possam trabalhar isso aí de uma maneira mais concisa, eu acho que é importante. (PROF4.16) mas eu não costumo passar vídeos por que na minha concepção programação é prática então se eles não praticarem eles não vão conseguir fazer,

		(PROF4.17) e tanto é que isso motivou até a forma como eu cobrava os alunos, então antes os trabalhos eram mais difíceis, em menos tempo, e na verdade isso não estava incentivando os alunos a persistir, a procurar, isso estava desmotivando, fazendo mais com que eles desistissem do que tivessem força de vontade pra continuar. (PROF4.18) Contato com o professor, acho que é mais uma questão dos alunos procurarem, não sei se os alunos têm medo (risos), de procurar os professores... não sei, de às vezes procurar, perguntar... e assim, eu nunca tive problema de um aluno mandar, mesmo que seja por e-mail ou por mensagem, por esses aplicativos, de tirar dúvidas, de indicar material, (PROF4.19) Temos o monitor da disciplina, inclusive eu tenho, costumo colocar monitores nas minhas disciplinas, justamente para aqueles que têm medo de perguntar ao professor, ou que não têm tanta intimidade, ou que se sentem mais a vontade com o monitor, eles podem procurar ajuda deles. (PROF4.20) e não que seja papel da Universidade fazer isso, mas a gente também tem que se preocupar com isso, ou seja, de alguma forma mostrar que aquilo daqui é aplicável, mostrar que aquilo dali pode ser usado por ele no mercado de trabalho, né. (PROF4.21) Não só limitar ele a minha aula e aos livros da disciplina, mas também dá essa indicação a eles de materiais que possam ser úteis para ele durante as disciplinas, durante o estudo deles. (PROF4.22) Eles estão conscientes destes problemas que não são necessariamente intelectuais, né, e das dificuldades intelectuais também dos alunos em relação ao conhecimento prévio para chegar na disciplina, e os professores, eu digo do meu caso, e também dos nossos colegas, os professores têm essa consciência, né, eles têm essa noção de toda dificuldade, das dificuldades que os alunos enfrentam, né? E têm tentado criar, digamos, achar soluções. (PROF4.23) por que antes, é uma coisa que a gente observava, que eu observei, por exemplo no meu período de graduação, era que você tinha professores que: "olhe o assunto é esse e estude, dê seu jeito, se vire, e se não conseguir, próximo ano tem de novo, é igual a carnaval, ne todo ano tem" (risos), e era um, e assim quando a gente se forma, e vai ser professor, a gente normalmente tende a repetir tudo aquilo que a gente foi ensinado, então, mas isso não é uma coisa que se observa muito, principalmente com os professores de programação, então, principalmente os nossos professores, eu digo no meu caso, e dos nossos colegas, então, a gente está sempre tentando achar novas metodologias, usar novos procedimentos, tentar diminuir esses problemas reduzir esses, ou deixar o máximo possível, que os alunos tenham o máximo possível de aprendizado, mesmo que não seja aquilo que a gente imagina que deva ser o adequado, ne. Por exemplo, ah o aluno tem que aprender 70% da matéria, mas aí você tem é, que o aluno aprendeu 50% bem, então já é um ganho, já é um avanço, né? Então os professores estão conscientes disso, e têm feito, têm tomado atitudes pra que isso, pra que esses problemas sejam reduzidos, mitigados, usando uma palavra mais bonita, mitigar os problemas, né? (PROF5.8) e eu até coloco uma coisa importante na formação de Construção de Algoritmos, é que é preciso separar o que é um programador do que é um desenvolvedor. (PROF5.9) Então entra, por exemplo, uma coisa que eu tô fazendo recorrentemente nas turmas é, há um curso gratuito na Udemy de introdução ao Git e GitHub, que é pra iniciar o processo de versionamento. (PROF5.10) e isso eu abordo na disciplina de Construção de Algoritmos e fico um pouco frustrado por que eu não consigo fazer com que eles entendam essa necessidade de entender o ambiente de execução dos programas.
	Papel dos colegas	(PROF1.14) Então, fatores externos, bom além da ajuda dos companheiros, né, dos amigos. (PROF2.26) Mas pra isso é preciso um nível de maturidade bom da turma, né? Eu só consigo fazer isso quando eles já estão fazendo, quando está todo mundo mais independente, chamando "professor só arrodeie aqui, pra ver aqui se deu certo"... (PROF2.27) Em todos os trabalhos que eu fiz, de pós, de especialização, mestrado, doutorado, pós doutorado, pós doutorado foi menos, mas até no doutorado, era um trabalho no laboratório com um grupo, em coletividade, que quando eu tinha dúvida eu ia lá e perguntava, e eles vinham me perguntava, e a gente trocava figurinhas, e além de fortalecer a amizade ajuda bastante. (PROF3.17) Eu disse falar dois, eu vou falar três, por que eu disse ambiente universitário, mas eu vou falar de dois ambientes: então o ambiente universitário também é importante, não só a estrutura, mas a turma, os colegas, se eles se envolvem, se eles não se envolvem, o quanto um é capaz de ajudar o outro, o quanto

		eles permitem que o conhecimento que adquirem possa ser repassado pros outros, então esse ambiente universitário é muito importante. (PROF4.24) Então por exemplo, que fatores podem auxiliar: a questão dos amigos, né? Dos colegas de turma, isso eu acho que é uma coisa muito importante por que nem sempre a pessoa é boa em todos os assuntos e quando você tem um colega que pode lhe auxiliar, e você pode auxiliar, ou quando se faz um trabalho em grupo e que o grupo consegue realmente, é , produzir, você vê que os alunos se desenvolvem, principalmente os alunos que têm um bom grupo, que têm um bom grupo de estudo, então eles conseguem se desenvolver, (PROF4.25) então é mais, eu vejo que os alunos têm mais intimidade com monitores, né? Com outros alunos, não vamos dizer só monitor, né? Outros alunos que já passaram daquela disciplina, e se eles tiverem também alguém a quem consultar, não só os próprios amigos da sala, né os próprios colegas da sala, mas também colegas de outras turmas, né então também é importante, alunos de outras turmas que possam atuar ali como conselheiros, como um suporte, né?
	Papel de outros	(PROF1.15) sites, videoaulas, conteúdo confiável e responsável, então isso aí ajuda muito. (PROF2.28) e se esse aluno não tiver um estímulo, que a Universidade hoje até tem um acompanhamento psicológico pra isso né, se não tiver ele desiste do curso, e isso influencia na vida dele lá fora, né, ele acaba achando que não tem condição, que nunca vai ser um profissional competente... (PROF4.26) é, se tiver por exemplo outra coisa que pode auxiliar, questão da monitoria, né? Ou seja, você tem uma pessoa mais experiente né? Que, que vai lhe dar direções, que pode lá lhe dar indicações, então também isso ajuda muito. (PROF4.27) Eu vou dizer monitor por que é o que a gente tem, digamos oficialmente, né? (PROF5.11) o que é que falta ao aluno? É a gente ter um curso que tenha clareza do que espera dele, e oferecer as ferramentas que permita que ele adquira essas competências.
Parâmetro	**Categorias**	**Unidades de Registro**
Onde	Onde o aluno estuda	(PROF4.28) eles se baseiam muito em testes, né, ou seja, você vai fazer um programa e ele tem um conjunto de testes para saber se o seu algoritmo funcionou ou não funcionou. (PROF4.29) Também tem essa questão das competições de programação. (PROF5.12) então o que falta aos nossos alunos é eles estarem engajados no ambiente acadêmico de alto impacto, que eles vejam a aplicação prática do que eles estão estudando no nível teórico.
	Onde busca informação	(PROF4.30) costumo indicar livros com exemplos para que eles possam praticar. (PROF4.31) Nesse último semestre remoto, além da prática também tem a questão dos vídeos, né? Muitos vídeos na internet, no Youtube ensinando conteúdos (PROF4.32) A pessoa não sabe por onde começar a estudar, como é que começa a estudar, que material é bom ou não é bom, e muitos materiais são muito fracos, muito limitados, ensinam de uma forma errada, então, você tem uma quantidade absurda de materiais, mas você tem poucos que realmente valem a pena, então eles ficam desnorteados, sem saber como estudar, ou o que procurar, principalmente quem vai com essa questão de ser autodidata, né, de ser proativo, e aí nas disciplinas eu tenho tentado, por exemplo, dar indicações de canais no Youtube, que tem bons conteúdos, ou páginas com livros ou tutoriais que realmente o material seja de boa qualidade, né?
Parâmetro	**Categoria**	**Unidades de Registro**
Quando	Entrada no curso	(PROF1.16) “Então, desde 2006, por volta de 2006, quando eu voltei do meu mestrado, eu comecei a trabalhar na Mater Christi, na Faculdade Mater Christi no curso de Sistemas de Informação. E lá eu já ministrei disciplina de programação estruturada, em C na época.” (PROF1.17) “14 anos já!” (PROF.40) “Professor de programação há 19 anos, na data da entrevista” (PROF3.18) “Professor de programação há 21 anos, na data da entrevista” (PROF4.33) “Professor de programação há 16 anos na data da entrevista” (PROF5.13) “Isso, eu acho que eu comecei isso quando você se afastou... eu acho que em 2017, 2016, eu acho que foi por aí que, eu acho que a gente deu uma trocada, eu comecei a pegar as cadeiras de algoritmos, né? Eu acho que eu me considero professor de programação mais ou menos a partir dessa época”
	Período regular – na entrevista	

	UC de programação	(PROF1.18) “Então, programação mesmo, direcionada, foi programação estruturada, programação orientada a objetos, e programação avançada também, entretanto outras disciplinas que têm programação também envolvida como por exemplo teoria dos grafos, programação metaheurística, otimização combinatória, pesquisa operacional, pronto tudo isso aí... é, estrutura de dados também que tem programação e computação gráfica, evidentemente, que eu leciono inclusive esse semestre, pronto, todas elas têm um pouco de programação direcionada, né?” (PROF2.29) “Construção de algoritmos, estrutura de dados, programação estruturada. Em um nível mais avançado, leciona Programação metaheurística (como optativa)” (PROF3.19) “Leciona as disciplinas: construção de algoritmos, estrutura de dados, programação estruturada. Disciplina que exigiam programação: inteligência artificial, Banco de dados, Introdução à ciência da computação.” (PROF4.34) “Eu ministro construção de algoritmos, a disciplina inicial de programação, programação orientada a objetos, ultimamente eu tenho ensinado menos programação avançada que é uma continuação de programação orientada a objetos, onde você vê umas coisas mais avançadas como programação concorrente, programação em rede, com banco de dados... e programação estruturada que é uma continuação de algoritmos onde se vê ponteiros, a questão de técnicas de programação, e já também ministrei uma disciplina de programação distribuída, que é uma disciplina optativa que era mais voltada para isso daí, programação concorrente e programação distribuída.” (PROF5.14) “Certo, aí inclui Construção de Algoritmos, que a disciplina introdutória do curso de programação, apesar de muitos alunos já terem algum contato com algoritmos e programação na disciplina de ICC, de Introdução à Ciência da Computação. E na disciplina de Programação Avançada. Aí são essas duas disciplinas nas quais eu trabalho especialmente com programação. Nesse semestre, esse ano, o semestre corrente atual 2020.2, eu considero uma cadeira na área de programação que é estrutura de dados, que foi a primeira vez que eu ministrei essa disciplina, que foi esse semestre, que inclusive era uma turma de carater especial.”
Parâmetro	**Categorias**	**Unidades de Registro**
Porque	Dificuldades enfrentadas	(PROF1.19) “Então muita gente entra na computação e em cursos relacionados e vai programar e tem aquele choque, não sabe o que é isso, não sabe pra quê que é isso” (PROF1.20) “Seriam os contrários mas eu queria frisar que é uma atitude não só para os alunos, eu acho que todo profissional, acho que todo ser humano, achar que já sabe de tudo. Então, se você viu lá o professor começando a dar aquela, começou o conteúdo, começou a dar aquela basezinha, “ah, vamos lá, programação inteiro, não sei o quê...” e o aluno “ah, isso aqui eu já sei”... não, tenha calma! você não sabe! Vá com paciência, vê todo o desenvolvimento daquilo ali. Então, acho que a pressa é a inimiga né, a famosa inimiga da perfeição.” (PROF1.21) “A pressa, a descrença em si próprio, você achar que não pode, isso aí é uma atitude negativa, isso aí é uma atitude que não vai levar você a desenvolvimento nenhum, né, desistir não pode. Então, pressa, descrença, desistência, tudo isso aí são coisas que o aluno tem que deixar de lado. Afinal de contas, não é por que você tem 40, 50 ou 60 anos, que você tá aprendendo agora que você vai dizer “ah, está muito tarde”. Não! Não tá tarde não. Começa tirando isso aí.” (PROF1.22) “O que pode prejudicar... fatores externos que prejudicam tudo é a dispersão principalmente com as novas tecnologias. Você estudar e estar com o celular do lado (PROF1 mostra o celular, exemplificando a ação), não ajuda de jeito nenhum. Você estudar assistindo um seriado não ajuda. É... algumas pessoas conseguem estudar, escutar música, eu adoro programar escutando música, por que eu acho que entro em sintonia né, mas além disso, tudo o que é fator externo prejudica.” (PROF1.23) “É, atrapalha. Você tentar programar em um ambiente muito barulhento, cheio de, com uma dinâmica muito forte atras e tal, isso também não ajuda, então é isso.” (PROF1.24) “Pronto. É, se falarmos de dificuldades da própria linguagem, vamos dizer assim, do próprio ato de programar, então a gente já coloca o que você citou, né? Ponteiros, é, leitura e escrita de arquivos, armazenamento em disco, essas coisas tudo, tem sempre uma coisa a mais, alocação dinâmica ou ponteiros, né? Tem sempre uma coisa a mais que o aluno vai estranhar e “ah, isso aqui tá diferente aí do printf”, dessas coisas né que ele aprendeu.” (PROF1.25) “É, se você vai fazer alguma coisa mais avançada né, se você vai trabalhar realmente com estrutura de dados otimizada, então isso também é difícil, não é que seja difícil, é uma coisa que não é da na hora que aparece, mas vamos lá.

