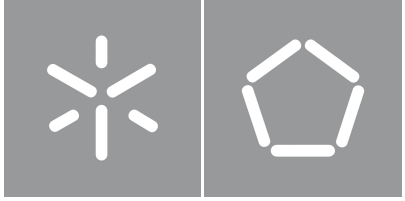
Universidade do Minho
Escola de Engenharia

Mariana Gonçalves Marques

Comparação de plataformas low-code

janeiro 2023



Universidade do Minho
Escola de Engenharia

Mariana Gonçalves Marques

Comparação de plataformas low-code

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática

Trabalho efetuado sob a orientação de
João Saraiva
Jácome Cunha

janeiro 2023

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:

CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/4.0/ *[Esta é a mais restritiva das nossas seis licenças principais, só permitindo que outros façam download dos seus trabalhos e os compartilhem desde que lhe sejam atribuídos a si os devidos créditos, mas sem que possam alterá- los de nenhuma forma ou utilizá-los para fins comerciais.]*

Agradecimentos

Queria em primeiro lugar agradecer aos meus orientadores, professor João Saraiva e professor Jácome Cunha, que prontamente aceitaram orientar-me neste tema, guiando-me ao longo de todo o processo de desenvolvimento da dissertação, apresentando por várias ocasiões sugestões fundamentais para o melhoramento do conteúdo deste tema, tornando a realização desta dissertação um verdadeiro processo de aprendizagem.

Para além disso, gostaria ainda de agradecer à minha família, especialmente à minha mãe, Manuela, à minha tia Graça e por último à minha irmã, Inês, sem as quais não seria possível atingir este objetivo.

Gostava ainda de agradecer a todos os meus amigos, que me acompanharam ao longo deste percurso, que ouviram as minhas preocupações e me aconselharam quando necessário, dando um especial agradecimento a todos aqueles que cederam um bocadinho do seu tempo para participar no meu estudo, permitindo assim, tornar a minha dissertação mais completa.

-

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.

Universidade do Minho, Braga, janeiro 2023

Mariana Gonçalves Marques

Resumo

A mudança contínua do mercado de *software*, acompanhado pelo constante aparecimento de novas tecnologias, pressiona as equipas de tecnologia a entregarem os seus projetos mais rapidamente sem comprometer a qualidade do produto. Assim, o surgimento de plataformas low-code (LCP) tornou-se inevitável e rapidamente se alastrou pelo mercado, sendo estimado que em 2023, 50% das médias e grandes empresas recorram a este tipo de plataformas.

O desenvolvimento *low-code* de *software* é um paradigma emergente, que combina a utilização mínima de código com interfaces gráficas interativas, que possibilita o desenvolvimento rápido de aplicações.

Assim, esta dissertação tem como objetivo o estudo de plataformas *low-code*, passando para um estudo pormenorizado das plataformas *Mendix*, *MS PowerApps* e *Outsystems*. Este estudo consiste na análise de várias características destas plataformas, sendo feitas também algumas comparações entre estas. Para além disso, é ainda feito um estudo empírico usando vários participantes de várias universidades. Deste modo, este estudo pode também ser usado para avaliar a usabilidade das plataformas.

Palavras-chave *low-code*, *Mendix*, *MS PowerApps*, *Outsystems*, plataformas de desenvolvimento *low-code*

Abstract

The continuous change of the market, together with the recurrent evolution of new technologies, pressures technology teams to deliver their projects faster without compromising product quality. Thus, the emergence of low-code platforms (LCP) became inevitable and quickly spread through the market. It is estimated that by 2023, 50% of medium and large companies will use this type of platform.

Low-code software development is an emerging paradigm that combines the minimal use of code with interactive graphical interfaces, which allows for faster software applications' development.

Therefore, this dissertation aims to study low-code platforms, going through a detailed study of Mendix, MS PowerApps, and Outsystems platforms. This study consists of the analysis of several features of these platforms, being made also some comparisons between them. Additionally, this study can also be used to evaluate the usability of the platforms.

Keywords low-code, Mendix, MS PowerApps, Outsystems, low code development platforms

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

API Application Programming Interface.

CSS Cascading Style Sheets.

HTML HyperText Markup Language.

IoT Internet of Things.

JSON JavaScript Object Notation.

LCDP Low Code Development Platform.

LCP Low Code Platform.

PaaS Platform-as-a-service.

RAD Rapid Application Development.

REST Representational State Transfer.

SQL Structured Query Language.

UI User Interface.

UML Unified Modeling Language.

URL Uniform Resource Locator.

Capítulo 1

Introdução

1.1 Contextualização

A mudança contínua do mercado, acompanhada pelo constante aparecimento de novas tecnologias, pressiona as equipas de tecnologia a entregarem os seus projetos mais rapidamente sem comprometer a qualidade do produto [27]. Assim, o surgimento de plataformas low-code (LCP) tornou-se inevitável e rapidamente se alastrou pelo mercado, sendo estimado que em 2023, 50% das médias e grandes empresas recorram a este tipo de plataformas [12]. Desta forma, tornou-se possível a rápida elaboração, modificação e entrega de aplicações com a utilização de plataformas low-code, em que o uso de código é nula ou quase inexistente devido às suas ferramentas de arrastar e largar, requerendo um esforço mínimo na instalação, configuração, treino e implementação da aplicação [30, 35]. Esta facilidade torna possível que para além de profissionais de tecnologia, elementos de outras equipas possam também produzir software [28].

Com o aumento na utilização deste tipo de plataformas, torna-se importante conhecer as plataformas *low-code* disponíveis no mercado, assim como fazer uma análise comparativa entre elas, permitindo inferir as suas capacidades.

Como tal, cada vez surgem mais negócios dedicados à criação dessas mesmas plataformas. No entanto, algumas empresas destacam-se mais nesta tarefa que outras.

Na criação de plataformas de *low-code*, geralmente alguns conceitos são tidos em conta, como a abordagem de desenvolvimentos de *software* orientada por modelos, um rápido desenvolvimento de aplicações, geração de código automático e por último programação visual [19].

Assim, no contexto desta dissertação, vamos estudar mais aprofundadamente três plataformas, *Mendix*, *MS PowerApps* e *Outsystems*. Para isso, desenvolvemos uma aplicação simples para cada umas destas plataformas selecionadas, permitindo realizar uma análise comparativa entre estas, podendo concluir várias características de cada plataforma e averiguar a sua usabilidade para o desenvolvimento de

aplicações.

Além disso, esta tese tem ainda como objetivo perceber de que forma vários tipos de pessoas se sentem face à possibilidade de utilização deste tipo de plataformas, onde alguns participantes foram desafiados a desenvolver algumas tarefas definidas nas três plataformas escolhidas.

1.2 Estrutura da dissertação

Esta dissertação está dividida em cinco capítulos. O **Capítulo** 1 é referente à introdução, onde é feita uma contextualização do tema da dissertação, que permite apresentar ao leitor alguma informação geral sobre plataformas *low-code* e explicitar os objetivos principais da dissertação. O **Capítulo** 2 corresponde ao estado de arte, onde são apresentados trabalhos que vão de encontro ao tema desta dissertação, sendo ainda apresentada uma introdução a plataformas *low-code*, isto é, exploração e explicação do que são plataformas *low-code*, de que necessidades do mercado levaram ao seu surgimento, a arquitetura deste tipo de plataformas, as vantagens e desvantagens na sua utilização, e ainda uma apresentação das plataformas mais utilizadas no mercado hoje em dia. No **Capítulo** 3, são apresentadas as plataformas escolhidas para prosseguir com o tema da dissertação, explicitando algumas características do funcionamento das mesmas. Além disso, é apresentado o caminho percorrido para o desenvolvimento das aplicações criadas, sendo apresentadas algumas características aprendidas após a utilização destas plataformas, sendo feita assim uma comparação entre as três plataformas escolhidas. No **Capítulo** 4 é apresentado um estudo realizado com vários participantes, tanto pertencentes ao meio de engenharia informática como de outros meios, onde a estes é designada uma tarefa para realizarem numa das plataformas, permitindo retirar novas conclusões acerca do uso deste tipo de plataformas.Neste capítulo, os participantes relativos à *MS PowerApps* revelaram ter maior facilidade na execução das tarefas quando comparados aos participantes das outras plataformas. Como seria de esperar, os participantes de informática revelaram, de forma geral, maior facilidade no desenvolvimento das tarefas atribuídas. Para além disso, foi ainda observado que, no final desta experiência, os participantes demonstraram aumentar o seu interesse no uso deste tipo de plataformas, sendo este aumento mais sentido nos participantes de áreas de informática. Por último, no **Capítulo** 5 são apresentadas as conclusões finais sobre o tema desta dissertação.

Capítulo 2

Estado da arte

2.1 Background Low-code

O surgimento de um conceito semelhante ao de *low-code* surgiu nos anos 80 com o surgimento da metodologia RAD (Rapid Application Development), que prioriza o ciclo de vida de desenvolvimento de forma a fornecer resultados mais rápidos e com maior qualidade quando comparados com o ciclo de vida tradicional [22]. Este processo de desenvolvimento procura utilizar e focar esforços e recursos na prototipagem do produto [14]. Desta forma, o ciclo de vida de desenvolvimento com RAD é o apresentado na Figura 1.

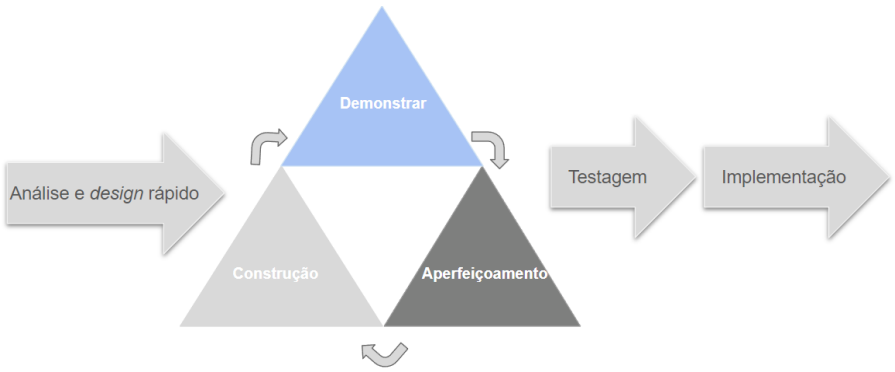


Figura 1: Ciclo de vida de RAD

No entanto, em 2014 a *Forrester Research* surgiu com o termo *low-code* para uma ferramenta muito mais atualizada quando comparada com metodologias *RAD* [27].

Com o aparecimento de plataformas *low-code* tornou-se possível desenvolver *software* sem a obrigatoriedade de vastos conhecimentos de programação por parte do desenvolvedor, requerendo pouco ou nenhum código na construção de aplicações e processos. Desta forma, uma plataforma *low-code* permite que o trabalho de codificação seja feito utilizando interfaces visuais com funcionalidades simples de lógica de arrastar e largar em vez de extensas e complexas linguagens de programação [10]. No entanto,

normalmente as plataformas *low-code* permitem sempre a adição de código por parte do utilizador [27].

2.1.1 Arquitetura

Por norma, plataformas de desenvolvimento rápido de aplicações seguem uma arquitetura por camadas [28], como representado na Figura 2.

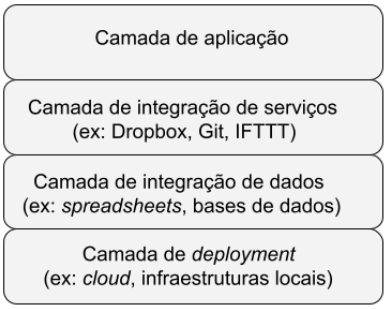


Figura 2: Arquitetura plataformas low-code

Na **camada de aplicação**, é apresentado aos utilizadores o ambiente gráfico, onde encontram os recursos necessários para a construção das interfaces gráficas da aplicação pretendida, construindo modelos com finalidade de especificar o comportamento da mesma [28].

Na **camada de integração de serviços** é feita a comunicação entre vários serviços, usando API's e mecanismos de autenticação [28].