		Isso aí é durante a programação, durante o ato de programar." (PROF1.26) "Já previamente, a maior dificuldade dos alunos daqui que eu noto é justamente o que a gente já conversou: base. Então o aluno chega e ele não sabe o que ele está fazendo ali, muitos alunos inclusive até entram assim, por serem jovens demais, não sabem se é realmente aquilo que eles querem da vida, mas chega lá e vê um bocado de comandos, de lógica diferente do que ele está acostumado e tudo, "não isso aí não é pra mim"." (PROF1.27) "Então é mais um problema de base mesmo. Base de raciocínio lógico matemático, acho que isso é o maior problema, é o principal problema, na minha opinião. Se o aluno chega com uma deficiência nesse lado aí, então, ele tem que correr atras para ver e continuar." (PROF1.28) O principal pra mim: achar que aprendeu com o professor fazendo no quadro. Se o aluno achar que ele aprendeu a programar com o professor resolvendo um problema no quadro, ou apresentando uma solução no quadro pra ele, isso atrapalha, por que ele vai achar que sabe, e ele não sabe. (PROF1.29) Ele consegue entender o raciocínio do professor, mas ele não conseguiu desenvolver o raciocínio dele. (PROF1.30) Pra mim é a pior atitude do aluno que vai aprender algoritmo é essa, achar que aprendeu com o professor fazendo no quadro. (PROF1.31) Então eu presencio e tenho casos de alunos que começam a disciplina, acham difícil, e nessa de vai dar certo, e não estuda, então como é que vai dar certo? Né, então é assim: "não se preocupe, vai dar certo, vai dar certo, vai dar certo...." Mas se eu não fizer minha parte, não vai dar certo. (PROF2.30) "É uma das disciplinas mais complicadas que eu acho de ensinar, não o conteúdo pela profundidade do conteúdo, mas a questão da didática de passar o conteúdo" (PROF2.31) "É duro, é difícil ensinar programação por que nem todos têm o mesmo nivelamento" (PROF2.32) "tem alunos que têm dificuldades, que cursaram algoritmos, por exemplo, com dificuldade, quando chegam em programação estruturada ainda estão com aquela defasagem de algoritmos" (PROF2.33) "É! É verdade. A grande dificuldade é de abstração, esse é o termo. Ele vem com dificuldade de abstrair e na era em que a gente está da informação, isso não é muito bom por que o aluno tem muita facilidade de encontrar o problema já resolvido, aí ele resolve o problema da atividade, de entregar, mas cria um vácuo na aprendizagem dele pro futuro." (PROF2.34) "Por que eu tenho visto muitos alunos estressados, com as linguagens de programação, com as disciplinas, eles acham pesado o curso" (PROF2.35) "eu tenho muita dificuldade, eu tenho 20 anos de [nome da Instituição omitida] e mais uns 10 anos fora, 10 anos em outras experiências, mas eu tenho muita dificuldade de eu estar ensinando algo e a pessoa estar falando, assim a mente está em outro canto." (PROF2.36) "Se eu estiver explicando aí o aluno tiver conversando, tiver com outra... principalmente no laboratório, aí eu interrompo, peço pra parar, ou então eu digo "olhe termine aí o que vocês estão dizendo, que aí depois eu explico"" (PROF2.37) "Eu acho que o maior problema que um aluno, principalmente nessa, na disciplina de programação é ele estar desmotivado por algum motivo." (PROF2.38) "Ele tem que é... às vezes a desmotivação tem a ver com outro fator, é a própria dificuldade dele, mas se ele tiver minimamente motivado ele pode superar essa dificuldade, mas precisa ter motivação, eu acho que o que mais atrapalha é a falta de motivação." (PROF2.39) "Eu inclusive, já teve casos de alunos que eu sabia que o aluno tinha condição intelectual, mas ele estava desmotivado, aí ele não queria, não queria aprender por algum motivo, por algum momento que ele estava passando, e isso atrapalha." (PROF2.40) "Ah, aí tem muitos, sabe CM? Tem muitos, tem, por exemplo esse período agora, o distanciamento social atrapalha, o ensino remoto atrapalha, por que pronto é uma barreira a mais, a maioria dos alunos não querem abrir a câmera, por timidez, ou por problema técnico mesmo, por falta de condição de um equipamento de qualidade, tem todos esses. Mas pensando no ensino presencial, tem vários fatores, né, é uma fase difícil da vida que você às vezes tá com um relacionamento difícil ou com a família, ou um relacionamento amoroso difícil, né, é uma fase que você tá os hormônios a mil,você está cheio de decisões importantes na vida, e esses fatores externos podem atrapalhar, podem fazer com que "você é um cara que

		intelectualmente é extraordinário, mas que tem dificuldades emocionais, vamos dizer assim, pra concluir o curso”.” (PROF2.41) “Nós tivemos exemplos também assim em computação, de pessoas que a gente sabia que tinham um potencial tremendo, mas que tiveram algumas dificuldades no decorrer do curso por questão emocional, por questão familiar, em alguns poucos casos, graças a Deus em computação não são muitos casos, mas por questão de drogas, então afeta. Esses fatores sociais afetam, né, mas em compensação, a gente tem situações que são, vamos dizer assim, os maus exemplos eu não vou citar não, mas os bons exemplos eu tenho prazer em dizer , tem situação de pessoas que socialmente se você olhasse dava pra desanimar e dizer não sei se termina o curso.” (PROF2.42) “Bem, eu atribuo a maior dificuldade, que não são todos os alunos, você percebe que os alunos que têm facilidade em outras disciplinas, eles também, raramente são os casos que eles não apresentem essa facilidade em programação” (PROF2.43) “É que programação eu costumo dizer que linguagem de programação, lógica, algoritmos, são disciplinas que já conseguem dar assim um certo diagnóstico do intelecto do aluno, né, tem essa condição pela própria estrutura, pela própria necessidade, de algumas construções mentais, que precisam ser feitas, né, de dar esse retorno. ” (PROF2.44) “Mas normalmente o aluno que tem essa dificuldade, é um aluno que tem dificuldade de abstrair, é um aluno que você mostra um exemplo que já vai fazendo uma ponte pra outro, mas quando chega no outro ele tem dificuldade de fazer o link... algumas situações eu acho que são intelectuais mesmo, de dificuldade, outras situações tem a ver com questões sociais, com o momento que o aluno está vivendo, com dificuldades que ele tem que a gente já falou, e faz com que ele não avance.” (PROF2.45) “Aí ligado a isso, não é exatamente a pergunta, mas tem outro fator que nas universidades públicas, principalmente na nossa, é muito latente e atrapalha, que é a dificuldade de infraestrutura, né?” (PROF2.46) “Você tá em laboratórios que não são os melhores, não são os melhores computadores, tem limitações estruturais, e isso atrapalha um pouco” (PROF2.47) então se eles não praticarem eles não vão conseguir fazer, então eles podem assistir mil horas de vídeo com outras pessoas programando, quando eles forem programar vai ter as mesmas dificuldades como se não tivesse assistido, ou simplesmente vão fazer um ctrl +c ctrl+v do que a pessoa que gravou o vídeo estava fazendo, e aí também não vão conseguir aprender. (PROF2.48) mas se for assim como eu disse, se for por adesão dos alunos é bem complicado porque a maioria deles não, se não valer ponto eles não vão se esforçar pra fazer. (PROF2.49) Atitudes dele, primeiro, como eu disse, a questão da prática, né? É... tem que ter a prática. Existem alunos que não praticam, então esse é um problema. (PROF2.50) Outra questão da cópia, ou seja, às vezes você tá limitado, por que as vezes o aluno se habitua a copiar os códigos de outros, e se você estiver sozinho, digamos assim, se você não encontrar um texto, ou um programa onde alguém ensina exatamente o que você quer, ele não consegue, então ele só consegue se tiver sido feito por alguém, e ele conseguir copiar, e isso é muito ruim (PROF2.51) E outra atitude é a questão do.. do... de você passar para um próximo conteúdo sem entender o anterior, né? (PROF2.52) é tentar aprender, digamos assim, pular etapas, ou tentar entender um assunto sem entender o que tá, o básico dele, o antes dele, então não entender, deixar acumular esses assuntos, sem compreender o anterior, é fatal, dificilmente a pessoa vai conseguir evoluir dessa forma. (PROF2.53) E é uma coisa que acontece com muita frequência, ou seja, a pessoa vai, aí não entende um determinado assunto, aí de repente vem um novo assunto, ele não conseguiu entender o anterior e vai tentar entender o novo (PROF2.54) Uma outra coisa que também atrapalha bastante é uma ânsia por querer ver uma coisa aplicada, né, por que você entra em um curso de Ciência da Computação, e então você acha que no outro semestre você vai estar fazendo um Facebook, ne, um sistema assim, e já vai estar ganhando dinheiro, ne, dez mil reais, segundo as pesquisas ai, a media salarial, e então isso causa nos alunos uma ansiedade de ver alguma coisa bem aplicada, e o que acontece é que pra você fazer alguma coisa aplicada, você vai ter que ir subindo degraus, então você não vai: ah, eu sou aluno de algoritmos e no final da disciplina de algoritmos eu quero ter um aplicativo pra android, ou quero ter um sistema web que faça alguma coisa, certo? E isso é uma coisa que eles têm essa ansiedade de ver a aplicação prática daqueles

		conhecimentos, não só em programação, mas no curso também todinho, né? É tipo: pra quê que eu estou aprendendo isso, né, isso não serve pra nada, ou, essa linguagem não está sendo usada. Ah, eu vou ensinar C, [o aluno questiona] onde é que se usa C hoje em dia? Ninguém usa C, todo mundo só usa Java Script ou Python, ou Ruby, ou C#, por que é que você não dá a disciplina em C#? ou, por que que a gente não faz um aplicativo em android? (PROF2.55) Aí beleza, eu boto pra eles fazerem o básico, eles não conseguem nem fazer um Hello World, né, então é... isso atrapalha bastante, por que há um tempo atrás, os alunos eles tinham mais, digamos assim, paciência né, ou seja, ah, eu tô no primeiro semestre então eu vou ver alguma coisa básica, eu tô no segundo semestre, eu já vou começar a aprendendo já... terceiro já vou começar... e existe esse problema dessa ansiedade, né, de ver alguma coisa prática, (PROF2.56) é, então é a questão da ansia, é de, ver as coisas funcionando, a questão da distância, e... (pausa), deixa eu ver aqui... eu acho que era isso mesmo... a questão de (pausa para pensar)... (PROF2.57) então tem muito, é uma geração que, é eu vou dizer geração que eu já estou em outra geração (risos) é uma geração da desmotivação, é uma geração que, acomodada, eu falei ah a pessoa tem que ser proativa, persistente, mas é acomodada, ou seja, se você não ficar ali atiçando, cutucando, faça meu filho, estude mais, então é um problema que tem sido muito recorrente (PROF3.20) "Por que é a disciplina mais fácil do mundo de entender quando você vê um problema o professor fazendo, mas é muito complexo quando você vai desenvolver." (PROF3.21) "mas eu acho muito difícil realizar isso, e uma dificuldade que eu ainda vejo disso é por que os alunos eles chegam com diferentes níveis no conhecimento de programação." (PROF3.22) "Eu tenho aluno que chega, principalmente em construção de algoritmo, que é uma disciplina básica, é a primeira de programação que eles veem, então é uma disciplina que já chegam alguns alunos que vem já tendo noção de programação, outros chegam sabendo programar, e outros chegam sem nunca terem visto nada. Aí torna que fica difícil a aula por que ou tá muito básica pra uns, ou tá muito complexa pra outros, e a gente tem que equilibrar, então fica difícil às vezes de aplicar alguma outra estratégia." (PROF3.23) "e aí eu tenho dificuldade por que realmente eu chego com alunos que sabem programar já estrutura de dados, e com alunos que não conseguem programar, essa é uma dificuldade que a gente tem ainda, que eu vejo que alguns alunos passam em construção de algoritmos sem um conhecimento completamente absorvido pras outras disciplinas..." (PROF3.24) "Não, não é estranho [o aluno ser aprovado sem aprender algoritmos], mas eu digo assim, tem até o que pensar por que como é uma disciplina que eu exijo muitos exercícios, que eles exercitem, eu passo listas de exercícios, e tenho que cobrar as listas de exercícios como nota, e aí muitas vezes a gente consegue saber que o aluno não fez aquela lista mas entregou, né, você tem como perceber. Mas eu não posso não dar nota pra eles se eles entregaram o trabalho todo, mas no fundo não foram eles quem desenvolveram." (PROF3.25) "A gente sabe, mas não tem como provar, a gente sabe que não foi ele, mas não tem como provar, por que ele pode dizer "prove que não fui eu"... e aí ele consegue a nota mínima pra passar na disciplina por copiar ou por pedir pra alguém fazer alguns trabalhos, e aí ele chega nas outras disciplinas sem saber. " (PROF3.26) "Aí deixa eu falar outras coisas negativas, que não é da atitude dele, é um problema social, mas que envolve ele: como eu falei da aptidão que a pessoa ter aptidão ela chega e aprende, existe um problema base, de base escolar que atrapalha muito. Hoje, aí eu não sei se a sua pesquisa se limita a falar de Brasil, ou ela é ampla nisso tudo..." (PROF3.27) "Então tem esse problema de base que desses que chegam sem, se ele tiver aptidão ele vai conseguir desenvolver, mas se ele não tiver aptidão, ele não vai conseguir, ele vai ficar parado, estancado por que ele não vai conseguir desenvolver o raciocínio algorítmico por que ele não tem a base matemática, que eu acredito que seja necessária pra isso. Né, não tô criticando as cotas, eu acho que elas são essenciais, são necessárias, mas é a realidade que a gente tem, então a gente chega numa turma de computação quando a gente vai trabalhar algoritmos, a gente consegue sem eu questionar quem é proveniente de escola pública e quem é de escola privada, a gente consegue meio que diferenciar na primeira avaliação ou o andamento da disciplina a gente consegue ver quem consegue desenvolver o

		raciocínio e quem não consegue, e aí normalmente bate certinho nas teclas, e existe essa divisão, infelizmente isso acontece. “ (PROF3.28) “Né, se você não tiver uma boa estrutura universitária, não tiver um ambiente universitário, não tiver um bom ambiente familiar, tudo é negativo. Tanto eu falei que a existência deles é muito positiva quanto o inverso dele é muito negativo.” (PROF3.29) “mas é uma coisa que depende deles mas, como eu disse também depende do ambiente familiar aí a gente consegue possibilitar para uns mas infelizmente a gente não consegue possibilitar para todos. Por questões sociais, e por questões de demanda, por questões que um determinado aluno pode achar que o professor não gosta dele, são problemas da sociedade de hoje, ne?” (PROF3.30) “Também, pra mim é mais fácil de aprender do que algoritmos, por que você já chega sabendo de algoritmos, né? Então assim, conteúdo eu não vejo uma problemática. O que eu vejo a dificuldade dele, no aprendizado, é exatamente a forma de pensar que eu falei anteriormente. Nós somos acostumados a pensar desde crianças, desde quando a gente começa a nossa educação, a gente se depara com matérias muito conteudistas, que os exemplos que são usados são solucionáveis sem desenvolver um raciocínio algorítmico em nada, nem as questões de matemática na verdade os colégios exploram algoritmicamente esses conceitos, então a gente é muito habituado e treinado por nossas escolas desde o começo a ter um raciocínio rápido pra responder questões, de maneira rápida por que a gente precisa disso, então assim eu preciso conhecer para responder, como eu pensei pra responder eu não sou acostumado.” (PROF3.31) “Então eu acho que a maior dificuldade que eles têm é exatamente isso: por que eles entram na Universidade com 17, 18, 19 anos, e passaram a vida toda deles pensando de uma maneira, e algoritmo quebra isso, quebra a forma de pensar, e aí eles não têm esse costume de pensar algoritmicamente, eles não têm o costume de pensar de maneira diferente, e aí você ensinar esse paradigma de pensar, essa forma de pensar diferente, depois que a pessoa já tem uma consciência de como ele aprendeu a pensar, é mais difícil. (PROF3.32) Então a maior dificuldade que eu vejo assim é por que na base, onde a gente deveria resolver as coisas usando lógica, usando conceitos matemáticos mais fortes, a gente não é acostumado nisso, apenas responde por responder, sem pensar como a gente tá respondendo. Isso é assim, isso é assim, isso é assim... “Mas como você chegou? -Não, eu não sei, só sei que é assim”. Então a gente nunca é acostumado a pensar desse jeito e quando chega em algoritmos a gente tem que pensar. ” (PROF4.35) “mas nem sempre isso consegue ser aplicado, os alunos nem sempre conseguem aplicar essa metodologia, e no final deixam tudo para o final.” (PROF4.36) “e aí você fica naquele dilema se continua por que alguns já conseguiram ou se você espera e aí quem já terminou vai ficar sem ter o que fazer... então fica algo bem complicado.” (PROF4.37) “É, existem na verdade algumas metodologias que eu já vi que é por exemplo utilizando alguma ferramenta de versionamento de código como o Git, então existem algumas, digamos assim, alguns projetos que eles colocam lá um código fonte e abrem os chamados, os tickets, para problemas que existem, funcionalidades que precisam ser implementadas e tal, e eu tentei fazer em um dos semestres mas não deu certo, né, eu fiz só como exemplo mas não deu certo” (PROF4.38) “E, é, mas normalmente as pessoas conseguem chegar até esse ponto, conseguem chegar até por exemplo, condicionais, conseguem chegar sem problemas, mas a partir do momento que chega uma parte mais complexa, por exemplo com laços, com vetores, funções, aí é que pessoal tem mais dificuldade.” (PROF4.39) “às vezes a condição psicológica, emocional, de lidar com problemas, de como resolver aquele problema, se ele consegue ou não resolver o problema, que a pessoa pode ter plenas condições intelectuais pra resolver mas por uma questão emocional que, por exemplo, tenta uma solução e deu errado, e ali aquele emocional, psicológico, abala a pessoa e fica criando bloqueios para que ela não consiga resolver né, então, não é só a questão intelectual, também tem essas questões e isso é uma coisa que ultimamente na nossa área de computação tem conseguido enxergar mais né” (PROF4.40) “Não é uma coisa que seja digamos assim, uma conversa que você vai desenvolvendo ou construindo, então o processo de programação é uma coisa bem rígida, por que você tem que seguir passos, tem aqueles passos e o resultado vai dar errado se você não segui-los, ou trocar um passo por outro, e não é uma coisa que seja um processo natural, é um processo que você tem que mudar sua forma de

		pensar pra poder conseguir, e existe muito da coisa de tentativa e erro, um teste dar errado e você refazer, e fazer de novo e dar errado, e refazer até que você consiga e mesmo quando você consegue fazer, pode não ser a melhor solução, pode ser uma solução ineficiente em relação ao desempenho, então é preciso ter muita perseverança, muita força de vontade, pra não desistir logo assim que tiver os primeiros erros." (PROF4.41) "Porque programação é, ah você vai ver, sei lá, está vendo condicionais. Se você não souber os comandos em uma forma sequencial você não consegue entender condicionais, da mesma forma se você não conseguir entender um condicional que você, é um passo só, que pode ou não pode executar, e você querer aprender laços, onde isso vai acontecer várias vezes, né, então você não vai entender laços, e se você não entender nem condicional nem laços, dificilmente você vai fazer uma função que faça aquilo dali." (PROF4.42) "Então o conteúdo, e isso aí você pode estender para programação orientada a objetos, por exemplo, é diferente de uma disciplina de, digamos assim, redes, se você não entendeu nada em uma camada física, de enlace, mas se você passou para a camada de transporte, de rede, da aplicação, você pode entender só aquilo sem conseguir entender o anterior, mas em programação é muito difícil, ah você não conseguiu entender o que é como é um objeto, ou os atributos, ou o construtor, e aí de repente você vê herança, se você não souber o mais básico, você não vai conseguir entender uma coisa mais avançada, então também é isso daí" (PROF4.43) "Mas se por exemplo, você não tem recursos, então dificilmente você, é, você tem que passar mais tempo para conseguir fazer alguma coisa, você tem que saber mais de alguma coisa, então é muito relacionado a isso aí." (PROF4.44) "então se a pessoa também não tiver tempo, é fica bem difícil," (PROF4.45) "É, fatores externos, na verdade praticamente tudo pode atrapalhar, né? (risos)" (PROF4.46) "Por que digamos, o aluno não tem, ele vai pra casa né, por exemplo não fica no laboratório, ele não tem um recurso bom, então ele vai pra casa, às vezes não tem um ambiente de estudo, né, um ambiente de estudo pra que ele possa estudar, que ele possa praticar, às vezes ele não tem um computador bom, às vezes mora com a família, não tem um quarto, não tem um espaçozinho que ele possa usar lá, então isso aí é, pode atrapalhar bastante." (PROF4.47) "A questão do tempo, voltando também a questão do tempo, é, essa é uma realidade bem específica da [nome da Instituição omitida] né, você tem alunos que moram em outras cidades, né? E esses alunos, eles têm um tempo de deslocamento, né? Às vezes mora em outra cidade, é uma hora e meia de viagem, fora o tempo que ele tem que se deslocar para ir pegar esse transporte, o tempo que ele tem que ficar esperando esse transporte chegar, então, digamos que tem aluno aí que por baixo, perdem umas cinco horas por dia, só nessa questão de transporte, uma hora e pouco, isso só não entre outras cidades, mas também Mossoró, que Mossoró tem a questão do transporte coletivo, né? " (PROF4.48) "Aí, imagine os alunos que passam por isso todos os dias, ou seja, tem pelo menos uma hora e meia, uma hora pra ir e voltar, então você já perdeu de duas a três horas, aí fora o tempo de você ir pegar o transporte, né , e esperar ele chegar, e aí você já perdeu, quatro ou cinco horas aí do seu dia, só nisso. E além disso, o cansaço, né? Ou seja, aí também se aplica a outras situações, né? O cansaço, o aluno que passa essas três horas aí, no transporte, aí chega em casa, vai estudar, então você já tem esse cansaço, ou o aluno que por exemplo tem um emprego, ou um estágio, né?" (PROF4.49) "Isso aí também não é só restrito a computação, pode ser pra qualquer área, você passa aí o cara estuda de manhã, pela manhã toda, vai pro estágio, passa quatro ou seis horas no estágio, aí chega em casa e pronto, agora eu vou começar a estudar, fora os problemas de casa, né?" (PROF4.50) "Então é difícil, e tem a questão familiar, né? Que às vezes você não tem o incentivo pra fazer, pra estudar, né?" (PROF4.51) "A questão de dinheiro, às vezes a pessoa vai trabalhar fora, e tal, e também quem, e aí a gente pode dizer na prática quem tem filhos, né? Também tem que se dedicar, então é difícil essa questão familiar, em relação a, se a família não apoiar, se tiver filhos, se tiver obrigações dentro da própria casa, ou seja, tem que sustentar a casa, tem que.. e tudo isso aí contribui negativamente, são fatores que não têm nada a ver com a Universidade, e contribuem para que ele não consiga tem um bom desempenho no curso. " (PROF4.52) "algumas dificuldades, primeiro a questão da, da, e isso principalmente