A **camada de integração de dados** possibilita a manipulação dos dados e integração destes para diversas fontes [28].

Por último, a **camada de deployment** cujo intuito é realizar o *deploy* da aplicação, isto é, tratar da implementação da aplicação, que dependendo da LCDP pode ser feita em *cloud* ou em infraestruturas locais. Para além disso, nesta camada são ainda tratadas a contentorização e orquestração das aplicações [3][1] [28]. A contentorização de aplicações consiste em colocar todo o código necessário para o desenvolvimento da aplicação em contentores com ficheiros de configuração, bibliotecas e dependências associadas necessárias à sua execução. A orquestração de aplicações consiste na configuração, gestão e coordenação automatizada de serviços, aplicações e sistemas de um computador, facilitando a gestão dos fluxos de trabalho e tarefas complexas [6].

[1] https://azure.microsoft.com/pt-pt/resources/cloud-computing-dictionary/what-is-a-container/

2.1.2 Utilizadores de plataformas low-code

Os utilizadores de plataformas *low-code* encontram-se normalmente dividimos em dois grupos, os *Citizen Developers* e *IT Developers* [8, 21].

Se por um lado um **IT Developer** é aquele que com conhecimentos e experiência em programação opta pelo uso de plataformas *low-code* para possivelmente facilitar o seu trabalho, um **Citizen Developer** é um utilizador que sem qualquer conhecimento em linguagens de programação cria novas aplicações ou modifica as já existentes. Assim, não existe a necessidade de recorrer às equipas tecnológicas, que por vezes se encontram sobrecarregadas, e não conseguem desenvolver os pedidos dos clientes atempadamente [8, 21].

De acordo com pesquisas realizadas por *Gartner*, 41% dos utilizadores de plataformas *low-code* consideram-se *Citizen Developers* [16].

2.1.3 Propriedades de plataformas low-code segundo a literatura

Por norma, uma plataforma *low-code* apresenta um ambiente visual para desenvolvimento da interface do utilizador, fluxos de trabalho e modelos de dados dos projetos, para além disso, exibe ainda um conjunto de conectores para vários *back-ends* ou serviços, tratando automaticamente de armazenamento, recuperações e de estruturas de dados. Por último, estas plataformas geralmente apresentam ferramentas automatizadas para construção, depuração e implementação, tratando ainda da testagem da aplicação [25].

Como seria de esperar, o uso de plataformas *low-code* visa facilitar o desenvolvimento de *software*, tornado este processo mais rápido. Este fenómeno torna-se possível não só devido as suas tecnologia de arrastar e largar mas também devido a funcionalidades prontas a usar em que é eliminada a necessidade de construção de módulos centrais para aplicações desde raiz [9].

Ao utilizar este tipo de plataformas, os utilizadores deixam de ter de lidar com detalhes por vezes demorados ao longo do desenvolvimento do *software*. Desta forma, deixam de ter a necessidade de tratar de detalhes relacionados com a configuração de infraestruturas, gestão da integridade de dados em diferentes ambientes, aumentando a robustez do sistema [28].

Mas estas não são as únicas vantagens da sua utilização. Como referido anteriormente, com a utilização deste tipo de plataformas os desenvolvedores têm a possibilidade de se adaptar aos objetivos empresariais crescentes e às tendências na indústria de *software*, permitindo a monitorização dos fluxos de trabalho e processos das aplicações acompanhando a sua eficiência em cada fase de desenvolvimento

[29]. Por fim, este tipo de plataformas possibilita uma monitorização comparativa entre o desempenho da aplicação com as outras aplicações existentes no mercado [29]. Para além disso, a compatibilidade com diversos dispositivos torna-se útil, tornando possível que os utilizadores construam aplicações que possam também ser executadas em todas as plataformas e nos principais dispositivos. Por último, o uso deste tipo de plataformas possibilita ainda a proteção contra o *churn* tecnólogo, referente ao ciclo de substituição dos produtos, linguagens de programação existentes por outros mais recentes [5, 29].

Embora as vantagens na utilização de plataformas *low-code* sejam praticamente consensuais, estas podem também ter algumas desvantagens. É sabido que estas plataformas oferecem uma maior eficiência, no entanto, estas podem não oferecer necessariamente segurança. As aplicações *low-code* dependem regularmente de redes, dados de *upstream*, isto é, o fluxo de dados com tráfego no sentido do utilizador para o servidor, e de outras aplicações que, quando comprometidas, podem introduzir dados defeituosos ou vulnerabilidades incapacitantes para as aplicações ou processos [18]. Para além disso, pode ainda apontar-se o *lock-in* presente em algumas plataformas *low-code*, isto é, por vezes uma plataforma *low-code* pode estar presa a um fornecedor *cloud* específico. Este aprisionamento pode tornar-se um problema por ser difícil mover bases de dados uma vez que estejam configurados especialmente numa migração em *cloud* que ao envolver uma movimentação de dados para um ambiente totalmente diferente pode levar a uma reformatação total dos dados. Além disso, quando uma nova ferramenta é incorporada em processos já existentes de uma empresa, essa empresa pode tornar-se dependente dessa mesma ferramenta [7, 26]. A utilização de plataformas *low-code*, revela também uma menor possibilidade de customização do produto quando comparado à utilização de código tradicional, o que poderá levar ao surgimento de produtos semelhantes [31].

A *Outsystems* realizou um questionário em 2019 com um total de 3300 profissionais de tecnologia distribuídos por vários continentes. Este estudo tinha como intuito concluir quais os principais fatores que leva estes profissionais a adotar ou não plataformas *low-code* [29]. Foram apontadas como principais razões para a utilização de plataformas *low-code*, a aceleração digital e o aumento da procura no mercado, isto é, conseguir responder melhor às mudanças e requisitos constantes do mercado (ver Figura 3).

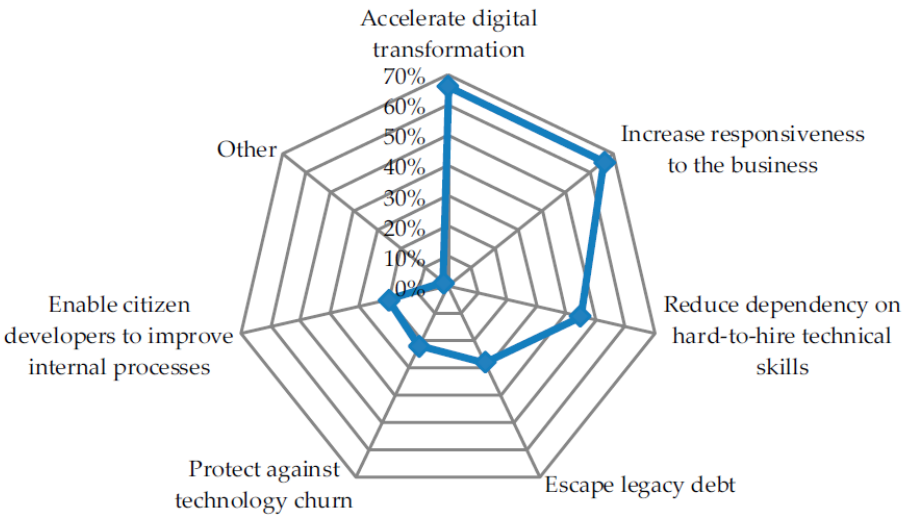


Figura 3: Razões para uso de plataformas low-code [29] (Figura em inglês)

Das razões mais referidas para a não utilização destas plataformas destacam-se, a falta de conhecimento sobre as plataformas e a preocupação com a possibilidade de *lock-in*. A Figura 4 sumariza os resultados.

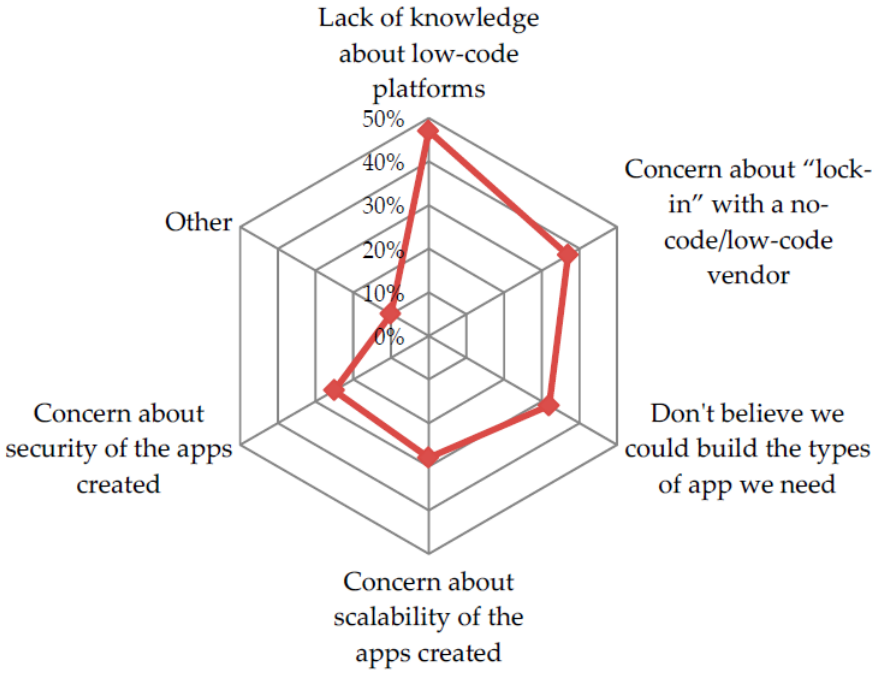


Figura 4: Razões para não utilizar nem considerar usar plataformas low-code [29] (Figura em inglês)

2.1.4 Plataformas low-code

Cada vez mais empresas utilizam plataformas *low-code*, sendo estimado que em 2023 50% das médias e largas empresas recorram a estas plataformas [12].

Como tal, cada vez surgem mais negócios dedicados à criação dessas mesmas plataformas. No entanto, algumas empresas destacam-se mais nesta tarefa que outras.

Na criação de plataformas de *low-code*, geralmente alguns conceitos são tidos em conta, como a abordagem de desenvolvimentos de *software* orientada por modelos, um rápido desenvolvimento de aplicações, geração de código automático e por último programação visual [19].

O quadrante mágico de *Gartner*, representa a influência de várias empresas desenvolvedoras de plataformas *low-code* no mercado em 2021. Para tal, ordena cada quadrante segundo a habilidade de execução, isto é, capacidade de execução e venda do produto, e a *completeness of vision*, isto é, a estratégia de *markting* e inovação do produto [19] (ver Figura 5).

Figura 5: Plataformas de Desenvolvimento Low-Code 2021, quadrante mágico de Gartner [36]

Em 2021, as empresas consideradas líderes nesta área de estudo foram o *Mendix*, *Outsystems*, *MS PowerApps*, *Salesforce* e *ServiceNow*.

O **Mendix** é uma plataforma *low-code* subsidiária da *Siemens* que não requer a utilização de código e todas as funcionalidades podem ser acedidas por capacidades de arrastar e largar [28, 36]. Esta plataforma distingue-se das restantes pela sua capacidade de criação de algumas aplicações com conectores pré-construídos, incluindo os da IOT e inteligência artificial [28].

A **Outsystems**, fundada em 2001, é uma empresa focada no desenvolvimento de plataformas que permitem a criação e modificação de aplicações *mobile* e *desktop*, que correm em infraestruturas locais ou em *cloud* e soluções mistas [28, 33]. A **Outsystems** disponibiliza algumas capacidades avançadas de

low-code como, por exemplo, o desenvolvimento com inteligência artificial, integração contínua e *deployment* contínuo (CI/CD) e testes à aplicação. Futuramente pretende ainda introduzir testes automáticos utilizando inteligência artificial, gestão de *API* nativo e ainda uma grelha de dados inteligente com intuito de facilitar a inserção e catalogação de dados sem descontinuidades [36].

A **Microsoft** disponibiliza um conjunto de serviços, conectores, aplicações e plataformas **Microsoft PowerApps**, que inclui as plataformas *low-code* **Power Automate** e **Dataverse** [34, 36]. Estas plataformas utilizam mecanismos de arrastar e largar e oferece aos seus utilizadores uma coleção de *templates* que possibilita a reutilização de modelos já desenvolvidos [28]. Este tipo de plataformas pode ainda integrar-se com muitos serviços Microsoft [28].