		em algoritmos, né? A questão de conhecimentos em matemática, por que em algoritmos a gente costuma falar sobre matemática, né? Fazer alguns algoritmos matemáticos, termos, MVC, aqueles algoritmos bem básicos para a gente aprender, triângulos, se é um triângulo equilátero ou não, e ultimamente a gente tem tido muito problema com relação a isso, os alunos chegam com poucos conhecimentos em disciplinas básicas como matemática, né? E aí quando você vai colocar isso aí é, um exemplo que seja contextualizado né, ah só é fazer isso aqui se você tirar o determinante da matriz, aí o cara: o determinante de que? Como é, como é que faz, não sei... (risos), então quando você pega um aluno que ele não tem esse conhecimento e você manda ele fazer um algoritmo sobre isso, né?" (PROF4.53) "Ai você já tem dois problemas: né, primeiro o cara não sabe e o outro ele não vai conseguir fazer o algoritmo de uma coisa que ele não sabe, ne, então fica muito, muito difícil, então essa parte dos conhecimentos prévios, né, dos alunos." (PROF4.45) "hoje todo mundo está usando Python pra, sei lá, Ciência dos Dados, ou todo mundo está usando Java Script para programação Web, ou todo mundo tá usando Java com não sei o que, pra é.. pra programação distribuída, então é... e isso a gente não tem, isso também é um problema, por que é... quando eles chegam, ou quando eles têm um estágio, ou uma coisa assim, fica aquela sensação "pô a Universidade não tá ensinando o que eu preciso", né..." (PROF4.55) "eh, não sei não, eu acho que muitas vezes uma coisa que não é, que é externa, que é, que não tem nada a ver mas que também é uma dificuldade, é um problema, é que, principalmente para os ingressantes, eles não sabem o que é que o profissional formado no curso faz." (PROF4.56) "Então quando você chega e mostra como é que é feito um programa, e, como é que é tudo ali por trás, né, que você vê aqueles códigos, aquelas instruções, e não tem nada a ver com aquelas coisas bonitinhas, fofinhas, que quando você clica dão um brilhozinho e tal, então também é um problema, né?" (PROF4.57) "Ou seja, a maioria não sabe o que é que o profissional faz, então a disciplina de programação, por que é que eu digo isso pra programação? Por que normalmente programação é a primeira disciplina que dá esse choque de realidade, por que á uma disciplina que é logo no começo do curso, ah enquanto ele tá vendo matemática, essas coisas, ele... é só uma base, não tem nada a ver com computação, mas na hora que chega assim: é computação, pronto, algoritmos, e eles vêm aquilo dali, dá aquele choque: "pô não era exatamente o que eu estava pensando o que eu ia fazer no curso", e isso também influencia, né? Então os alunos entram sem saber o que é que o profissional faz, por que todo mundo que, ah, um médico o que é que ele faz? Ele faz uma consulta, ele faz uma cirurgia... um dentista o que é que ele faz? Ele faz um canal, ele faz... e um cientista da computação, o que é que ele faz? Hackeia Facebook (risos)... ou, arrasta um ícone pra ca, formata um computador, faz perfil no instagram, edita foto no photoshop... então as pessoas não têm a noção, pensam que é tudo menos aquilo que realmente o profissional faz, e algoritmos é o primeiro choque deles nessa, nesse digamos assim, nesse baque aí, nessa distância, nesse abismo entre o que ele pensa que o profissional faz e o que ele realmente faz, né." (PROF4.58) "é o que também é um problema que não é só de programação, não é só das disciplinas de programação, que é um problema geral da Internet, que é a abundância de materiais, né, você tem muito material, muito vídeo, muito livro, muito tutorial, e a pessoa fica perdida, né?" (PROF4.59) "Que outros fatores, principalmente a questão de desmotivação, né... você tem uns alunos que são muito desmotivados, ah, é eu falei da questão da proatividade, mas a grande questão é que a gente tem uma quantidade maior de alunos que são desmotivados" (PROF4.60) "e não são persistentes, ou seja, eles não persistem, assim que eles têm a primeira dificuldade a atitude deles, é mais a principal é vou desistir, vou trancar a disciplina, vou desistir do curso, vou trancar o curso, então você tem muito mais problemas com relação a isso" (PROF4.61) a pessoa tipo assim, você tem uma atividade, aí você vai lá passa a atividade, aí você tem que fazer digamos cinco coisas, pelo menos cinco coisas, né, cinco tarefas, o normal seria fazer cinco tarefas pelo menos, e se faz uma sexta uma sétima, ótimo né. Mas hoje é: você passa cinco tarefas, ele faz três, ai diz que não consegue fazer mais, que está desmotivado, que o curso não incentiva ele, e que essas três tá bom, e se fizer mal feito também ta certo, por que é o que eu consegui fazer, então assim, é uma coisa que não tem nenhuma explicação digamos, a pessoa tem capacidade de fazer mais? Tem! Mas se você disse que é pra fazer cinco coisas, ele faz quatro e meia, e diz quatro e meia tá bom né? Precisa fazer mais do que isso

		não. Então trabalhos de qualquer jeito, não tem o zelo, não tem o cuidado, e assim, é algo que infelizmente não está melhorando. (PROF5.15) “Pela dificuldade deles, é construírem os pseudocódigos, e posteriormente levar isso para a linguagem de programação. Por quê? Por que ele já tinha uma vivência no uso do VisualG mas o VisualG ele já tem uma estrutura do pseudocódigo muito focado na estrutura da linguagem Pascal, e aí as maneiras de expressão disso, envolvendo declarações de variáveis, ciclos de repetição, trazia dificuldades para quando ele ia expressar isso na linguagem principal da disciplina que era Python” (PROF5.16) “Então eles tinham dificuldades de relacionar o conhecimento prévio que eles tinham das estruturas de controle do VisualG, com as estruturas e as representações gráficas dos fluxogramas e consequentemente trazer isso para uma linguagem de programação diferente. ” (PROF5.17) “os alunos ficam com dificuldades de instrumentalizar o terminal, e instrumentalizar o que é da linguagem de programação.” (PROF5.18) “e o que acaba acontecendo é que um dos elementos norteadores da programação é que a programação é o instrumento para se utilizar o computador, e eu considero que utilizar o computador, é interagir através de suas interfaces de comando, e as interfaces de comando visual gráfica, tem muito do seu valor, mas as estruturas de programação, as estruturas primitivas de interação, via console, via chamadas de sistemas, eu passo a perceber que os alunos têm muita dificuldade de compreender a instrumentalização do próprio computador, e isso, dessa maneira no ensino remoto, que é a experiência mais recente, tem gerado muito mais dificuldades, que eu não esperava que fosse haver, dificuldades de ordem de manuseio do computador.” (PROF5.19) “o que é que leva o ensino à distância, remoto, que é essa a nomenclatura que a gente usa aqui, o que é que eu acho que é o grande desafio: que enquanto no presencial nós temos uma homogeneidade do ambiente de trabalho, a heterogeneidade do ambiente remoto a gente não sabe, qual é a capacidade computacional da máquina, muitas vezes não tem máquina, e a gente perde o controle do experimento da aula, por que nós não temos controle sobre o ambiente que esse experimento, que essa vivência está ocorrendo.” (PROF5.20) “Ai agora, além do problema da expressão dos exercícios no início da programação, tem o uso do próprio ambiente de programação, a dificuldade de expressão na ferramenta.” (PROF5.21) “E assim, e essa turma atual que está no ensino remoto, ela é uma turma que não teve nenhuma experiência na presencialidade até agora. Então essa turma não teve nenhuma experiência da interação em sala de aula, na presencialidade, não conhece os laboratórios, já vão entrar no terceiro semestre de um curso, em um curso presencial ministrado na modalidade remota, sem que a gente tenha um preparo para ser um curso a distância.” (PROF5.22) “Uma das maiores dificuldades que eu tenho tido é chegar no final da disciplina e vi que consegui construir com domínio programas que tenha a interface gráfica do usuário, usando as interfaces de janelas, e uma estrutura de processamento com bastante desacoplamento, então eu acho que assim, se eu fosse elencar qual é o objetivo que eu não consegui alcançar em nenhuma turma ainda é fazer um, é orientar na construção dos algoritmos é... a separação dos requisitos funcionais e não funcionais dos sistemas computacionais, e aí os conceitos de modularização de programa, é de estruturas, de reutilização de código é o que eu acho que é o elemento mais difícil de ser aprendido pelos alunos,” (PROF5.23) “Eu acho que hoje, se a gente for analisar com bem clareza, é onde reside a dificuldade de programação: é entender o que se pede.” (PROF5.24) “e aí isso é algo que é difícil pra muitos alunos que é pensar no resultado primeiro antes de pensar nas variáveis de entrada.” (PROF5.25) “Só que isso tem sido difícil pros alunos, por quê? Por que eles estão muito acostumados a ideia do VisualG, de que toda vez que se precisa de uma entrada de dados, se imprime alguma coisa na tela.” (PROF5.26) “E aí é onde eu enfatizo que talvez esse tenha sido metas muito ambiciosas que eu esteja traçando na construção de algoritmos, eu espere que o aluno entenda o funcionamento do computador e ele não entende. E aí ele não consegue instrumentalizar o computador e aí assim, chegando assim a, na minha avaliação, das abordagens que eu tô fazendo na disciplina de Construção de Algoritmos, nas disciplinas iniciais de programação, o que eu percebo é que não há um esforço, digamos assim, anterior para se instrumentalizar o computador.”

		(PROF5.27) “E eles não sabem instrumentalizar o computador, então hoje se eu fosse assim elencar qual é a dificuldade cognitiva do aluno: é a capacidade de estruturar a resolução de problemas, o problema está na formação lógica-matemática do aluno. Segundo: tá na competência técnica, ele não sabe usar o computador, e isso é muito frustrante por que nós temos uma disciplina de Introdução à Ciência da Computação do primeiro semestre que não ensina o que é Computação para os alunos.” (PROF5.28) “A gente vê uma distorção do que a gente espera do aluno e das ferramentas que a gente oferece pro aluno.” (PROF5.29) “Então formar programação é muito complicado. Então a gente pede: eu quero um sistema Web, então eu falo: onde é que a gente ensina sistema Web para essa galera? A gente não ensina em canto nenhum.” (PROF5.30) “Então assim, o desafio da gente o tempo todo é permeado sobre uma pressão externa que a gente recebe do perfil do profissional que a gente forma e que entra em conflito com o perfil do profissional que é resultado dos nossos processos de ensino que não se encontram com esse perfil que a gente espera.” (PROF5.31) “E aí tipo, e aí a gente observa a dificuldade do nosso aluno de conseguir trabalhar com estrutura de dados, tentar compreender a estrutura interna do que é um, de pelo menos, o que é que um sistema SGBD tem que garantir fazendo o uso dessas referencias cruzadas dessas estruturas de dados distintas uma lista, ou seja, uma pilha, que tá referenciada por uma árvore. ” (PROF5.32) “A gente tem essa multiplicidade de expressões computacionais e o aluno não aprende, não apreende o funcionamento do computador, e isso a linguagem de programação que é uma forma de expressão acaba sendo uma barreira muito grande. Por quê? Por que ele não entende o funcionamento do computador.” (PROF5.33) “E aí entra, [breve pausa] e aí é quando ressurge a dificuldade de programação básica, de programação estruturada e estrutura de dados, por que eles não sabem implementar uma árvore.” (PROF5.34) “a gente vive numa confusão formativa, que a gente não sabe o que é que esse aluno tem que ter, qual é o perfil do aluno. Então hoje a minha, como eu tô no começo do curso e no final.. é porque eu tô no sétimo período e no segundo-período, então... e aí esse é o problema que a gente vive. A gente tem uma distorção entre o que a gente oferece e o que a gente pede. ” (PROF5.35) “Eu acho que a gente tem um desafio que vai além disso. Eu acho assim, que é o desafio de fazer essa geração nova aprender.” (PROF5.36) “Eu percebo que a gente tá entrando assim num ciclo que o que a gente considera de valores, não são valores para os nossos alunos, então, ah, vamos falar dos valores técnicos: instrumentalizar o uso do computador, saber usar o computador e entender. Os nossos alunos não querem fazer isso, eles querem clicar nas coisas, então assim, a primeira coisa é valores.” (PROF5.37) “Então nós não temos alunos com perfil de que estão ali investigando, com esse carater investigativo das coisas, de querer entender a fundo o funcionamento das coisas e um comprometimento com o estudo continuado e sistematizado.” (PROF5.38) “E ai qual é o problema disso? Na nossa condução metodológica, o que é que a gente faz pro aluno ler o livro? Manda-o fazer resumo. Então esse aluno ele não aprendeu a estudar, e nem aprendeu a sistematizar o que ele estudou, por que ele está mais preocupado em cumprir um prazo pra entregar um resumo, então a nossa maior dificuldade é que nós não temos parâmetros de avaliação das aprendizagens” (PROF5.39) “Falta uma vivência num ambiente acadêmico de alto impacto, e isso enquanto docentes, nós não conseguimos construir dentro da Universidade, e esse desafio de vivência no ambiente acadêmico de alto impacto fica pior agora nesse momento de pandemia, mas na presencialidade isso já existe,” (PROF5.40) “os nossos desafios enquanto professores de programação é que não conseguimos relacionar a nossa teoria com a nossa prática.” (PROF5.41) “Então a gente tem uma distorção muito forte entre o que é que a gente oferece e o que é que a gente pede. [trecho retirado pelo entrevistado] Então eu coloco hoje a grande deficiência do nosso aluno, é que ele não sabe o que se espera dele. E eu vejo os alunos reclamando muito disso: que eles passam muito tempo fazendo resumo, passam muito tempo fazendo tarefas que não vão agregar, então a gente vê” (PROF5.42) e aí há uma tendência natural dos alunos quererem fazer uma aplicação interativa o tempo todo e desacoplar isso do aluno é muito custoso, (PROF5.43) por que os alunos não vão pro laboratório, os alunos passam na disciplina sem ler livro, as listas de exercício depois viram fax simili da prova e eles passam...

	Consequências das dificuldades	(PROF1.32) “Caso contrário, ele vai cursar uma disciplina todinha sem entender o que está fazendo, no final não vai saber programar, se conseguir ter êxito na disciplina, foi por pura sorte, vamos dizer assim, né, e não por competência dele.” (PROF1.33) “Se por acaso a disciplina não for bem cursada, se ele realmente não aprendeu durante aquele tempo, então ele não vai conseguir ter um vislumbre muito bom das próximas disciplinas, por que como eu te falei logo no início, tem muitas disciplinas que utilizam né, estrutura de dados, a computação gráfica, que utilizam programação, e aí eles vão chegar com essa deficiência lá e não vão conseguir direito cursar. (...) pode causar uma desmotivação geral, que vai se propagar até o final do curso.” (PROF2.58) “Mas assim, mesmo que a gente faça um malabarismo, tente, sempre tem algum que fica para trás, inevitavelmente” (PROF2.59) “e às vezes até desistem por que não tiveram a calma suficiente e o autoconhecimento, o conhecimento de si mesmo para fazer esse tipo de coisa.” (PROF2.60) “Se ele estiver desmotivado ele não vai aprender não. Se tiver desmotivado por que o curso não é aquele para ele, ou por que ele brigou com a namorada ou com o namorado, por que ele não gosta do professor, ou por que ele tem algum preconceito com a metodologia, se tiver uma desmotivação não vai, né?” (PROF2.61) “E pior: atrapalha os outros também, né? Por que ele fica ali, as vezes vai puxar conversa, desanima o professor...” (PROF2.62) “Não significa dizer que aquele aluno que tem dificuldade não vai conseguir aprender não.” (PROF2.63) “Tem, não são raras as situações de alunos de computação que pagam algoritmos 3, 4 vezes, pra poder conseguir, né? Ele às vezes se matricula 4, 5 vezes pra poder conseguir. Fora aqueles que desistem.” (PROF2.64) “Traz aspectos que atrapalham, logicamente, eles passam a se sentir limitados, menos capaz” (PROF3.33) “mas se ele não tiver aptidão, ele não vai conseguir, ele vai ficar parado, estancado por que ele não vai conseguir desenvolver o raciocínio algorítmico por que ele não tem a base matemática, que eu acredito que seja necessária pra isso. ” (PROF3.34) “Bem, como essas disciplinas de programação acontecem muito no início do curso, muitas vezes desmotivam os alunos a desenvolver o restante do curso. Então às vezes alguns alunos chegam em construção de algoritmos, aí são reprovados, e aí tentam pagar de novo, e são reprovados, e aí eles abandonam o curso, desistem, ou deixa prá la a área de programação achando que pode pagar a qualquer hora e vão desenvolver outras disciplinas que não precisam muito de programação, mas o nosso curso basicamente ele vai precisar de programação todos os semestres e ele começa a não se dar bem em nenhuma disciplina, e isso aí é um fator gerador de desmotivação, mas que não começa nas disciplinas de programação, é que começam nas disciplinas de cálculo que vêm antes da programação, por que as disciplinas de cálculo têm o mesmo perfil que eu te falei das dificuldades” (PROF3.35) “e aí eles se deparam com o primeiro período que possui muitas disciplinas de matemática, aí já vem uma desmotivação, e aí quando chega em algoritmos ele já chegam desmotivados em algoritmos, não conseguem desenvolver o raciocínio algorítmico, e isso aí desestimula mais ainda” (PROF3.36) “mas os que não conseguem se dar bem em programação são aqueles alunos que vão se enganchar, que não vão conseguir terminar o curso nos 4 anos, vão passar pra 5, às vezes 6” (PROF3.37) “e essa semana eu tava discutindo com o departamento um aluno que entrou em 2008, ou seja, vai pra 13 anos de curso já que tá fazendo, então a gente tem aluno com esse tempo, e aí com certeza foram aqueles alunos que se desestimularam nas programações, nos algoritmos, e não conseguiram desempenhar o curso bem.” (PROF4.62) “e aí no final das contas ele desiste por que já virou a bola de neve e ele não consegue.” (PROF4.63) “quer dizer, e isso leva ao problema de não conseguir aprender, não conseguir passar, não conseguir fazer os trabalhos...” (PROF4.64) “é a questão da desmotivação, da desistência, da falta de zelo nos trabalhos, e aí você leva aí nas dificuldades” (PROF4.65) “principalmente os alunos, que no último semestre tem o trabalho de conclusão de curso e aí uma das consequências disso, no trabalho de conclusão de curso, que é o ápice né do curso, onde você tem que demonstrar seus conhecimentos, a gente fica com muitos alunos retidos em trabalho de conclusão de curso, ou seja, a pessoa passa o curso todo, vai empurrando com a barriga, vai

		fazendo coisas em grupo, vai de qualquer jeito, e quando chega no trabalho de conclusão de curso, tem gente que desiste, que passa o curso todo, quando chega no trabalho de conclusão de curso desiste, ou passa anos e anos, existem casos de dois, três anos que a pessoa fica só no trabalho de conclusão de curso e não consegue, não sai, por que precisaria ter feito algo anteriormente" (PROF4.66) "e a questão, é voltando agora um pouco mais a questão entre o distanciamento e a ansiedade deles por querer fazer alguma coisa prática já com pouco conhecimento, de pular etapas, o que acontece é que eles não focam no aprendizado do básico, e quando chega no avançado que seria para desenvolver um sistema um web, um aplicativo, ou fazer alguma coisa mais complexa, eles não conseguem por que eles não aprenderam o básico, ou seja, eles querem pular as etapas do conhecimento, e aí não aprendem o básico e quando chegam no avançado não conseguem aprender o avançado, por que não conseguem aprender o básico." (PROF4.67) "Sobre esse distanciamento entre universidade e mercado, é, a questão também é da dificuldade de conseguir emprego às vezes, né, apesar de que a gente já tem conseguido um sucesso, o pessoal que tem se formado tem conseguido os empregos, tudo, os estágios, mas esse distanciamento entre universidade e mercado também causa algumas dificuldades pro pessoal conseguir emprego, por que muitas vezes têm conhecimentos que são exigidos, que não são dados nas disciplinas, mas não é uma obrigação da disciplina ensinar o que está na moda, e os alunos não têm a proatividade de ir atras, então você fica com um cara que nem sabe, nem aprendeu o que tem na disciplina, nem tem o que oferecer pro mercado, e aí tem dificuldade de conseguir um emprego, de conseguir um estágio, mas eu acho que isso para o nosso curso, para a nossa área é relativamente menor, por que mesmo pessoas com pouca experiência, as empresas elas têm contratado, né? Nem que seja para ela fazer treinamentos, para adquirir experiência, mas não vejo isso como um problema grande, mas é algo que precisa ser visto" (PROF5.44) "Então quando você tem isso, o aluno não consegue instrumentalizar isso por que ele não consegue implementar na linguagem de programação os conceitos necessários pra estrutura de dados" (PROF5.45) "o aluno chega no sétimo período, ele não entendeu em canto nenhum as máquinas de estados, ele não entendeu as linguagens formais em canto nenhum, ele não entendeu a estrutura de uma linguagem de programação, as palavras chaves, os tokens, análises léxicas, análise sintática, ele não aprendeu isso enquanto estavam aprendendo a programar e quando chegam lá no sétimo período, ele não consegue expressar nada"
Parâmetro	**Categorias**	**Unidades de Registro**
Como	Metodologia do discente	(PROF1.34) Olha, programação é exemplo! É exemplo e muita dedicação, leitura também, a gente nunca deixa de aprender, sobretudo na programação que está sempre avançando, apesar de várias linguagens já serem muito bem conceituadas, muito bem é... como é que a gente chama? Já estarem bem fixas né, no mundo da programação, mas sempre aparece alguma coisa, sempre aparece uma atualizaçãozinha, uma novidadezinha, e nós temos que estar sempre antenados. É, estudar programação é você pegar os exemplos mais básicos, e começar a implementar. Não tem pra quê ficar decorando comando nem nada não. Implementa com o livro do lado, com o livro do lado ou com a referência do lado também. (PROF3.38) mas como hoje os alunos gostam muito de trabalhar com outras linguagens, tipo Python, tipo é, qualquer uma outra, (PROF3.39) Por que uma dificuldade que eu vejo muito dos alunos é que eles tentam reproduzir o que o professor faz (PROF3.40) Eu acho que algoritmos basicamente é dedicação. Se eles chegarem, se dedicarem àquilo e trabalharem àquilo, né, como eu já falei um pouco antes, é treinar. Se eles treinarem algoritmos, se eles tiverem um tempo pra se dedicar diariamente ou três vezes por semana, a dizer assim: "eu vou sentar aqui uma hora pra desenvolver problemas que eu não sei solucionar" aí entender o problema e solucionar, entender o problema e solucionar, eles conseguem desenvolver isso aí. (PROF4.68) é.. eu sempre assim gosto muito da prática, né? Eu gosto deles praticarem, e eu sempre passo muitos exercícios, eu dou exemplos na aula e sempre passo muitos exercícios, mas não exercícios valendo ponto, e sim exercícios para praticar. Então eu costumo deixar muitos exemplos (PROF4.69) então se você não procurar resolver mais problemas, procurar mais materiais, até por que a aula é num tempo curto, então o professor não consegue passar tudo, né, de todas as formas, todos os exemplos, então você tem que procurar por fora, você tem que aprender uma linguagem, daqui a pouco tem outra nova, então