A **Salesforce** oferece a plataforma *low-code* **Salesforce Platform**, onde as ferramentas apresentadas vão desde o uso de interfaces visuais, não sendo requerido a utilização de código, até a utilização de código com *Apex* e *JavaScript* [28]. Algumas das plataformas de *low-code*, como referido anteriormente, permitem o uso o desenvolvimento de inteligência artificial nas suas aplicações, no entanto, a *Salesforce* não oferece essa capacidade.

A **ServiceNow**, oferece a plataforma *low-code* **Now Platform** onde se encontram a **App Engine** e **Creator Workflows**. Esta plataforma está focada em desenvolver mais processos e automatismos para oferecer suporte aos *citizen developers* [35, 36].

2.2 Revisão de Literatura

O estudo exploratório de Bock and Frank utiliza sete plataformas *low-code*, visando avaliá-las e concluir algumas características que as tornem únicas quando comparadas ao desenvolvimento tradicional. As plataformas usadas foram a *Appian*, *Bonita Platform*, *Mendix*, *Microsoft PowerApps*, *Pega*, *Quickbase*, *Wavemaker*. Os autores concluiram que, o que distingue este tipo de plataformas do desenvolvimento tradicional, é que estas integram, num único ambiente, vários sistemas comuns e componentes tradicionais de *design* de modo a reduzir os esforços na implementação de aplicações. Este estudo demonstrou ainda que, as plataformas vendidas sob a legenda '*low-code*' não constituem uma classe bem definida de tecnologia, com um conjunto uniforme de características. Para além disso, estes autores acreditam não poder atribuir um termo a usar para este tipo de plataformas. No entanto, acham que manter uma terminologia constante para este tipo de plataformas é algo essencial, sendo importante atualmente considerar *low-code* como um termo científico [12].

De forma a perceber de que maneira as plataformas *low-code* podem facilitar o desenvolvimento de

aplicações seguras e escaláveis, Talesra and Nagaraja realizaram um estudo utilizando a plataforma *Oracle Apex*. Estes autores acreditam que esta plataforma permitiu o desenvolvimento de uma aplicação segura e de alta performance, com baixo custo de implementação da aplicação. Embora o uso de plataformas *low-code* traga vários benefícios, os autores acreditam que estas apresentam algumas limitações, como a fraca oferta de customização das aplicações criadas e a dependência ao fornecedor da plataforma. Isto é, se um utilizador investir todo o seu tempo e conhecimento numa plataforma *low-code*, será muito mais trabalhoso utilizar outro tipo de plataformas deste género, fazendo com que este utilizador se prenda à plataforma escolhida. Para além disso, segundo os autores, as opções de integração são reduzidas. Estes acreditam que este tipo de plataformas não conseguem eliminar a necessidade das competências do desenvolvimento tradicional de aplicações. Por último, afirmam que se este tipo de plataformas, com todas as suas ferramentas associadas, não cumprir os requisitos impostos pelo mercado para desenvolvimento de aplicações, será necessário recorrer aos conhecimentos de um desenvolvedor de código tradicional, com todas as suas competências de programação [32].

O artigo escrito por Lichtenthäler et al. , estuda três plataformas *low-code*: *Appian*, *Outsystems* e *MS PowerApps*. Os principais objetivos deste estudo foram perceber as vantagens e desvantagens destas plataformas, e responder à questão "Até que ponto as plataformas de desenvolvimento *low-code* permitem o rápido desenvolvimento de aplicações apesar da baixa experiência anterior?". Para isso, utilizaram participantes experientes em desenvolvimento de *software* tradicional, sem conhecimento de plataformas *low-code*.

Os autores concluíram que foi possível realizar rapidamente grande parte das tarefas pretendidas recorrendo a plataformas *low-code*, apesar da baixa experiência dos participantes neste tipo de plataformas. No entanto, estes consideraram que o seu conhecimento em engenharia de *software*, como, por exemplo, o seu conhecimento em *design* clássico de aplicações, foi imprescindível para o sucesso da utilização destas plataformas. Acreditando que, utilizadores sem conhecimentos gerais em engenharia de *software* também conseguiriam recorrer a estas plataformas para o desenvolvimento rápido de aplicações, no entanto, de uma forma mais trabalhosa. Além disso, tal como referido noutros estudos, estes autores notaram que o vasto conhecimento numa destas plataformas pode ser difícil de aplicar noutra plataforma *low-code*. Por esta razão, os autores concluem que seria mais proveitoso uma formação em ciências da computação do que uma formação neste tipo de plataformas.

Por fim, [20] constataram que a *Outsystems* revelou ser a melhor plataforma para *citizen developers*, sendo aquela que permitiu um melhor desenvolvimento nesta experiência. No caso na *PowerApps*, esta demonstrou oferecer vários serviços do ambiente *cloud Microsoft Azure*, sendo mais fácil integrar os

serviços associados à *Microsoft*. No caso da *Appian*, e contrariamente às outras plataformas, oferece uma linguagem de *scripting* personalizada com as restantes componentes. Esta plataforma permite o desenvolvimento rápido de aplicações sem necessidade de vastos conhecimentos da plataforma [20].

Para perceber como se comportam as plataformas *MS PowerApps*, *Mendix* e *Outsystems*, Gürcan and Taentzer , desenvolveram dois tipos de aplicações nestas plataformas. Com este estudo concluíram que as três plataformas foram bem sucedidas no desenvolvimento de ambas as tarefas, no entanto, das três plataformas, apenas a *Outsystems* apresentou suporte de encaminhamento para páginas individuais, notando que no caso da *MS PowerApps* e no *Mendix*, é gerado um *URL* único para toda a aplicação. Além disso, foi concluído que, no caso da necessidade de juntar a aplicação desenvolvida com um *backend* local, apenas o *Mendix* mostrou ser bem sucedido, uma vez ser o único com possibilidade de execução local [17].

Capítulo 3

Análise empírica das plataformas

Para atingir o objetivo desta dissertação, foram selecionadas as três plataformas *low-code* melhor cotadas segundo o estudo *Gartner* referido anteriormente. Desta forma, estas plataformas serão exploradas mais pormenorizadamente, explicitando algumas características do seu funcionamento.

3.1 Introdução às plataformas

3.1.1 Mendix

O *Mendix* é uma plataforma *low-code* criada com o intuito principal de desenvolvimento de aplicações *mobile*, *web* e *IoT*.

Mendix é considerado um meio para desenvolvedores experientes.

A interface do *Mendix*, segue o conceito "o que tu vês é o que recebes", tendo o intuito de tonar a experiência do utilizador desta plataforma o menos complexo possível [24]. Assim, aos utilizadores desta plataforma é permintido o desenvolvimento de *software* sem código de programação, recorrendo a elementos pré-programados através de funciona lides de arrastar e largar[17]. Contudo, estes elementos podem sofrer alguns ajustes adicionais, recorrendo ao uso de *Java* ou *JavaScript* [17].

Os utilizadores de *Mendix* podem tratar da lógica dos eventos das suas aplicações utilizando fluxos, permitindo adicionar validações, cálculos, peças de integração e outras atividades [24].

No *Mendix*, não há a necessidade de decidir que base de dados usar, nem de a gerir, adicionalmente, não exige consultas *SQL* nem funções gerais para a arquitetura da aplicação [24].

A estrutura da base de dados, segue os princípios UML (*Unified Modeling Language*), composto por três componentes principais: entidades, atributos e associações.

As entidades internas nesta plataforma, podem ser criadas com atributos que guardam os dados num formato tabular [17]. As associações (isto é, a relação entre as entidades com base na carnalidade

de um para muitos, um para um, ou muitos para muitos) armazenam a chave primária das entidades relacionadas automaticamente [24].

A análise geral e gestão das entidades criadas é feita no modelo de domínio desta plataforma, que suporta um *design* orientado a objetos. É feita ainda uma distinção entre dados persistentes e não persistentes, sendo usados conectores quando surge a necessidade de conexão a dados externos ou envio de dados internos. Estes conectores são disponibilizados no *Marketplace* [17].

Quando a aplicação desenvolvida está pronta para ser publicada, esta pode ser lançada num ambiente *cloud* ou localmente.

3.1.2 Outsystems

A *Outsystems*, permite desenvolver aplicações *web* e aplicações *mobile*, apresentando um conjunto de ferramentas que cobre todas as fases do desenvolvimento da aplicação pretendida [17].

Para desenvolver a parte de *UI* (User Interface) de cada aplicação desenvolvida, cada desenvolvedor pode utilizar *templates* e blocos pré-construídos que funcionam para diversos dispositivos, havendo ainda a possibilidade de estender estes elementos recorrendo a *HTML, JavaScript* e *CSS* [23]. Como seria de esperar, todos os aspetos relacionados com o *backend* são desenvolvidos a partir da linguagem visual da *Outsystems*, recorrendo também ao *Service Studio*, *IDE* cedida por esta plataforma [17, 23].

Quando o conteúdo a visualizar é uma componente gráfica, a *Outsystems* disponibiliza uma pré-visualização da parte gráfica da aplicação, tornando possível um acompanhamento ao longo do desenvolvimento da interface da aplicação e alteração desta, caso necessário [15].

Na *Outsystems*, os dados podem ser inseridos manualmente ou importados através de ficheiros *Excel* via *Bootstrap* e transferidos para entidades [17].

Em semelhança ao *Mendix*, na *Outsystems* também os dados podem ser armazenados internamente em entidades. Cada desenvolvedor cria a entidade internamente, preenchendo-a com os atributos pretendidos. Contrariamente ao *Mendix*, o conteúdo destas entidades pode ser visto e editado [17]. Estas entidades referidas anteriormente, são similares a tabelas, sendo estas compostas por chaves primárias que são criadas automaticamente como um atributo chamado “Id” [15].

Na *Outsystems*, a lógica da aplicação é definida por ações que podem executar no servidor e ações que podem executar diretamente no dispositivo do cliente [17].

Após o desenvolvimento da aplicação pretendida, o desenvolvedor pode proceder à execução desta na *cloud* ou em infraestruturas locais. Para além disso, a *Outsystems* providencia recursos que possibilitam publicar a aplicação via *URL* com um simples clique de um botão [28].

3.1.3 MS PowerApps

MS PowerApps é uma plataforma *low-code* que fornece duas opções de desenvolvimento de aplicações. **Canvas app** e **model-driven apps**. A primeira possibilita o desenvolvimento de aplicações de raiz, ao arrastar e largar componentes, permitindo fazer alguns ajustes no tamanho e formatação destes componentes. Quando o desenvolvedor estiver satisfeito com o *design* desenvolvido pode conectar às fontes de dados pretendidas, recorrendo a fórmulas básicas do *Excel*. Por outro lado, as *model-driven apps* desenvolvem a maioria dos componentes utilizados na interface, composta por dados escolhidos pelo utilizador a partir do *Dataverse* [11, 13].

Para proceder ao desenvolvimento da lógica para os eventos pretendidos, primeiro tem de se considerar a aplicação criada. No caso das *Canvas Apps*, a lógica pode ser feita de uma forma mais simples, sendo apenas necessário recorrer a fórmulas do género do *Excel*, sendo processada no dispositivo em que a aplicação está a ser executada. Por outro lado, no caso das *model-driven apps* a lógica dos eventos é mais complexa, sendo necessário recorrer a fluxos para que a lógica de eventos seja desenvolvida [17].

Na *MS PowerApps*, são disponibilizadas várias conexões na *cloud* para fontes de dados e aplicações externas, como, por exemplo, o *Excel*, *Sharepoint*, *SAP*, entre outros [17].

Podem ainda ser criados conectores personalizados para a conexão de *APIs* ou outros serviços que não são oferecidos por *default*, sendo estas funcionalidades pagas na *PowerApps* [17].

A *PowerApps* permite ainda o acesso ao *PowerAutomate*, possibilitando a criação de fluxos em que dados são enviados entre aplicações e serviços [17].