		você tem que procurar aprender essa linguagem nova baseado no que você já tem, (PROF4.70) ou seja, na verdade é aquela coisa: você tem uma geração que tem mais recursos, que tem mais livros, que tem mais material em vídeo, que você pode a qualquer momento usar a internet pra procurar, tirar qualquer dúvida, e ao invés de você ter eles aproveitando essa quantidade de recursos pra evoluir, na verdade eles estão jogando fora, ou vendo o que não presta, e fazendo coisas cada vez menos importante, menos, mais mal feitas, ne, e ao esforço mínimo, né, vai fazer se realmente for necessário, (PROF5.46) E aí, quando eles vão aprender alguma coisa é quando na terceira unidade o professor aborda algoritmos usa o VisualG que esconde a complexidade do sistema computacional. (PROF5.47) Por que você chega hoje em qualquer turma de graduação da [nome da Instituição omitida] em qualquer período e pergunta: qual foi o livro que você leu de qual disciplina? Não tem livro lido, então falta a nossa dificuldade de fazer leituras.
	Ferramentas que utiliza	(PROF1.35) Eu programo muito em C++, e aí tem o site que é C++ references, né? *Cplusplus.com/references* que é um site básico para se fazer uma boa programação. (PROF1.36) Então. Desde 2008 que eu utilizo o Visual Studio. Eu utilizei o Visual Studio 2008, de 2008 até 2014, mais ou menos. Foi quando eu fiz o *upgrade* para o Visual Studio 2014 (risos). Só ele mesmo. (PROF1.37) Exato, e assim como eu mexo com programação de metaheurísticas, e de certa forma eu tenho que utilizar softwares que são extras, por exemplo, o CPlex, que é um software de otimização, então eu já aprendi como *linkar* tudo no Visual Studio, é por isso que eu utilizo ele ainda, né, já sei todos os caminhos... (PROF1.38) Mas existem outras, como se diz, outros ambientes que podem ser explorados, né? (PROF2.65) Eu gosto muito, eu gosto de usar compilador GCC e o ambiente se tiver uma IDE legal, que dê para o aluno usar sem gastar muito tempo para ele aprender a usar a IDE eu não tenho preferência por nenhuma mas recomendo. Mas por exemplo, para trabalhar com CPLEX, por exemplo, eu uso uma ferramenta da Microsoft chamada Visual Studio por uma razão simples: ela é a que mais facilita a conexão, do ponto de vista de usar as APIs do CPLEX, ela já tem uma vasta documentação que explica como fazer, e ainda tem mais: dependendo da versão ainda dá dor de cabeça, por que às vezes são incompatíveis, a ferramenta da IBM com a da Microsoft. (PROF2.66) Mas aí a gente consegue, né? A primeira vez que eu usei essa IDE com essa ferramenta foi lá na França quando eu estava no pós-doutorado e eu trouxe pra cá e quando eu vou ensinar essa parte de modelagem matemática e resolução de problemas de métodos exatos, eu uso essa IDE que é o Visual Studio, de preferência o 12 que é o que menos dá problema em termos de compatibilidade, com a ferramenta da IBM com o CPLEX, o IBM-LOG-CPLEX, para esta específica. (PROF2.67) Para programação de maneira geral, eu deixo os alunos muito a vontade pra que eles usem as IDEs que preferirem. O que é que eu amarro? Eu amarro o compilador se for C++ para que eu não tenha problema de versionamento de compilação, que não deu certo isso ou aquilo, mas as IDEs eles ficam muito a vontade. (PROF2.68) Eu tenho usado muito o Visual Studio talvez por que eu já usava na minha máquina para esse fim com o CPLEX, mas tem muito aluno que usa o Dev-C++, apesar dele ser bem problemáticozinho, ter umas coisas que são chatas, mas eu deixo os alunos muito a vontade. (PROF2.69) Tem uns que querem pegar uma contramãozinha e eu não impeço não, que é o seguinte: eu tô ensinando, eu apresento os exemplos quando vou mostrar como foi que eu fiz em C++ e o aluno está trabalhando em Java. Eu digo: “bom, não tem problema, desde que você aprenda, e entenda e resolva o problema a ferramenta não é tão importante. (PROF2.70) Mas eu não me prendo muito às ferramentas não. (PROF2.71) Quando eu dou aula de construção de algoritmos, que faz tempo que eu não ensino, a gente usava uma ferramenta, eu não me lembro agora o nome, que é uma pseudolinguagem, pra tirar um pouco das informações da complexidade do tecnicismo da aplicação e fazia com eu o aluno se prendesse mais a resolver o algoritmo. É uma pseudolinguagem por que os comandos eram simplificados, né? (PROF3.41) Pronto, assim, como eu permito que eles usem qualquer linguagem de programação, eu também permito a utilização de qualquer ferramenta, claro. (PROF3.42) Nenhum compilador pra linguagem C, que é a linguagem que eu uso pra apresentar os exemplos pra eles, eu exijo que eles usem um específico. (PROF3.43) Eu uso hoje o Dev, é o que eu uso por que tanto ele roda, existe a versão

		pra Windows, existe a versão pra Linux, e aí ele fica maleável pra quem quiser usar a linguagem C, e eu consigo executar e rodar todas as funções no Dev, mas eu não limito, então tem uns alunos que fazem em ferramentas on-line, tem outros que usam o Eclipse, aí cada um usa o que quer, aí eu não limito isso, aí eu uso o Dev como exemplo e até eu digo: "olhe, ele é simples de baixar, é simples de usar, não tem muito mistério, se vocês quiserem usar é fácil, mas principalmente nesse período agora, nesse último período mais recente que a gente fez ele remoto, existia muita limitação dos alunos a respeito de equipamento, aí muitos deles fizeram os exercícios, os trabalhos, por website , por que tem uns que só usam o celular, u tablet para assistir aula, não tem acesso a computador, aí fica muito pesado baixar compiladores e tal, e aí muitos deles estavam fazendo via compiladores on-line. (PROF4.71) Em relação às IDEs para os alunos eu geralmente deixo em aberto, com relação às ferramentas tem o que eu uso o Juiz online, tem um que é bem conhecido que é em português que é o URI, que é da Universidade de Erexim , no Rio Grande do Sul, e ele é muito grande, tem muitos problemas, aceita muitas linguagens diferentes, e eles sempre agregam problemas de várias competições, né, competições que existem, maratonas de programação, fora outras olimpíadas então sempre deixo em aberto, certo? (PROF4.72) Com relação às IDEs é como eu falei, eu costumo, digamos assim, adotar mas não obrigar os alunos a usarem aquela IDE, inclusive muitas vezes eu faço sem IDE, eu pego o código no editor de texto simples, compilo com uma linha de comando, por que se você usa a IDE, a IDE ele meio que esconde o que está acontecendo né por trás, como é que terá a compilação, quais são os arquivos envolvidos, etc, e algumas vezes é importante o aluno, principalmente o aluno de nível superior de computação entender o que é que está acontecendo por trás das cenas né, e o IDE ele costuma esconder muito do processo. Então às vezes nem IDE eu não uso. (PROF4.73) Mas quando eu uso, eu adoto alguns, mas logo na primeira aula eu já cito alguns que os alunos podem usar, citando nominalmente, no último semestre pra disciplina de algoritmos e programação estruturada eu estou usando Visual Studio Code, por que é um que é muito usado para várias linguagens de programação, então é uma IDE que está em alta, e pra parte de programação orientada a objetos e programação avançada eu uso o NetBeans, né, que é para Java, pode ser para outras linguagens, mas normalmente é para desenvolvimento em Java. E em alguns projetos eu uso Eclipse, por que o Eclipse ele tem muitos plugins, então tem plugin para quase tudo e algumas ferramentas de desenvolvimento eles têm ferramentas específicas para o Eclipse, então eu uso também o Eclipse. (PROF4.74) Aí se for esporadicamente a gente usa muita coisa o Android Studio, algumas IDEs para Linux também eu já usei, para desenvolvimento em C C++, é mais essas Visual Studio e o NetBeans atualmente. Mas já usamos muitas outras. (PROF5.48) No que diz respeito a ferramentas, geralmente eu fazia uma abordagem um pouco mais ortodoxa, né? Por que enquanto existe uma tendência de ao se ensinar programação utilizar IDE, Ambientes Integrados de Desenvolvimento, onde você tem uma noção do espaço de expressão textual do código fonte, você tem um espaço de expressão de variáveis auxiliar logo assim ao lado, ou através de uma ferramenta de depuração integrada também, e a manifestação da saída do código eu foquei muito em trabalhar a ideia de que você tem para programar um editor de texto, você tem o ambiente de compilação, e você tem o ambiente de depuração, desacoplar isso. E eu tenho feito esse tipo de abordagem pra tentar distanciar o conceito de que você precisa ter o IDE para programar. E o grande desafio é que como geralmente eu utilizo ambientes em modo texto, e aplicativos de edição em modo texto, como bloco de notas, ou kate ou G-Edite, ou Nano, e uso intensivamente o terminal, (PROF5.49) Então para muitos deles é... eu avalio que... eu ainda não consegui fechar uma ferramenta de ensino de programação definitiva (PROF5.50) Então, a primeira coisa que você faz no URI, que é uma coisa que eu tenho tentado ensinar ao aluno. O URI hoje ele tem a estrutura de você montar em blocos, ou seja, aquela estrutura para você montar bloco já tem dentro do URI hoje em dia para alguns problemas. (PROF5.51) Pois é, ao invés de eu usar o Scratch no início, a gente usa a estrutura dos blocos de construção dentro do URI, para exemplificar esse processamento.
	Metodologia do docente	(PROF1.39) É, bom, existem várias estratégias, né? (PROF1.40) gente tem que conhecer a turma para poder traçar a própria, mas o ensino de programação é algo que você tem que fazer com que o aluno entenda através de analogias, e através de justificativa do por que dele estar fazendo aquilo. (PROF1.41) então acho que a primeira coisa a fazer é justamente explicar o por quê e

		como e tentar, tentar mostrar o básico do básico, vamos dizer assim o bê-a-bá, o *hello world* , como é que se troca uma lâmpada, naqueles exemplos de algoritmos, de construção de algoritmos, então começando pelo básico aí depois introduzindo as demais características de programação, né? As suas estruturas de repetição, e assim por diante. (PROF1.42) eu estou muito satisfeito, assim, com a forma que eu lido com a disciplina (PROF2.72) Então durante muito tempo eu achei que mostrar o código para uma aluno de um exemplo fosse uma boa estratégia, mas aí com o tempo a gente vai ficando né os cabelos brancos não é só pro cabra ficar com a aparência de idoso, você vai aprendendo também alguma coisa (risos). (PROF2.73) Então eu percebi que essa estratégia de dar um exemplo, e mostrar e querer que o aluno repita não faz com que de fato ele aprenda, ele começa a fazer as coisas, até consegue fazer funcionar um determinado código, mas que com se você muda alguma coisa, pede algo diferente, o aluno não consegue pensar para abstrair e programar. (PROF2.74) Então, é, de um certo tempo para cá, o que é que eu tenho feito: eu tenho mostrado a lógica do que eu quero, o que é que eu quero, ou seja, o que é o problema, e como resolvê-lo. (PROF2.75) Aí eu tento mostrar várias formas diferentes de resolver um mesmo problema, aí vou mostrando níveis de programação pra resolver aquele problema, o mais trivial, mesmo que seja o mais forçado para o computador, em termos de processamento, aí a gente vai resolvendo. Aí eu começo com problemas bem simples, os triviais, (PROF2.76) aí eu mostro: “olha aqui um problema bem simples, mas um algoritmo bem casca grossa para resolver, aí é esse aqui, que você gasta pouco juízo e muitos dedos” (PROF2.77) Aí depois eu digo “agora vamos melhorar”. Aí eu vou fazendo, e aí eu consigo sair de um problema bem simples e aí vou para algoritmos que já tem essa característica, como por exemplo, algoritmo de busca, de ordenação (PROF2.78) Aí eu pego: “olha, o algoritmo de ordenação ou de busca, o mais trivial é esse... vai gastar mais tempo... mas não vai gastar muito raciocínio, por que é só você percorrer a lista e ver se encontrou”. Aí vou mostrando, e aí eu dou tempo para que eles tentem fazer. (PROF2.79) mas qualquer disciplina que ele tenha conhecimento de codificação, aí sim eu já tenho condição de partir do problema e pedir que ele resolva. Aí eu digo “olha você se não tiver a condição de escrever o pseudocódigo, ou o código, comece escrevendo o que você vai fazer ou o que o computador faria para resolver”, né? (PROF2.80) Obviamente que um processamento simples, de um programa simples, mas que você verificava a corretude do algoritmo. E aí eu tento fazer isso. Uma coisa também que me ajudou bastante em termos de estratégia e de didática para eu ensinar, é o fato de eu estar com uma disciplina no mestrado de projetos de análise de algoritmos, né, que também envolve programação. Apesar de que nela eu não ensino uma linha de código, mas envolve a análise do algoritmo, se ele é bom, se ele é ruim, então é mais difícil fazer isso do que programar por que você tem que verificar dentro do pseudocódigo, do algoritmo se o algoritmo é bom ou não, ir fazendo o cálculo de complexidade, então eu tento quando vou ensinar por exemplo programação estruturada, já dar algo disso antes de entrar na codificação. Tenho até alguns colegas que programam, que trabalham com linguagem de programação ensinando que acham que em determinada linguagem, determinado nível, por exemplo, construção de algoritmos, não deve ensinar nada que o aluno programe, né, que use linguagem. (PROF2.81) Eu acho que depende muito da turma, depende do nível da turma. (PROF2.82) Mas eu uso essas estratégias, fazer ele pensar, resolver o problema sem programar, e depois codificar. Aí na codificação o que é que eu faço: eu deixo eles a vontade, e se tiver dificuldade aí eu trago a minha codificação, aí eu digo “olha esse aqui é o meu estilo, eu resolvi assim, mas você pode fazer de maneira diferente”. E eu também digo “olhe, fiz em C++, mas vocês podem fazer em C ou sem usar classe, em Java, aí eu vou mostrando. E aí tem dado certo. (PROF2.83) mas de maneira geral se você coloca o aluno para pensar, e escrever a solução do problema de alguma forma, com fluxograma, com pseudocódigo, com pseudoestruturado, como pseudolinguagem, é melhor que ele faça isso que ele juntar na hora de programar a necessidade de conhecer a sintaxe de uma linguagem com a necessidade de saber resolver o problema. Como eu tenho feito nas minhas últimas disciplinas, eu tenho trabalhado assim.