3.2 Desenvolvimento das aplicações

Tendo presente a noção das plataformas *Mendix*, *MS PowerApps* e *Outsystems*, foi realizado um estudo sobre estas, visando perceber a sua usabilidade. Por último, fez-se uma análise comparativa entre estas plataformas, tanto relativamente às funcionalidades que cada uma destas apresenta como à sua facilidade de utilização e tempo gasto necessário para realização das tarefas pretendidas. Para tal, foi desenvolvida uma aplicação simples para cada das plataformas.

A implementação destas aplicações dividiu-se em duas partes. Primeiramente, foi realizada a autenticação, ou seja, as páginas de *Login* e *Resgister*. Seguidamente, e após o *user* estar autenticado, foi implementada uma página onde serão apresentados uma lista de livros (fornecida pela Google API), onde o *user* pode escolher o livro que quiser e aceder à página de detalhes de cada livro.

Para que o grau de conhecimento de cada plataforma não influenciasse a utilização da próxima (isto

é, conhecendo algumas *features* de uma plataforma as restantes poderiam ser semelhantes em certos aspetos), as plataformas foram utilizadas de forma intercalada, assim como a realização das tarefas.

A seguir será explicado o processo da realização de cada tarefa em cada uma das plataformas.

3.2.1 Mendix

Autenticação

Relativamente à página de *login* e de registo, foi necessário a criação das entidades e respetivos atributos no modelo de domínio. Desta forma, foram criadas três entidades: "Users", "Login" e "Registration´ ´. Embora estas três tabelas tenham praticamente os mesmos atributos, é imprescindível que sejam criadas, pois na tabela de "Users" é armazenada de forma persistente toda a informação dos utilizadores criados, enquanto nas tabelas "Login" e "Registration" são utilizadas como estruturas temporárias, para serem usados os dados de utilizadores que pretendam criar novas contas ou para passar dados a partir das caixas de texto (ver Figura 6).

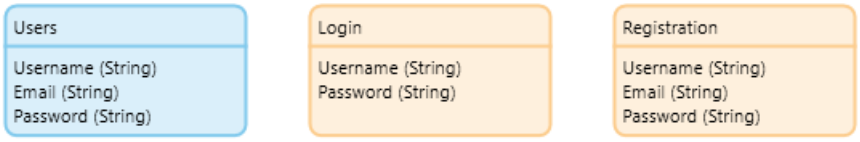


Figura 6: Modelo de domínio para a criação e utilização dos dados dos utilizadores

Após a criação das estruturas que nos permitem armazenar e utilizar dados dos utilizadores, será necessário a utilização de fluxos que nos permitirão tratar da parte lógica da aplicação. Isto é, um conjunto de elementos que possibilitem que ao ocorrer um determinado evento (por exemplo, acionamento de um botão), aconteça algo.

No *Mendix* existem dois tipos de fluxos que permitem expressar a lógica da aplicação, **nanoflows** e **microflows**. A maior diferença entre estes dois fluxos é que *microflows* são executados no servidor durante o tempo de execução da aplicação, não sendo por isso usados em aplicações *offline* [1]. Por outro lado, os *nanoflows* executam diretamente no *browser*/dispositivo, podendo ser utilizados numa aplicação *offline*, o que poderá trazer benefícios de rapidez para tarefas que não precisem de aceder ao servidor [2].

Há um conjunto de elementos nestes fluxos que permitem fazer a lógica de aplicações. Estes elementos podem dividir-se em cerca de seis tipos diferentes de elementos:

- Elementos de evento, onde podemos encontrar eventos de começo, fim, erro, continuação e paragem dos fluxos (ver Figura 7).

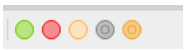


Figura 7: Elementos de evento nos fluxos no *Mendix*

- Elementos de decisão, compostos por três tipos de elementos diferentes, decisões baseadas em condições, seguindo um único fluxo de saída, decisões do tipo objeto, que faz uma escolha tendo em conta a especialização de um objeto selecionado, e por último a decisão de *merge* utilizado para combinar várias sequências de fluxos num só (ver Figura 8).

Figura 8: Elementos de decisão nos fluxos no *Mendix*

- Elemento de atividade que trata das ações executadas ao longo do fluxo. Dentro deste elemento existem muitas atividades fornecidas pelo *Mendix* como criação e mudança de variáveis e objetos, navegação para outros ecrãs, chamada de novos fluxos, busca de dados, entre outros (ver Figura 9).

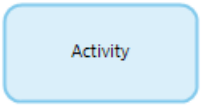


Figura 9: Elemento de atividades nos fluxos no *Mendix*

- Elemento *loop*, sendo este usado para iterar sobre uma lista de objetos. Neste elemento podemos escolher um *loop* sob a forma de um *foreach* e sob a forma de um *while*. Este não permite a existência de eventos de fim e começo no seu interior (ver Figura 10).

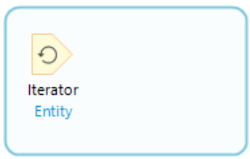


Figura 10: Elemento de ciclos nos fluxos no *Mendix*

- No elemento parâmetro são fornecidos os dados que servirão como *input* do fluxo (ver Figura 11).

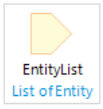


Figura 11: Elemento de parâmetro de entrada nos fluxos no *Mendix*

- Por último, no elemento de anotação, é permitido pôr comentários ao longo de todo o fluxo (ver Figura 12).

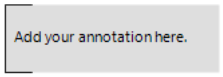


Figura 12: Elemento de anotação nos fluxos no *Mendix*

Desta forma, no caso da página de registo, para que ao clicar no botão "Resgister" os dados do novo utilizador fossem validados e adicionados foram necessários dois fluxos, o primeiro serve apenas para ser criado o objeto da entidade *Registration* para ser dado como *input* no fluxo da criação do registo em si (ver Figura 13).

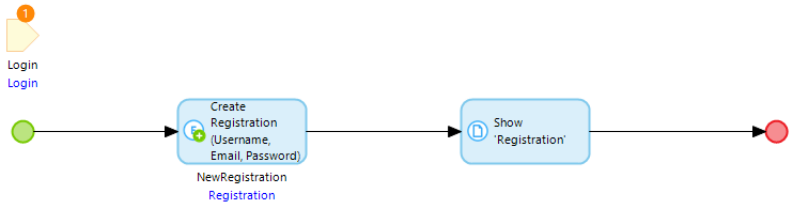


Figura 13: Lógica para criação do objeto "Registration"

Este segundo fluxo trata de toda a validação dos dados introduzidos pelo utilizador e caso estes sejam válidos adiciona-os na base de dados. Para isso é necessário validar se algum dos campos no formulário de registo está vazio usando elementos de decisão e mensagens de erro. No final, caso os dados fornecidos pelo utilizador já estejam a ser usados pelo outro, o fluxo acaba, cancelando o registo, caso contrário a conta é criada (ver Figura 14).

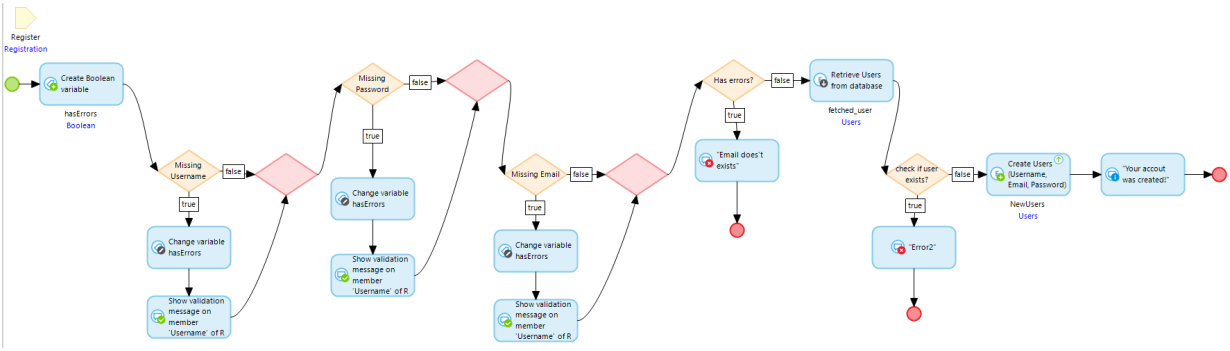


Figura 14: Lógica para validação e criação de novos utilizadores

Quanto à página de *login* foram necessários também dois fluxos para que o utilizador, caso existente na base de dados do projeto pudesse prosseguir na aplicação.

O primeiro fluxo, tal como anteriormente, apenas tinha como função a criação do objeto dado como *input* no fluxo de validação de dados (ver Figura 15).

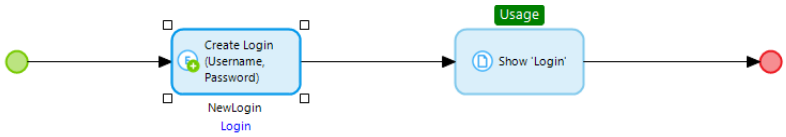


Figura 15: Lógica para criação do objeto "Login"

O segundo fluxo, apenas verifica se o *username* do utilizador é existente na base de dados, caso seja verifica a *password* inserida coincide com a dos dados da aplicação. Para além disso, neste fluxo, é ainda criado o objeto *Input* para que a página de listas de livros seja sempre obrigada a receber um termo de procura, podendo assim estar atualizada para as pesquisas por parte do utilizador, como será explicado na próxima secção. Este termo de procura tem como valor *default* a *String* "a" (ver Figura 16).

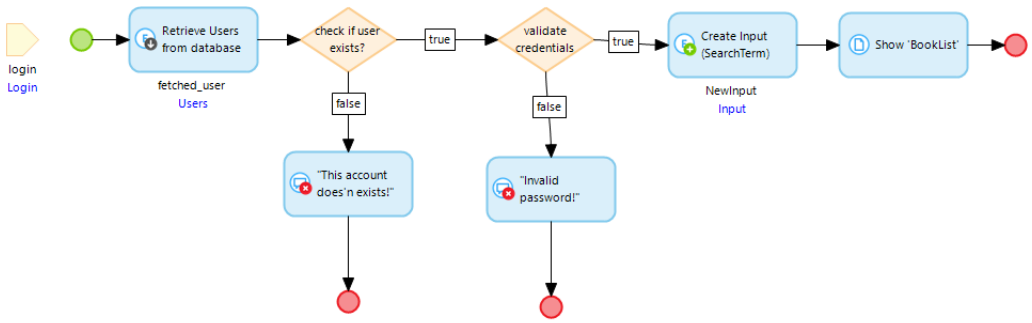


Figura 16: Lógica para validação de contas já existentes

É ainda de notar que no ecrã de *Login* foi adicionada a possibilidade de revelar e esconder a *password*. Esta funcionalidade não existia no *Mendix*, no entanto, foi rapidamente encontrada no *marketplace* do *Mendix*, onde vários componentes podem ser descarregados.

API

Para facilitar a compressão de todas as decisões tomadas, é necessário primeiro perceber de que forma é feito um *request* à *GoogleBookAPI*. Em contexto da aplicação pretendida serão apenas dois tipos de *request*, um para procurar um conjunto de livros por título e outro para procurar um livro específico tendo em conta o seu 'id'. Para a primeira opção o *request* é feito sob a forma de "`https://www.googleapis.com/books/v1/volumes?q=search+terms`" na segunda sob a forma de "`https://www.googleapis.com/books/v1/volumes/id`".

Inicialmente foi implementada a página referente a todos os livros. Para isso primeiramente foi necessário usar uma componente do *Mendix* "JSON structure", onde armazena um fragmento do JSON referente à API e o converte numa estrutura de esquema que o permitirá ser usado na componente *Import Mapping*, também esta disponibilizada pelo *Mendix*. No *Import Mapping*, é selecionado o esquema anterior e criadas as entidades e atributos correspondentes aos dados da API escolhida, sendo ainda feito o mapeamento destas mesmas entidades (a partir da opção *Map automatically*). O *Import Mapping* adiciona ainda estas entidades e correspondentes atributos ao modelo de domínio. Seguidamente, para facilitar a procura e possibilitar a pesquisa de livros específicos, foi adicionado ao modelo de domínio já estabelecido uma ligação a uma nova entidade "input", em que o atributo é apenas uma *String* "SearchTerm".

Depois de todas as entidades estarem criadas e mapeadas, é necessário criar os fluxos que possibilitarão o *request* à API. Para este efeito, foram necessários dois fluxos.

O primeiro fluxo é utilizado para que sempre que o utilizador pretenda fazer uma procura de livros por alguma palavra-chave, este seja novamente redirecionado para a página de livros, dando esse mesmo termo de procura como parâmetro de entrada, sendo assim atualizada a lista de livros consoante a procura do utilizador (ver Figura 17).

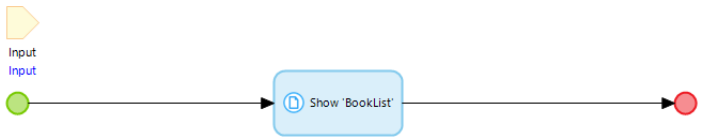


Figura 17: Lógica para atualização da lista de livros

O segundo fluxo, recebendo o termo de procura, faz a chamada à API, e retira a lista de "itens" para

esse termo, retirando ainda lista de objetos de cada livro resultante, ou seja, todas as características que depois queremos usar no ecrã de livros (ver Figuras 18 e 19).

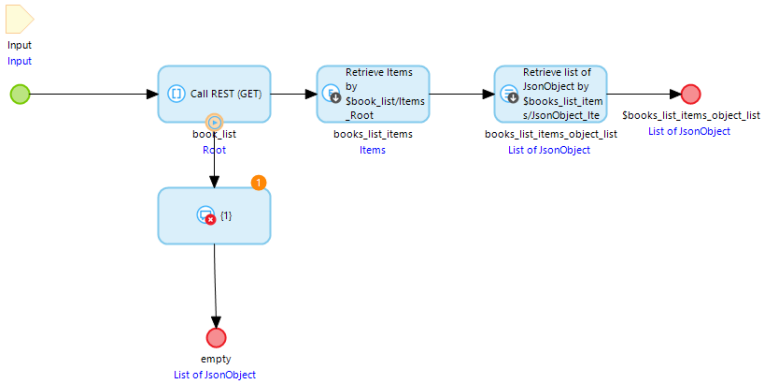


Figura 18: Lógica para obter a lista de livros pretendida

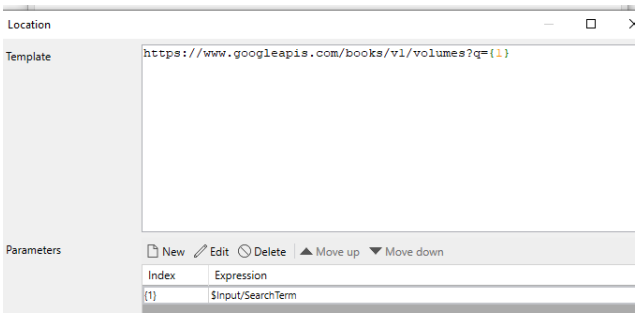


Figura 19: Detalhes chamamento REST

Por fim, no ecrã de lista de livros, é necessário usar *List view* para aceder aos dados gerados por ambos os fluxos e assim gerar a lista de livros pretendida. Neste ecrã são usados os dados referentes ao título e *thumbnail* de cada livro. Cada livro pode ser selecionado onde será gerada uma página de detalhes desse mesmo livro, que será explicado a seguir.

Para a página de detalhes de um livro foram necessários também dois fluxos. O primeiro fluxo, visa chamar o fluxo correspondente à chamada do livro exato e ainda redirecionar o utilizador para o ecrã de detalhes (ver Figura 20).

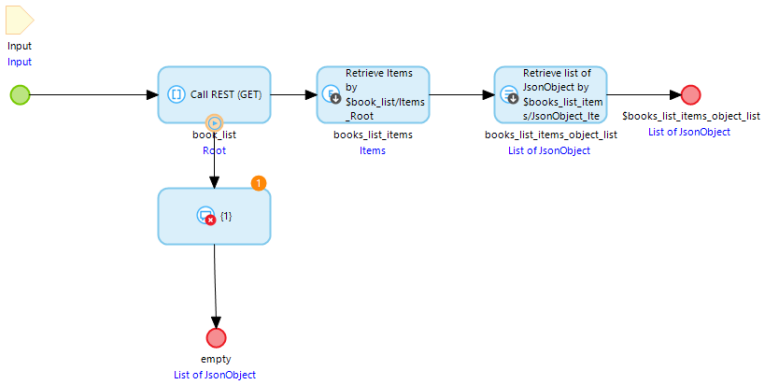


Figura 20: Lógica para obter id de um livro especifico

O segundo fluxo, recebe como parâmetro o id do livro (passado pelo fluxo anterior) e faz o *request* à API tendo em conta esse mesmo id, e retira dos dados obtidos a informação sobre o livro pretendido (ver Figura 21 e Figura 22).

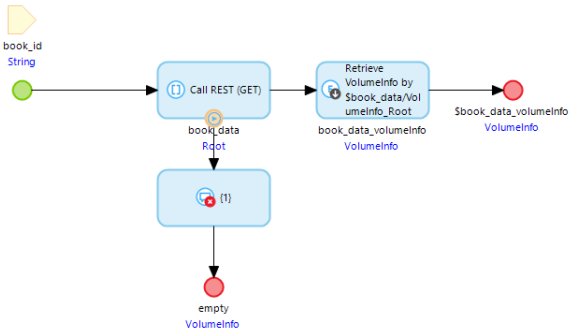


Figura 21: Lógica para obter informação de um livro específico

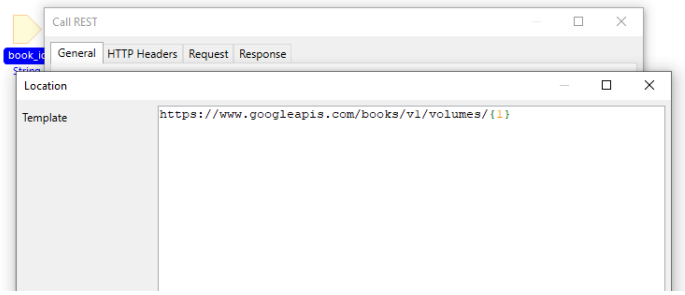


Figura 22: Detalhes chamamento REST

3.2.2 Outsystems

Autenticação

Para proceder à realização das páginas de *Login* e Registo, inicialmente seria necessário ponderar as estruturas de dados onde as informações dos utilizadores poderiam ser guardados. Para este efeito optou-se pela utilização da entidade "Users" já incluída na plataforma, sendo desta forma utilizados os atributos pré-definidos para essa mesma entidade (ver Figura 23).

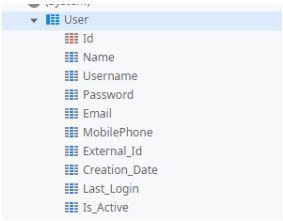


Figura 23: Entidade "Users"

Após solucionada a problemática relativa à estrutura de dados dos utilizares, é necessário passar à realização dos fluxos que tratarão da lógica de cada ecrã.

Desta forma, seria necessário recorrer a um fluxo para que a lógica de validação e armazenamento de dados fosse feita. Na *Outsystems*, na parte da interface em cada ecrã é possível adicionar dois tipos de ações que tratam da parte lógica das ações, como, por exemplo, lógica relativa ao acionamento de um botão. Estas ações dão-se pelo nome de *Client Action*, que executa a lógica do lado do cliente, e a *Data Action*, que retira dados complexos da base de dados, as *Data Action* com o lado do cliente começam a retirar dados simultaneamente quando o ecrã carrega [4].

Para a lógica do *Login*, a *Outsystems* disponibiliza já um conjunto de três fluxos que trata de toda a validação dos dados, exceções e caso os dados sejam validados (contidos na tabela de users), é redirecionado para a página de lista de livros.

Para a lógica da página de registo, existe primeiro uma validação de dados através do método *valid* disponibilizado pela *Outsystems*, seguidamente trata da encriptação da *password* escolhida pelo utilizador, e por último adiciona os dados à base de dados (ver Figura 24).

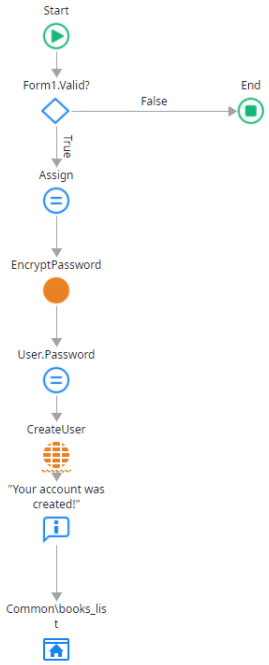


Figura 24: Lógica necessária para registar um utilizador

API

Para tratar da integração de uma *API* num projeto da *Outsystems*, é necessário utilizar uma componente na parte de lógica desta plataforma, "consume REST API", onde permite duas opções: adicionar um método único ou adicionar vários métodos. Foi escolhida a opção de adicionar um único método, onde foi colocado o caminho para o *URL* da API e uma porção do formato JSON do pedido a essa mesma *API*. Desta forma é criado automaticamente o método "GetVolumes" sendo também adicionado à zona de dados a entidade de livros e respetivos atributos.

Para que a página de lista de livros fosse feita, foi primeiramente adicionado ao método anteriormente conseguido um paramento de entrada "bookSearch", alterando o caminho de *URL* para a *API* para receber um termo de procura, em semelhança ao que foi necessário fazer no *Mendix* (ver Figura 25).

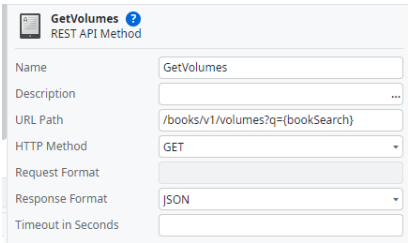


Figura 25: Caminho url para pedido de termo específico à API

No ecrã de lista de livros, foi adicionada a variável local "bookSearch" do tipo *Text* com o valor

default de “harry” utilizada para guardar o texto procurado pelo utilizador na caixa de procura. Foi ainda adicionada a variável “bookSearchData”, do tipo *Books_volume*, destinada a guardar os dados recolhidos do *request* à API dos livros resultantes da procura feita pelo utilizador (ou seja, tendo em conta o que ele escreveu no motor de busca).

Seguidamente, para que estas variáveis sejam utilizadas e para que as ações sejam efetuadas é necessário fazer essa lógica nos fluxos. Assim, é criado o fluxo, em que é feito o *request* à *API* tendo em conta o termo escolhido, guardando os dados na variável “bookSearchData”. Este fluxo é feito sempre que o utilizador procura por um termo diferente. No ecrã de lista de livros, cada vez que este fluxo é executado, são retirados o título e *thumbnail* de cada livro dos dados resultantes (ver Figuras 26 e 27).

Figura 26: Lógica necessária para fazer *request* à API com um termo específico

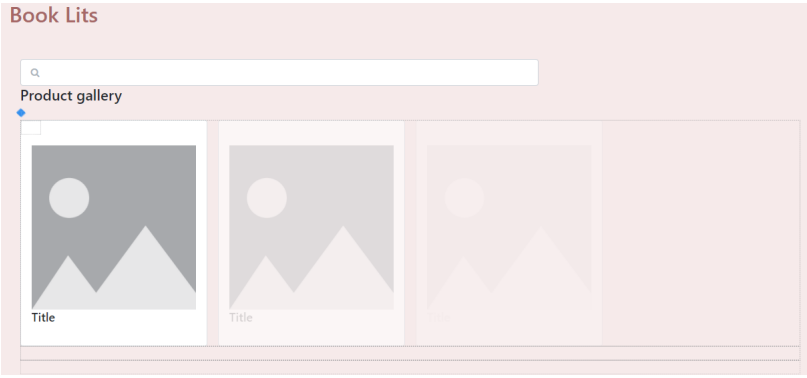


Figura 27: Interface do ecrã lista de livros

É ainda realizado mais um fluxo, em que ao serem selecionados tanto o título como a *thumbnail* de um livro, o utilizador é redirecionado pata a página de detalhes de livro. Ao fazer este redirecionamento, é também enviado o id corrente, ou seja, o id do livro selecionado, esta possibilidade é útil, pois permite guardar o id do livro que será passado como termo de procura na página lista de detalhes, permitindo detalhar um livro específico (ver Figura 28).