		(PROF2.84) É, o que eu faço talvez seja muito influenciado pela forma como eu aprendi. (PROF2.85) Mas o que eu faço com os meus alunos é o seguinte: eu sugiro problemas, e digo aquela mesma história "aprenda a resolver o problema, trace os passos que você precisa para resolver o problema, quando você compreender o problema, que sabe resolver, aí você vai para a codificação, que você pode ter feito também em pseudocódigo, depois você vai para a codificação". (PROF2.86) Aí eu sugiro para os alunos né que por exemplo, numa disciplina, nessa disciplina de metaheurística, que a gente não vai passar por nada de programação, de rotina de laço de repetição, de estrutura de dados, não vai passar por nada disso, eu vou direto pra o problema e digo: "olha, uma metaheurística para resolver o problema vai ter um aleatorizador, vai ter um cara que sorteia, vai ter um aqui que corta e emenda, vai ter um que gera uma população, e são rotinas que você precisa, eu passo a lista das rotinas. Aí depois ele diz "professor, cada um é um problema diferente com um algoritmo que eu vou criar?" aí eu digo: "exatamente". Só que aí antes deles começarem a programar essas coisas, nas primeiras aulas eu digo: "olhe, façam uma implementação de uma lista duplamente encadeada, faça uma implementação desse e desse algoritmo de ordenação, desse e desse algoritmo de busca, por que vocês vão precisar deles na disciplina. Se você fizer agora como exercício de revisão, quando chegar lá vocês vão só copiar e colar o código ou chamar a biblioteca e usar, por que inevitavelmente na metaheurística vocês vão precisar de lista duplamente encadeada, pilha, vai precisar de algoritmo de ordenação", então eu passo para essa disciplina avançada que o pessoal já pagou as outras disciplinas, eu passo uma série de algoritmos clássicos que eu já sei que eles vão precisar, por que aí quando chegar lá na frente eles só fazer, e aí eu já tive aluno que diz "professor eu não lembro mais como é que cria uma biblioteca" aí eu digo "pronto então vamos criar uma biblioteca e você vai usar essa biblioteca". E aí funciona assim. E é muito bom, (PROF2.87) Mas de maneira geral, CM, não tem muita receita pronta não, por que às vezes você pega um grupo diferente e aí você tem que criar na hora, às vezes como professor você tem um insight em uma aula de fazer uma coisa que ajude, né, e que nunca fez antes, mas é assim mesmo. (PROF2.88) É... tem, tem uma parte da disciplina que quando eu chego já, eu já passei os problemas para eles resolverem, e tal, e que eu vou compartilhar com eles o que eu fiz, né, os problemas que eles estão resolvendo eu já resolvi, eu fiz assim, vou só mostrar aqui como funciona e tal... aí o que é que eu uso, que é uma ferramenta legal: eu uso do meu notebook sentado com eles, eu vou projetando no projetor multimídia, de forma interativa, aí eu vou e ou eu construo rapidinho um módulo simples, ou então eu já mostro o módulo pronto. Aí eu vou e coloco em destaque na codificação o que eu julgo que foi mais complexo naquele módulo de implementação, ne? (PROF2.89) Ou então que eu fiz diferente. Eu digo "olha gente, nesse módulo aqui que está fazendo isso, eu implementei desse jeito, mas existem muitas formas diferentes, eu só tô chamando atenção aqui por que eu fiz assim, mas vocês podem fazer diferente". Então, se tem algo que eu sei que é o maior foco de erro naquela codificação, aí eu digo destaco, coloco como se fosse numa apresentação no slide, mas na verdade é o próprio ambiente codificado, aí eu mostro como foi que eu fiz, aí eu rodo um exemplo, às vezes eu rodo os módulos todos separados, aí durante o exemplo eu digo: "eu vou dar uma entrada que é problemática, o que é que pode acontecer nessa entrada?" aí eu coloco "pode acontecer isso" aí coloco e mostra a saída "olhe, não aconteceu, por que eu tratei esse erro aqui dentro, como foi que eu tratei?" aí então eu abro e mostro o código. (PROF2.90) Aí eu vou tratando do que eu chamo dos gargalos naquela algoritmo, naquela problema, que eu já sei que ali é a maior fonte de engancho que eles podem ter. (PROF3.44) Primeiro eu conceituo né as instruções, os comandos, para que é que eles servem, sem dar muito enfoque, e aí eu tô falando de construção de algoritmos, porque em estrutura de dados eu pressuponho que eles já sabem, mas em construção de algoritmos eu pressuponho, é, eu mostro as instruções, mostro os comandos, desenvolvo a lógica por trás deles, e aí trabalho muito com exemplos, exemplos e mais exemplos, eu não foco muito em sintaxe de linguagem por que eu tento fazer de uma maneira que os alunos aprendam a programar, não aprendam uma linguagem, mas eu uso sim a linguagem C como fundo, né, então os exemplos eu desenvolvo em linguagem C, o meu enfoque é para que eles consigam fazer um *if*

		ou um laço em qualquer linguagem de programação, então eu explico conceitualmente o comando, a instrução, e depois aplico esse comando usando normalmente a linguagem C, por mais que eu deixe aberto e essa característica, pelo menos ultimamente, no começo eu trabalhava só com Linguagem C. (PROF3.45) eu foco com linguagem C na aula, nos exemplos (PROF3.46) Sim, eu costumo comparar programação com eles com matemática. (PROF3.47) Né, então o que eu estimulo eles é exercitarem, é realmente fazerem exemplos, fazerem algoritmos, não copiarem os meus algoritmos, tentarem fazer os deles, podem usar os que eu desenvolvo como uma base, como uma consulta, mas o que eu recomendo é que eles façam, é nesse sentido que eu oriento o estudo deles. (PROF3.48) E eu não tento aplicar muitas outras estratégias por que eu vejo que a que eu estou usando hoje tem efeito, e aí na hora que eu percebo que ela não teve efeito, aí sim eu tento modificar durante o período corrente, ou então eu tento ser bem flexível com respeito a esse ensino, no decorrer do período, por que depende muito da turma, do conhecimento da turma, do tamanho da turma, do envolvimento deles, aí eu vou fazendo minhas adaptações, de acordo com o que cada uma das turmas exigem. (PROF3.49) Aí eu falei muito de construção de algoritmos, né, já em estrutura de dados, que eu cobro implementações, mas não ensino, eu dou duas ou três aulas de programação mesmo, por que muitas vezes eles chegam em estrutura de dados sem saber ainda de ponteiros, estruturas, tal, e eu preciso muito, então eu dou duas ou três aulas sobre esse assunto, pra antecipar outras disciplinas de programação que eles têm no mesmo período, que eles veem isso, mas é mais na frente, né então eu antecipo um pouco pra eles usarem. Então eu não uso o conteúdo de programação, eu aplico o conteúdo de programação, (PROF3.50) Tanto que quando eu começo a disciplina de construção de algoritmos eu tento começar com duas ou três aulas bem diferentes das coisas que se faz, tentando a forçar eles a entender por que eles têm que fazer as coisas e não simplesmente fazer as coisas. Então quando eu começo a disciplina, eu falo assim: “gente, como é que vocês fazem pra chegar aqui, pra chegar na Universidade, como é que vocês fazem pra vir pra Universidade? E assim, eu não quero que você diga que acorda entra num carro e chega, eu quero que você pense no que faz inteiramente” e aí eu tento mostrar que a gente tem que ter um pensamento muito detalhado o nível de nossas ações no dia a dia, pra que a gente consiga pensar algoritmicamente as coisas, né, e eu acho que se a pessoa conseguir pensar em forma de algoritmo, ela consegue desenvolver qualquer algoritmo. Então é uma coisa que eu tento trabalhar com eles isso. Dizer assim: “Ah, rapaz pra eu escovar minha boca o que é que eu tenho que fazer? Ah, eu tenho que pegar minha escova, eu tenho que pegar o tubo de pasta, eu tenho que abrir o tubo de pasta...” por que se a gente fizer a pergunta direta, ele diz “não, eu vou lá na pia e escovo” .... Mas o que é que você fez pra escovar? Aí na hora que a gente começa a pensar que as nossas ações do dia a dia são algoritmos, e que a gente tem que fazer várias pequenas ações pra concluir uma ação um pouco maior, a gente começa a começar a pensar algoritmicamente, aí é isso que eu tento mostrar eles nas primeiras aulas exatamente pra forçar um pouco esse pensamento algorítmico. (PROF3.51) Né eu tento mostrar, eu pego alguns problemas básicos e digo assim “ó ta aqui, eu posso resolver desse jeito, desse jeito e desse jeito, a forma que eu pensar resolve, mas a forma que EU penso é essa, não é a que você pensa” pra tentar forçar, mostrar a eles e forçar que cada um tem a sua solução e não acreditar na do professor. (PROF3.52) Então por isso que eu falei que eu dedico aquele inicio de disciplina pra dizer assim: é assim, vamos começar a pensar diferente, vamos pensar algoritmicamente, por que ninguém tá acostumado a pensar algoritmicamente. (PROF4.75) Basicamente todas as disciplinas que eu ministro eu peço trabalhos na área de programação. Por ser um professor da área de programação, eu entendo, no ensino superior, que a programação é uma ferramenta, não é o fim do curso, não é o objetivo do curso formar programadores, mas usar a programação como uma ferramenta para você desenvolver trabalhos da área, então por exemplo quando eu dou a disciplina de compiladores, então em compiladores eles têm que desenvolver um compilador, já ultimamente também já dei umas disciplinas no mestrado, na pós graduação, e uma das disciplinas era a de ciências de dados, introdução a ciências de dados, então no final da disciplina eles tinham que fazer uma análise de dados mas isso aí envolvia programação, então se pegar meu histórico assim, a não ser uma disciplina mais teórica, por exemplo, análise de sistemas, aí eles não precisavam fazer