Figura 28: Lógica necessária para redirecionar o utilizador para a página de detalhes do livro passando o id do livro específico

Para a que o ecrã de detalhes de livro seja realizada, é necessário primeiro criar um método "Get-Volume", recebendo como *input* um id de livro, mudando o caminho de *URL* para a API dando como paramento o id do livro (ver Figura 29).

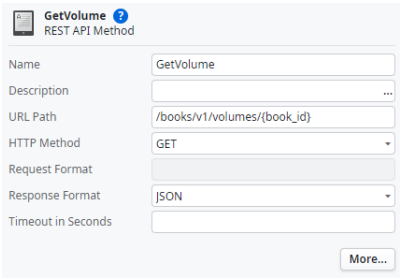


Figura 29: Método necessário para fazer *request* à API de um livro específico tendo em conta um id

A página de detalhes de livro recebe como parâmetro de entrada, o id passado no fluxo anterior. Recebendo este parâmetro de entrada, é feito um fluxo (*Data Action*) que devolve dados da entidade livros, onde o *request* é feito dando o id do livro pretendido (ver Figura 30).

Figura 30: Lógica necessária para retirar dados de um livro específico

3.2.3 MS PowerApps

Autenticação

Para realizar a autenticação foi necessário encontrar uma forma de guardar os dados dos utilizadores para serem usados sempre que necessário. Assim, foi adicionado à zona dos dados um *Excel*, onde foram adicionados uma coluna para cada atributo necessário, neste caso os atributos criados foram *username*, *password* e *email*. Para o armazenamento de dados dos utilizadores a *PowerApss* oferece várias outras opções, como por exemplo *SQL Server*, que seria um benefício para a integração de dados na aplicação, no entanto, esta opção é paga na *PowerApps*, pelo que não foi escolhida. Para que a lógica das páginas de registo e *login* funcionasse, ao contrário das plataformas estudadas anteriormente, não exigiu nenhuma utilização de fluxos, apenas sendo utilizadas fórmulas da *PowerApss* (ver Figuras 31, 32 e 33).

```
If(TextInput4.Text in Users.Username && TextInput4_1.Text in Users.Password;

Navigate(ListaLivros; Cover);


Notify("Account does't exist!";Error; 5000)&Reset(TextInput4)&Reset(TextInput4_1))
```

Figura 31: Exemplo de fórmulas utilizadas na *MS PowerApss*

Após colocar os elementos destas páginas funcionais, tratou-se da customização destas.

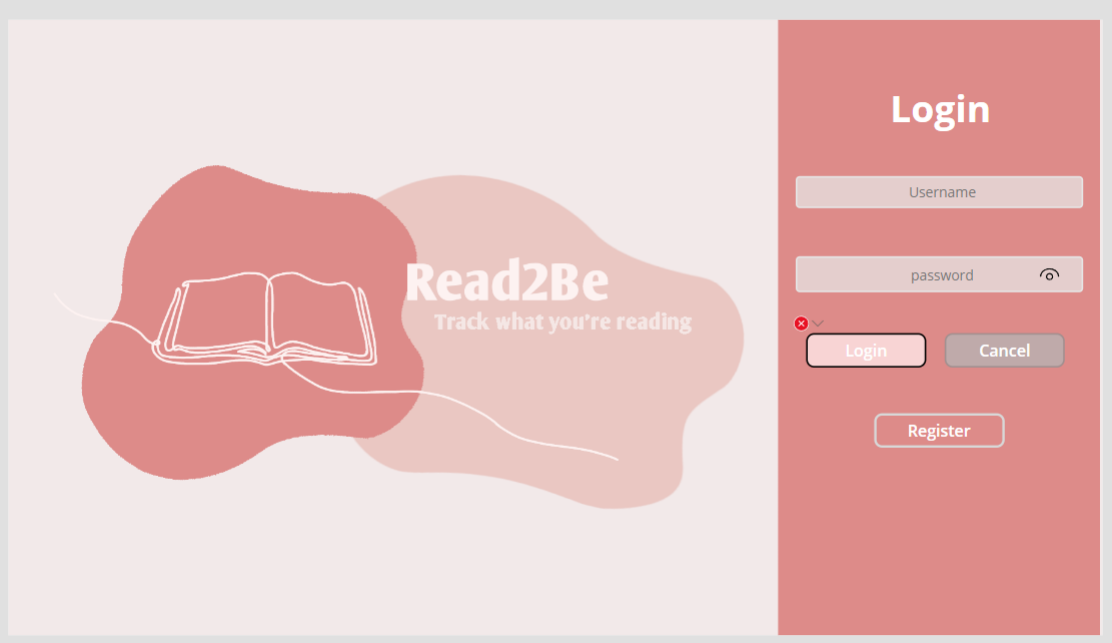


Figura 32: Ecrã de *Login*

Figura 33: Ecrã de Registo

API

Os recursos para fazer o consumo de uma API no *MS PowerApps* são pagos, pelo que para esta plataforma não foi possível realizar esta tarefa. No entanto, como substituição foram implementadas as páginas listagem de livros e de detalhes de livro com dados obtidos a partir de um *dataset*. Como a *PowerApps* não aceita *datasets* com valores em branco, foi ainda necessário fazer um tratamento de dados. Assim, todas as entradas com valores em branco do *dataset* foram eliminadas através de um *script* em *Python*. O *dataset* resultante foi então adicionado aos dados da plataforma.

Após a problemática dos dados estar tratada, apenas é necessário usar novamente as fórmulas do *MS PowerApps* para procurar por listas de livros tendo em conta o título procurado pelo utilizador e ao selecionar algum livro especifico redirecionar o utilizador para a página lista de livros onde se encontram todos os detalhes do livro selecionado. Sendo ainda necessário tratar da customização de ambos os ecrãs (ver Figuras 34 e 35).

Figura 34: Ecrã de Lista de Livros

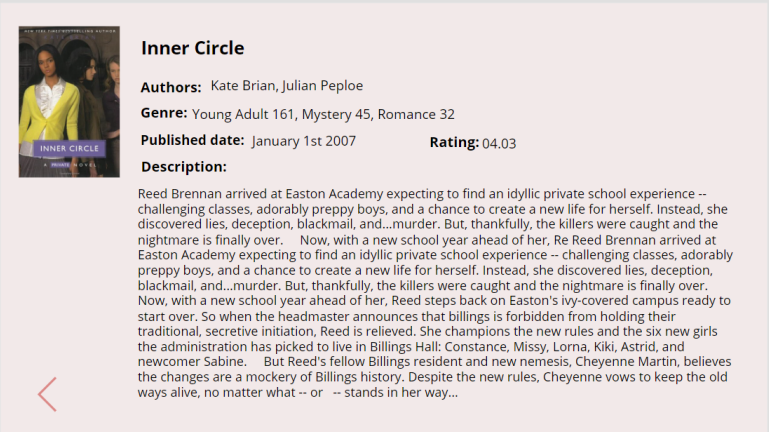


Figura 35: Ecrã de Detalhes de Livros

3.3 Dificuldades sentidas ao longo do desenvolvimento

Ao longo do desenvolvimento das aplicações e tendo em conta a experiência vivida, considero que, de uma forma geral, o desenvolvimento da *MS PowerApps* revelou ser mais fácil. Esta plataforma apresentou um forte apoio *online*, sendo relativamente fácil perceber as fórmulas a aplicar para as tarefas a desenvolver, para além disso, foi encontrado bastante apoio *online* para o desenvolvimento de tarefas semelhantes. Desta forma, ao visualizar vídeos de tarefas semelhantes e ao estudar as fórmulas, foi possível chegar a um resultado positivo. No entanto, esta plataforma apresentou algumas limitações quando comparadas as restantes plataformas. Por exemplo, como referido anteriormente, nesta plataforma foi utilizado um *Excel*, uma vez que as restantes opções mais avançadas eram pagas, para armazenamento dos dados dos utilizadores, o que pode não ser tão eficiente para a gestão de dados como a utilização de bases de dados relacionais e não relacionais como utilizado nas restantes plataformas. Para além disso, como

referido anteriormente, não foi possível a recorrer à *API* de livros da *Google*, uma vez ser também uma funcionalidade paga nesta plataforma. Para além disso, considero que o sistema de avisos de erros das fórmulas desta plataforma é mais incompleto quando comparado com as restantes plataformas.

No caso da utilização da *Outsystems* para desenvolvimento das tarefas pretendidas, encontrei algumas dificuldades, pois considero a documentação um pouco incompleta e sem exemplos práticos, para perceber melhor a sua utilização. Para além disso, principalmente para o desenvolvimento da segunda tarefa, deparei-me com a dificuldade de encontrar informação e vídeos que fossem de encontro à tarefa a desenvolver, pelo que levou muito tempo a entender profundamente os conhecimentos necessário para aplicar nas tarefas em específico. Para além disso, ao colocar os elementos necessários no local da interface, foi inicialmente um pouco confuso e limitante, uma vez que, é necessário colocar os elementos em locais específicos, colocando-os sempre num contentor. No entanto, a introdução de base de dados nesta plataforma é algo relativamente simples, sendo a sua gestão tratada automaticamente. Por último, relativamente à primeira tarefa, já existia um fluxo simples pré-definido para o login, o que simplificou bastante a tarefa.

Finalmente, no *Mendix*, o desenvolvimento das tarefas demonstraram ser de especial dificuldade. A introdução de base de dados é um pouco mais complexa que a nas restantes plataformas, sendo necessário definir as tabelas e colocar as ligações entre estas. Para além disso, a documentação é um pouco confusa não sendo uma ajuda fundamental para iniciantes nesta plataforma, no entanto, como é uma plataforma muito utilizada foram encontrados muitos vídeos explicativos para iniciantes. Mais uma vez, como não foi possível encontrar informações *online* que ajudassem diretamente no desenvolvimento da aplicação, foi necessário apreender um grande conhecimento, antes de desenvolver a aplicação. Além disso, esta plataforma apresenta ainda muitas limitações no que se refere à customização da aplicação, sem recorrer a *css*, pelo impossibilitou uma customização da interface tão completa como pretendido.

Para além disso, a aprendizagem de uma plataformas não permite transferir conhecimentos que são utilizados nas restantes plataformas, o que consistiu também numa dificuldade.

3.4 Conclusões de utilização das plataformas

A seguir são apresentados os resultados obtidos do estudo das plataformas descrito anteriormente. Na Tabela 1, podemos observar o tempo, em horas, necessário à execução de cada tarefa.

Tarefa/Plataforma	Outsystems	Mendix	MS PowerApps
Autenticação	3	6	4
Páginas de Livros	5	8	3

Tabela 1: Tabela de tempo utilizado na execução das tarefas propostas

De uma forma geral, a plataforma *Mendix* foi a mais demorada para a execução da aplicação e a *MS PowerApps* demonstrou ser a de mais rápida execução. No entanto, a análise da rapidez associada à *PowerApps* pode estar enviesada, uma vez que, a tarefa relativa ao uso da *API* de livros da *Google* não foi realizada, mas substituída pelo uso de um *dataset* de livros.

3.4.1 Características analisadas

Nas seguintes Tabelas 2 e 3 sumariei algumas características de cada plataforma. A primeira tabela apresenta uma descrição para cada uma das características, sendo estas analisadas e assinaladas, na segunda tabela, para as plataformas em que estas características se verificam.

Característica	Descrição
Interface Gráfica	
Arrastar e largar	Arrastar elementos envolvidos na criação de uma aplicação
Abordagem de apontar e clicar	Semelhante ao anterior mas em vez de arrastar os elementos e os largar envolve clicar no item
Suporte de interoperabilidade	
Interoperabilidade com serviços externos	Possibilidade para incorporação de diferentes serviços e plataformas como Microsoft, Google, etc. Permitindo ainda a incorporação de diferentes plataformas low-code
Conexão com fonte de dados	Conecta a aplicação com base de dados relacionais e não relacionais
Suporte de segurança	
Segurança da aplicação	Mecanismo de segurança de uma aplicação que envolve confidencialidade, integridade e disponibilidade de uma aplicação quando necessário
Segurança da plataforma	Gerenciamento da segurança para que a característica anterior seja assegurada
Suporte ao desenvolvimento colaborativo	

Table 2 continuação da página anterior

Característica	Descrição
Colaboração Offline	Diferentes desenvolvedores podem trabalhar ao mesmo tempo no mesmo projeto. Podendo trabalhar offline localmente dando mais tarde commit dessas alteração no servidor remoto que precisam de ser incorporadas corretamente
Colaboração Online	Diferentes desenvolvedores simultaneamente no mesmo projeto. Os conflitos são tratados durante a execução
Suporte à reutilização	
Formulários pré-construídos	Formulários editáveis reutilizáveis que o utilizador pode empregar no desenvolvimento da sua aplicação
Fluxos de trabalho integrados	Fluxos de trabalho reutilizáveis mais comuns a empregar na criação de uma aplicação
Dashboards pré-construídos	Painéis de controlo pré construídos que podem ser utilizados pelo utilizador no desenvolvimento da aplicação
Suporte ao Deployment	
Deployment na cloud	Permite que a aplicação seja implementado online numa infraestrutura cloud quando esta estiver pronta a ser usada
Deployment em infraestruturas locais	Permite que a aplicação seja implementado numa infraestrutura local do utilizador quando esta estiver pronta a ser usada
Tipos de aplicações suportadas	
Minotoramento de eventos	Permite recolha de dados, análise do evento decorrente desses mesmos dados e sinalização de eventos ocorrentes nos dados
Controlo de fonte	
Backup	Possibilidade de guardar uma cópia da aplicação desenvolvida até ao momento
Controle de alterações	Posiibilidade de voltar a qualquer versão de desenvolvimento da aplicação anterior
Sistema automático de aviso de erros	Quando um erro ocorre o utilizador é informado instantaneamente
Controle de conflitos	Quando duas versões são publicadas e existe um conflito entre ambas é dado um aviso e é acionado um mecanismo de suporte à resolução deste mesmo conflito, não podendo ser publicada a versão da aplicação sem estar resolvido este problema
Sistema de aviso de dependências	Caso haja alguma dependência necessária para o funcionamento da aplicação a desenvolver o utilizador é avisado
Controlo de acesso e permissões	Controlo de utilizadores que trabalham na mesma aplicação, sendo necessário aprovação por algum constituinte especificado

Table 2 continuação da página anterior

Característica	Descrição
Mecanismos de teste e deployment automático	Mecanismos que permitem a testagem e deployment da aplicação desenvolvida
Target	
Desktop	Possibilidade de desenvolver aplicações desktop
Web	Possibilidade de desenvolver aplicações web
Mobile	Possibilidade de desenvolver aplicações mobile
Controlo de Versões	
Histórico	Possibilidade de análise de todas as modificações feitas à aplicação
Marcação de versões estáveis	Possibilidade de marcar uma versão específica da aplicação, podendo mais tarde o utilizador voltar a essa mesma versão
Sistema de comparação de versões e junção das diferenças	Permite que quando dois utilizadores trabalham ao mesmo tempo na mesma aplicação, após as modificações estarem feitas , automaticamente é feita uma comparação entre as diferentes versões sendo feita a junção de forma funcional
Suporte GitHub	Permite guardar o projeto ao longo no final do seu desenvolvimento no GitHub
Suporte SVN	Permite guardar o projeto ao longo no final do seu desenvolvimento utilizando SVN
Branches	Permite dividir o projeto em branches de forma a que vários utilizadores possam trabalhar nas partes pretendidas
Customização	
Guardar templates	Possibilidade por parte do utilizador de guardar conjuntos de elementos já utilizados, guardando o seu funcionamento e customização
Contem CSS	Possibilidade por parte do utilizador de customizar detalhadamente a interface da aplicação utilizando código em vez das opções disponibilizadas da plataforma
Possibilidade ampla de customização da interface sem recorrer a css	Possibilidade por parte do utilizador de customizar detalhadamente a interface da aplicação sem necessidade de recorrer a código, utilizando apenas as opções pré-definidas da plataforma
API	
Url encoding automatico	Codificar automaticamente strings de input para serem compatíveis com o encoding de um URL
Outros	

Table 2 continuação da página anterior

Característica	Descrição
Marketplace	Possibilidade de no desenvolvimento da aplicação acrescentar ferramentas disponíveis num mercado da plataforma que facilitam o desenvolvimento pretendido
Debugger	Ferramenta de auxilio ao desenvolvedor com mecanismos de controlo do fluxo da aplicação a fim de corrigir algum comportamento irregular
Organização em modelo mvc	Organização de elemntos da plataforma sob a forma de Model-View-Controller
Integração interna de base de dados	Possibilidade de criação de todas as entidades necessárias para a criação, uso e controlo de dados sem necessidade de recorrer a ferramentas externas

Tabela 2: Explicação das características concluídas

Característica	OutSystems	Mendix	MS PowerApps
Interface Gráfica			
Arrastar e largar	√	√	√
Abordagem de apontar e clicar	√	√	√
Suporte de interoperabilidade			
Interoperabilidade com serviços externos	√	√	√
Conexão com fonte de dados	√	√	√
Suporte de segurança			
Segurança da aplicação	√	√	√
Segurança da plataforma	√	√	√
Suporte ao desenvolvimento colaborativo			
Colaboração Offline	√	√	√
Colaboração Online	√	√	
Suporte à reutilização			
Formulários pré-construídos	√	√	√
Fluxos de trabalho integrados	√		
Dashboards pré-construídos	√	√	√
Suporte ao Deployment			

Table 3 continued from previous page

Característica	OutSystems	Mendix	MS PowerApps
Deployment na cloud	√	√	√
Deployment em infraestruturas locais	√	√	
Tipos de aplicações suportadas			
Minotoramento de eventos	√	√	√
Controlo de fonte			
Backup	√	√	√
Controle de alterações	√	√	
Sistema automático de aviso de erros	√	√	√
Controle de conflitos	√	√	√
Sistema de aviso de dependências	√	√	√
Controlo de acesso e permissões	√	√	√
Mecanismos de teste e deployment automático	√	√	√
Target			
Desktop		√	
Web	√	√	√
Mobile	√	√	√
Controlo de Versões			
Histórico	√	√	√
Marcação de versões estáveis	√	√	
Sistema de comparação de versões e junção das diferenças	√	√	
Suporte GitHub	√	√	√
Suporte SVN	√	√	
Branches	√	√	√
Customização			
Guardar templates	√	√	√
Contem CSS	√	√	√

Table 3 continued from previous page

Característica	OutSystems	Mendix	MS PowerApps
Possibilidade ampla de customização da interface sem recorrer a css	√		√
API			
Url encoding automatico	√		N/A
Outros			
Marketplace	√	√	
Debugger	√	√	√
Organização em modelo mvc	√	√	√
Integração interna de base de dados	√	√	

Tabela 3: Características concluídas

3.4.2 Limitações e vantagens encontradas

Após comparadas as características destas plataformas, foram ainda retiradas algumas limitações e vantagens sobre cada uma das plataformas estudadas.

Relativamente à *MS PowerApps*, foram notadas algumas limitações, tais como, os problemas frequentes na conexão com o *Excel*, o desempenho mais fraco quando comparada com as restantes plataformas e os erros desconhecidos que desaparecem após o reiniciar da aplicação. Para além disso, não guarda automaticamente o trabalho desenvolvido, resultando por vezes na perda de todo o processo de implementação. Por fim, algumas das funcionalidades importantes para o desenvolvimento de uma aplicação completa são pagas. No entanto, esta plataforma também apresenta aspetos positivos como, o suporte *online* satisfatório, não sendo necessário descarregar a *app* nativa para a sua utilização e a sua interface de fácil compreensão.

No *Mendix* as limitações sentidas são a fraca customização (sem recorrer a *css*) e a interface da plataforma ser mais confusa que as restantes. Além disso, a inclusão de base de dados e o processo do consumo da *REST API* revelou-se mais complexo quando comparada com a *Outsystems*. As vantagens apontadas para esta plataforma são o facto de grande parte das funcionalidades serem gratuitas, pelo menos para o tipo de desenvolvimento deste estudo, e guardar automaticamente o trabalho desenvolvido. Adicionalmente, na ocorrência de um erro na aplicação, esta plataforma possibilita que o utilizador seja diretamente redirecionado para o local onde este ocorre.

Na *Outsystems*, as limitações observadas são a fraca documentação disponível, quando comparada com as restantes plataformas. Foi igualmente notado que a versão *web* da plataforma apresenta algumas limitações pelo que obriga à utilização da versão *desktop*. Além disso, contrariamente ao *Mendix*, nesta plataforma o processo de criação de uma aplicação é mais rígido, exigindo aos utilizadores que escolham logo de início o tipo de aplicação a criar, em vez de se poder fazer essa escolha ao longo do desenvolvimento. As principais qualidades notadas nesta plataforma são, o facto de grande parte das funcionalidades serem gratuitas, pelo menos para o tipo de desenvolvimento deste estudo e guardar automaticamente o trabalho desenvolvido. Adicionalmente, possibilita a inclusão de dados com facilidade, apresenta uma grande facilidade no consumo de uma *REST API* e na ocorrência de um erro na aplicação, possibilita que o utilizador seja diretamente redirecionado para a zona de erro.

Capítulo 4

Estudo empírico com utilizadores

Visando perceber a usabilidade das três plataformas selecionadas e as suas características, positivas e negativas, foi realizado um estudo empírico, com vários participantes de várias universidades. Neste capítulo este estudo será descrito.

4.1 Desenho do estudo

De forma a conseguir avaliar a usabilidade destas plataformas com outros utilizadores, procuramos encontrar um conjunto de pessoas disponíveis para desenvolver parte da aplicação que descrevemos no capítulo 3. Naturalmente, procuramos participantes no meio académico. Uma vez que queríamos *feedback* por parte de utilizadores com e sem experiência em programação, parte destes participantes teriam de pertencer a cursos de informática, o que corresponde, a participantes com alguma experiência no desenvolvimento de aplicações ou com conhecimento em certas linguagens de programação. Os restantes participantes teriam de integrar cursos não relacionados com informática, o que corresponde, a participantes sem experiência no desenvolvimento de aplicações e sem conhecimentos em linguagens de programação. A cada participante foi atribuído uma das tarefas referidas no capítulo anterior para uma das plataformas, sendo dado a cada participante duas horas para realização da tarefa. Não é crucial que todos os participantes consigam acabar as suas tarefas neste tempo, uma vez que, o esperado é que estes percebam o funcionamento da plataforma atribuída, de forma a conseguirem formar uma opinião acerca desta. Para finalizar o estudo, é apresentado a cada participante um questionário onde será avaliado a sua experiência com a plataforma.

Para auxiliar a melhor compreensão em cada uma das plataformas e tarefas, foi disponibilizado um guião[1], onde as tarefas foram explicadas, sendo ainda apresentado o objetivo do estudo, apresentando

[1] O guião foi acedido através deste *link*: https://docs.google.com/document/d/e/2PACX-1vSayLnKsFPSEOc-nDjy-PB_NyVSmpC9BckDRnoSRaPbpJ_li48zn4nF9jJY2ZOPVSmAvYpUpjwEqvHd/pub

também todos os links de instalação das plataformas, da documentação e ainda de algumas páginas úteis para iniciantes nestas plataformas. Para além do guião, cada participante teve acesso a um vídeo[2] onde a tarefa é desenvolvida. Este vídeo apenas mostra de forma geral os passos necessários a fazer para realizar a tarefa, não explicando pormenorizadamente todos os passos para que os participantes possam também descobrir mais da plataforma pelos seus próprios meios, por exemplo, recorrendo à documentação da plataforma que será também uma questão mencionada no questionário.