		a implementação, mas fora essa a grande maioria dos trabalhos, das disciplinas, tem como um dos objetivos, um dos trabalhos dos projetos finais, envolvendo programação. (PROF4.76) É, para ensinar programação existem estratégias diferentes de acordo com o passar da disciplina. (PROF4.77) Normalmente nas aulas iniciais, nas aulas mais introdutórias, eu costumo fazer pequenos exemplos desconectados um do outro, então são exemplos sobre várias coisas diferentes. Então cada assunto, vários exemplos. Mais pro final quando já tenho um, digamos assim, eles já têm um conhecimento maior, e também já desenvolveu vários exemplos, a gente vai, eu vou tentando refatorar os anteriores né, explicando esses novos conceitos e tentando mostrar as vantagens de uma técnica nova, de um conteúdo novo, podem aplicar para resolver problemas antigos, né? E tem também a questão dos projetos. Como eu te disse em toda disciplina tem o projeto final, esse projeto final ele não vai, apesar de eu tentar e às vezes não conseguir, o objetivo sempre é ir fazendo, construindo ele aos poucos, por etapas, então em cada avaliação a gente tem uma evolução, então há um certo ciclo de desenvolvimento evolutivo (PROF4.78) Também existe em algumas disciplinas de algoritmos e programação estruturada as competições de programação, então existem alguns portais de problemas né, de programação, e aí eu passo esses problemas e esses portais (PROF4.79) E também, as minhas disciplinas elas são muito práticas então eu gosto sempre de colocar exercícios, fazer normalmente mesclando aulas teóricas com aulas práticas, então a gente vê um conteúdo e na outra aula a gente faz exemplos e exercícios, ne, e ultimamente também uma construção digamos assim mais colaborativa, então ao invés de eu mostrar o exemplo e resolver, eu chamo cada um dos alunos que, no caso antes da pandemia era mais fácil né? Então eu chamo cada aluno que estava no laboratório para fazer uma parte da solução. Então eles vão se revezando, e eu divido no caso o problema em partes que precisam ser resolvidas e ai cada um vai fazendo uma parte e deixando pro outro que vai fazendo a outra parte, até chegar na resolução. E essa daí tem até sido bem legal por que muitas vezes quando você deixa o exercício aberto, eles não têm a iniciativa de fazer ou então acabam muito rápido e os outros ficam sem fazer, né, (PROF4.80) Então essa estratégia do aluno fazer o programa de maneira participativa é boa por que eu também incentivo os alunos a contribuírem com os colegas né, perguntando “ah, o que é que tem que fazer” que aí quando ele não sabe eles ajudam dizendo o que é que tem que fazer. (PROF4.81) A minha metodologia é mais essa fazer competições, fazer a competição com os robôs, usando os robôs, pegava um robô e programava, na verdade não era nem um robô físico era um robô virtual era uns tanquezinhos, e cada um programava o comportamento do robô, e eles batalhavam. Também foi bem bacana pra época, tivemos um resultado bem legal. (PROF4.82) mas por outro lado também existe isso, né, essa distância entre o que a gente ensina e o que tá sendo usado hoje no mercado, ne... que a Universidade ela não foi feita, digamos assim, para atender só o mercado, ou pra ensinar o aluno... não é um curso técnico que você vai aprender o que está sendo usado. (PROF4.83) A gente dá as bases, dá o conhecimento teórico, aplica isso em alguma programação, aplica isso em alguma linguagem, mas não necessariamente a gente vai botar pro cara estudar que está lá, (PROF5.52) Tem. É, comumente eu utilizo Python e faço algumas pinceladas na área de C. Mas a disciplina, se for para determinar qual a linguagem que a gente usa é Python mesmo. (PROF5.53) Programação Avançada, aí a gente já tem uma abordagem um pouco diferente. Por que? Em programação Avançada os tópicos estão relacionados à integração com Banco de Dados, desenvolvimento de aplicações, um pouco na parte de redes, em programação em rede, e os conceitos relacionados a padrões de projetos. Então, dentro dessa disciplina a gente aborda Java, como uma linguagem de referência, tem C++ já que o livro dos quatro apresenta exemplos de padrão de projetos em C++, a gente usa Python pra apresentar o conceito de features e funcionalidades em linguagem de programação, e a gente observa que dependendo das funcionalidades da linguagem de programação, a maneira de implementar os padrões de projetos mudam. E a gente faz uma ênfase muito forte nisso, né, que os padrões de projetos eles são relacionados às dificuldades internas das linguagens de programação ao invés de serem técnicas de modelos pré-definidos. E um pouco de Java Script, por que? Por que um catálogo de padrões de projetos muito famoso e

		também relacionado à área de refatoração de código, é... apresenta exemplos em JavaScript também, que também é relevante devido a hoje, é... uma presença do JavaScript tanto no front end quanto no back end, então não tem um foco de uma linguagem específica em Programação Avançada. (PROF5.54) Certo [pequena pausa]... as metodologias que eu apresento, geralmente, são na resolução de problemas. A gente define um problema a ser abordado, e a ideia é construir soluções para resolver esse problema. Nesse semestre especificamente, eu dei um foco muito maior na construção dos fluxogramas e nos pseudocódigos, e eu vi que isso, nesse contexto atual do ensino remoto, acabou trazendo um pouco mais de dificuldades pro aluno (PROF5.55) Então nesse semestre, houve uma confusão muito grande nisso, e foi uma experiência que eu fiz dá um enfoque nessa parte de pseudocódigo e fluxograma. (PROF5.56) Anteriormente, as disciplinas de introdução à programação, ou seja, construção de algoritmos, era muito mais focada na instrumentalização da linguagem Python, e eu notava que havia um desempenho maior. Então eu comecei, a minha conclusão é que nesse momento, pelo menos no ensino remoto, dar uma grande ênfase na representação do algoritmo trazia dificuldades para a expressão disso em uma linguagem de programação. (PROF5.57) Então um exercício que eu tenho feito comumente é pegar problemas de livros de matemática do ensino fundamental. (PROF5.58) Então o que é que a gente aprende no ensino fundamental? Enquanto a gente está lendo o enunciado de um problema a gente já começa a selecionar quais são as variáveis de entrada, a gente começa a selecionar o que é que é a saída esperada do problema, entender o que é que o problema está pedindo, e com isso a gente começa a estrutura os relacionamentos entra as variáveis de entrada do problema. (PROF5.59) Então assim, no URI, a gente tem aquela estrutura de resolução em blocos, então o que é que eu começo a fazer: a gente começa com output. Então a primeira coisa que a gente tem que fazer é estruturar o output. E o legal da gente usar blocos, é que quando a gente usa blocos, e eu só usei os blocos usando programação em Python, ele vai fazendo a geração de código ao lado, então qual é a estratégia que eu estou usando: a gente vai usando a estrutura de geração de código usando blocos que gera código Python e orienta usar um interpretador no computador, para ir preparando pra interpretar se a saída tá dando conforme a expectativa da formatação que o problema do URI pede. (PROF5.60) Por que muitas vezes há problema ou o aluno passa muito tempo no *presentation error*, no erro de apresentação, ou seja, o alunos às vezes calculou até correto, mas a forma não coincide com o que é pedido. E aí o URI dá aquilo equivocado. Então a gente começa a fazer uma estratégia de traz pra frente. (PROF5.61) Que aí já começa, a gente começa já a segmentar, tentar fazer uma abordagem de segmentar, que a parte de resolução do problema, o seu núcleo, basicamente ele tem que ser expresso na estrutura de definição de uma função, então você define uma função, cujos parâmetros são as variáveis de entrada e que retorna uma saída e essa saída vai ser formatada para ser apresentada pra alguém.
Parâmetro	**Categoria**	**Unidades de Registro**
Quanto	Importância do aprendizado	(PROF2.91) Isso é cumulativo, você vai conseguindo ter aí um repositório de conhecimento, de informações e de segurança em programar à medida que você faz isso. (PROF2.92) Mas assim, eu acho que o fator que tem é que ele tenha a motivação certa, importante para o bem é a motivação certa: é pensar em aprender para aprender, não tá com a mediocridade de dizer que sabe o suficiente para fazer a prova e passar e pronto, que não esteja lá só pelo canudo. Se tiver a sede de aprender mesmo que não pense numa grande empresa, o emprego é consequência. (PROF2.93) O aluno que tem a condição de se sobressair, de estudar por fora, ele supera isso. (PROF5.62) Que eu acho que isso traz uma carência muito grande, por que? Por que um programador, ele sabe instrumentalizar na linguagem de programação. O desenvolvedor além de instrumentalizar a linguagem de programação, ele interage com o contexto do programa executado e desenvolvido. (PROF5.63) Então o desenvolvedor ele sabe versionar o seu código, o programador não precisa fazer isso, ele precisa programar, criar código, o desenvolvedor ele entende o contexto da execução do programa, o programador ele acaba esquecendo isso por que o foco do programador é implementar os mecanismos que as interfaces solicitam, os protótipos de função, e assim por diante... então há uma diferença entre

		o programador e o desenvolvedor (PROF5.64) o que o aluno não tem o que ele precisa para ele aprender programação? Que a gente ensine o que é programação! E em canto nenhum a gente ensina, e a gente tem pontos de programação dispersos.
	Dedicação e iniciativa	(PROF1.43) Bom, se em sala de aula é cobrado para você fazer o exercício A, B, C, então você vai e além disso aí, você crie seus próprios, vá atrás de outros exercícios, D, E F, G,Z... então seja proativo nesse sentido. "Ah, vou saber curiosidades sobre a linguagem de programação que eu utilizo", " Ah, vou procurar novos caminhos para fazer aquelas mesmas linhas de código, sei lá, *tô* com dez linhas de código, então como é que eu posso reduzir aí pra cinco..." então tem que ter esse tipo de curiosidade. (PROF2.94) E isso é muito bom por que às vezes tem alunos que sugerem, que fazem soluções que eu nem esperava, que eu nem tinha pensado, aí eles vão "ah, eu fiz assim professor, diferente", aí quando eles dizem diferente eu vou logo olhar, aí às vezes é uma solução que o cara pensou, e que criou mesmo e que às vezes inclusive menos complexo que a solução que eu propus, e isso é muito bom. (PROF2.95) Se eu não souber, não tiver uma rotina, ah, eu sei que Java tem uma classe, uma rotina, ou C++ tem uma rotina que resolve isso. Às vezes os alunos riem com isso né, diz que não vão fazer isso, eu digo: "eu prefiro fazer a minha", se eu não lembro como é que usa, ou onde é que tá, eu vou lá e faço a minha (risos), aí eles dizem não...
	Visão do futuro	(PROF1.44) Mas como tudo está sempre avançando né, evidentemente eu não vou ficar estagnado nisso sempre. Eu quero algo mais, realmente, quero algo colaborativo, quero algo que os alunos consigam fazer todos em grupo, mas sem ser aquele grupo em que um fica se aproveitando do outro, e que eles participam e aprendam, e mostrem todo o potencial de uma equipe inteira, era isso que eu gostaria de utilizar, ou seja, fazer mais uma aula voltada pra projeto colaborativo. (PROF1.45) Onde cada aluno faz um mega programa, sei lá, um programa, vamos dizer um jogo, então tem aquele cara que é o programador do som, o programador das interações, o programador 3D, e assim por diante. (PROF2.96) Todos os alunos de computação que eu conheço, não só da [nome da Instituição omitida] mas que foram meus colegas por exemplo quando eu tava no mestrado, que eram alunos de graduação na [nome da Instituição omitida] que tinham essa perspectiva de sonhar alto, todos eles se deram muito bem, que eram aqueles que não pensavam em programar para fazer a prova. (PROF2.97) É, eu acho que os fatores externos que podem influenciá-lo para o bem, vamos dizer assim né (risos), é ele perceber que aquele conhecimento de programação vai influenciar no futuro dele como profissional, né, que ele pode ser um bom programador de jogos, que ele pode ser um bom programador científico, ou que se ele se destacar como programador, ele pode galgar uma posição profissional numa empresa como Google, Microsoft, né, eu acho que é fundamental que o aluno de computação, nesse aspecto, ele sonhe alto. (PROF2.98) Ele diga "olha, eu quero aprender a programar não é só pra cursar a disciplina não, eu quero aprender por que eu tenho a expectativa de trabalhar com isso." (PROF3.53) É, assim, eu não sei, se a gente podia usar muito em programação eu até gostaria, mas aí serviria para todas as disciplinas, tipo usar a sala de aula invertida, e dizer assim: "ó próxima aula a gente vai trabalhar isso, e vocês estudem e venham e tragam pra você já chegar na aula sabendo o conteúdo pra gente discutir e aplicar" (PROF4.84) Mas essa é realmente uma metodologia, essa metodologia colaborativa usando versionamento de código era o que eu gostaria de utilizar mas não consegui utilizar. (PROF4.85) Mas o que eu não usei ainda seria esse tipo de desenvolvimento compartilhado e cada um implementava uma funcionalidade de forma colaborativa. (PROF4.86) Embora os alunos não percebam, mas os professores da área têm tentado se adaptar, têm tentado mudar, tem tentado usar novas tecnologias para ensinar, eles têm buscado conhecer o que tem no mercado, o que está sendo usado no mercado para trazer alguma coisa pra dentro da sala de aula, ne? (PROF5.65) Por que quando a gente vai implementar um algoritmo, o que é que a gente está fazendo? A gente está fazendo uma estratégia para que o profissional de programação entregue valor pra quem o contrata. E qual o valor que o profissional de programação agrega para quem o contrata? É se ele gerar resultados, então a gente começa pelos resultados. O contratante está pouco se lixando pro seu conjunto de entrada, tá pouco se lixando pro que você tá fazendo no meio do caminho, ele quer

		que seu resultado apareça, então a gente foca no output, (PROF5.66) Se pensa muito no processo, se gasta muito tempo no processo, e você faz às vezes um processamento que não interessa a ninguém, então, por que não é reutilizável, não está formatado, vai precisar de um outro esforço pra transformar aquele resultado numa informação apreciável para quem solicitou a resolução do problema. Então a gente inverte o fluxo de trabalho: a gente dá um foco no output, depois dá um foco no processamento, e depois prepara a entrada das variáveis. (PROF5.67) o que é você espera de um aluno de Ciência da Computação? O mínimo é que ele saiba programar, e o que é que eles vão fazer? Vão fazer um trabalho teórico, que não precise programar, fazer um trabalho de Banco de Dados, que não precise programar, vão fazer uma API, então o aluno faz uma API, e ele usa uma interface gráfica que ele usa as técnicas de construção rápida de interface gráfica, e ele não sabe o que está fazendo, e depois esse aluno ele não consegue se colocar no mercado, e nós não temos um forte engajamento com os processos produtivos.