O questionário[3] é dividido em duas partes. A primeira parte tem o intuito de saber algumas informações sobre o participante, como o sexo, o curso, se já utilizou alguma plataforma *low-code* anteriormente. A segunda parte do questionário é constituída por perguntas comuns a todos os participantes e perguntas específicas tendo em conta a tarefa atribuída. As perguntas comuns a todas as tarefas consistem em perceber se o utilizador conseguiu acabar a tarefa que lhe foi atribuída, avaliar as plataformas em termos de facilidade de utilização, facilidade na criação de novos ecrãs, facilidade na criação da lógica dos eventos para os ecrãs desenvolvidos, facilidade de customização das aplicações desenvolvidas e utilidade da documentação. Questionaram-se também estes alunos se, na sua opinião, existe uma relação entre a divulgação destas plataformas e a sua utilização e, ainda, se consideram utilizar esta plataforma *low-code* novamente no futuro. Relativamente às perguntas específicas, para as tarefas relativas à utilização da *REST API* (*Mendix* e *Outsystems*), os participantes foram questionados sobre a facilidade de uso da *REST API* e facilidade de utilização dos dados obtidos a partir da *REST API*. Para a tarefa relativa à autenticação presente nas três plataformas, o utilizador apenas é questionado sobre a facilidade de validação, criação e utilização dos dados relativos aos utilizadores da aplicação criada.

Para a realização deste estudo, todos os participantes foram obtidos por convite direto, sendo todos amigos, familiares e conhecidos. O estudo foi realizado presencialmente ou por chamada online (onde o ecrã podia ser partilhado, podendo acompanhar todo o progresso de desenvolvimento), consoante preferência dos participantes.

4.2 Variávies

As variáveis independentes são: A plataforma, a tarefa atribuída e o grau de dificuldade atribuído a cada etapa necessária para o desenvolvimento da tarefa atribuída.

[2] Os links de acesso aos vídeos estão disponíveis os seguintes *links*, estando pela ordem de plataformas de *Mendix*, *MS PowerApps* e *Outsystems*: https://youtu.be/mkROO8pKJTk, https://youtu.be/BvygZOWBFDA, https://youtu.be/KtKzL6_faCU

[3] O questionário foi acedido através deste *link*: https://docs.google.com/forms/d/e/1FAIpQLSewlRsc7p8EucepbK-y39epLkktS5Dygo6RNYTTk_8NzqoGkQ/viewform

4.3 Sujeitos e Objetos

Como referido anteriormente, os participantes neste estudo pertenciam a vários meios de estudo (ver Figura 36), tendo estes diferentes níveis de experiência em informática, principalmente a nível do desenvolvimento de aplicações e de conhecimento em programação. Dos vinte e um participantes, treze do sexo masculino e oito do sexo feminino, sete dedicaram-se à plataforma *Mendix*, sete à *MS PowerApps* e sete à *Outsystems*, sendo distribuídos pelas três tarefas descritas no capítulo anterior (ver Tabela 4).

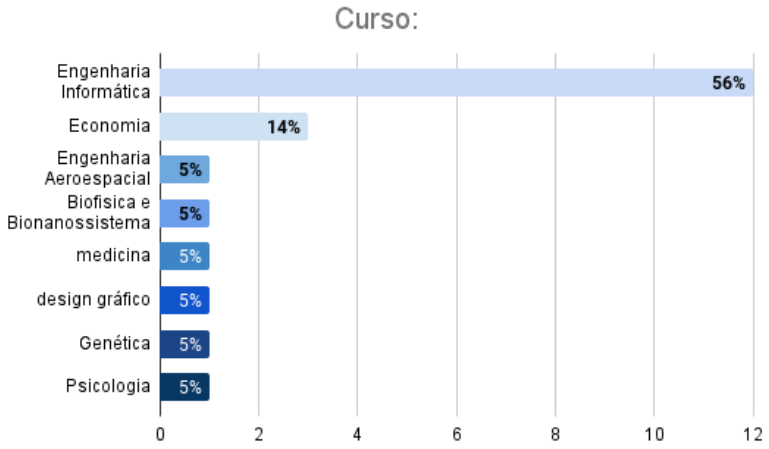


Figura 36: Contagem de participantes distribuídos por curso

Tarefa/Plataforma	Mendix	MS PowerApps	Outsystems
Autenticação	3	4	4
REST API	4	N/A	3
Uso de dataset para procura de livros	N/A	3	N/A

Tabela 4: Distribuição dos participantes pelas tarefas e plataformas

O objeto deste estudo é perceber de que forma os participantes se sentem face à utilização das plataformas *low-code* selecionadas, comparando a facilidade de utilização destes e inferindo quais das plataformas despertou mais interesse por parte dos participantes, concluindo ainda se estas plataformas são intuitivas quando apresentadas a indivíduos sem qualquer *background* em informática.

4.4 Instrumentação

Para a realização deste estudo os participantes precisam apenas de fazer *download* das plataformas *Mendix* e *Outsystems*, enquanto no caso da *MS PowerApps* podem usar a versão *web*. Para a utilização destas plataformas os participantes precisaram ainda de criar uma conta na plataforma atribuída. Para além destes requisitos, para os participantes cuja tarefa atribuída corresponde à utilização da *API* de livros da *Google* para o desenvolvimento das páginas de lista de livros e detalhes de livro, é ainda necessário estes conhecerem os caminhos *url*, tanto para o *request* relativo à recolha de vários livros como ao *request* relativo à recolha de informação de um livro específico tendo em conta o seu *id*, sendo estes dois caminhos *url* fornecidos aos participantes antes da experiência através do guião mencionado anteriormente. Por outro lado, para participantes cuja tarefa atribuída corresponde ao desenvolvimento das páginas de lista de livros e detalhes de livros na plataforma *MS PowerApps* deparam-se com a necessidade de aceder ao *dataset* com a listagem dos vários livros. O *link* de acesso a este dataset está também disponível no guião referido anteriormente no início da experiência. O questionário a que os participantes são submetidos foi elaborado através do *Google Forms*. Como referido anteriormente, este formulário apresenta algumas versões com conjuntos de perguntas diferentes dependendo da tarefa atribuída ao participante.

4.5 Procedimento de Análise

Como referido anteriormente, a cada participantes foi requerido que respondesse a um questionário no final da experiência. Este questionário apresenta algumas versões, dependendo da tarefa atribuída a cada participante. Assim, ao longo do questionário cada participantes precisou de referir:

- a plataforma atribuída;
- a tarefa atribuída;
- a dificuldade que sentiu em várias etapas no desenvolvimento da tarefa proposta.

Assim, tornou-se possível comparar a dificuldade sentida nas várias etapas necessárias ao desenvolvimento das tarefas, permitindo comparar esta dificuldade para cada plataforma em específico. Desta forma, para comparar a dificuldade de execução das várias etapas, nós vamos:

- calcular a dificuldade geral na execução das várias etapas de desenvolvimento da tarefa;

- calcular a dificuldade de execução das várias etapas organizado por plataforma, permitindo inferir em qual das plataformas nessa etapa se revelou dificuldade mais elevada.

As questões que permitiram inferir as várias dificuldades foram classificados entre 1 (muito fácil) a 5 (muito difícil).

Na tentativa de ajudar o participante a aprender o máximo possível sobre a plataforma e tarefa atribuídas, foram disponibilizados vários *links* que poderiam ajudar na compreensão das plataformas a iniciantes, sendo ainda fornecido um vídeo onde as tarefas eram explicadas de forma muito incompleta, na tentativa de impedir que o participante copiasse todos os passos explicados no vídeo. Para além disso, todos os participantes foram acompanhados durante todo o processo da experiência, sendo dado todo o apoio necessário, desde que não comprometesse o intuito da experiência.

4.6 Execução

Como mencionado anteriormente, este estudo foi realizado presencialmente para parte dos participantes e via chamada de voz para os restantes. A cada participante foi pedido que instalasse a plataforma atribuída antes da realização do estudo, no entanto, como alguns destes estudantes encontraram dificuldades na execução deste pedido, apenas a instalaram um pouco antes da experiência com supervisão e ajuda. Durante todo o processo da experiência, estive presente para a eventualidade de algum participante precisar de algum esclarecimento de dúvidas ou caso necessitasse de alguma ajuda.

No início da experiência a cada participante foi explicitado em que consistia do estudo e o que teriam de fazer. Seguidamente foi dado acesso ao guião onde poderiam encontrar algumas informações úteis para a realização da tarefa e foi ainda disponibilizado um vídeo gravado por mim, onde era demonstrado de forma sucinta a tarefa a desenvolver, para dar algum apoio inicial aos participantes. Após a visualização deste vídeo foi dado a cada participante um total de duas horas para tentarem fazer a tarefa pretendia, sendo de seguida submetidos ao questionário.

4.7 Ameaças à validação dos resultados

Como seria de esperar, ao longo da realização deste estudo enfrentámos alguns problemas que poderiam enviesar os resultados, no entanto, todos se resolveram para que desta experiência resultasse a melhor amostra de dados possível. Destes problemas, o maior está associado ao reduzido número de participantes, no entanto, apresentamos estudantes de dois grupos distintos (informática e outras áreas), o que

resultou numa maior variedade de dados. Para além disso, o facto de os participantes relativos à tarefa das páginas de livros para a plataforma *MS PowerApps* não utilizar a *API* para obter os dados, tornando a tarefa mais simples quando comparada com as outras plataformas, podia também ser visto como uma ameaça aos resultados destes mesmos participantes. No entanto, o intuito do estudo é conseguir avaliar a plataforma, acabando por não ser visto como um problema.

4.8 Dados obtidos

4.8.1 Dados gerais recolhidos no estudo

Dos vinte e um participantes, apenas dois conseguiram terminar a tarefa que lhes foi atribuída, ambos atribuídos a tarefas da plataforma *MS PoweApps*, no entanto, com tarefas distintas.Destes participantes, um deles era de informática e concluiu a tarefa relativa às páginas de listas de livros, enquanto o outro participantes era de um curso não relacionado com informática e terminou a tarefa relativa à autenticação. Para os restantes participantes os resultados foram os seguintes:

- Dos restantes participantes, no caso da *MS PowerApps*, relativamente à tarefa da autenticação, nenhum dos restantes estudantes conseguiu fazer ambos os ecrãs funcionais, sendo que um destes, pertencente a um curso de informática conseguiu fazer o necessário para a página de *registo*. Dos participantes cuja tarefa atribuída consta no desenvolvimento das páginas de livros, um dos estudantes de informática conseguiu listar os livros, não permitindo fazer a procura por títulos. Para além disso, nesta tarefa, um participante de uma área externa a informática conseguiu também fazer a listagem de livros, no entanto, não conseguindo colocar nada para além dos títulos nessa listagem.

- No caso do *Mendix*, dos participantes relativos às páginas de livros, apenas um, pertencente a informática, conseguiu terminar uma das páginas, relativa à lista de todos os livros disponíveis, não conseguindo fazer a procura por títulos de livros. Dos restantes estudantes destinados a esta tarefa, apenas um, também de informática, conseguiu fazer o consumo da *REST API*, não conseguindo aplicar os dados obtidos. Os restantes participantes, todos de cursos não relacionados com informática, não conseguiram chegar a nenhuma etapa implementada. Nenhum dos participantes desta plataforma, atribuídos à tarefa de autenticação conseguiu terminar a tarefa atribuída, no entanto, um dos estudantes de informática conseguiu terminar o fluxo necessário para a lógica da página de *login*, não conseguindo aplicá-la no ecrã de*login*.

- No caso da *Outsystems*, um dos participantes de informática conseguiu fazer uma página de registo funcional, sendo que os restantes ou não conseguiram testar a seu progresso para saber o quanto conseguiram implementar, ou não conseguiram terminar os fluxos necessários para nenhum dos eventos dos ecrãs pretendidos. Relativamente aos participantes destinados ao desenvolvimento das páginas de livros, nenhum conseguiu terminar nenhum dos ecrãs, no entanto, um dos participantes, pertencente à área de informática, conseguiu listar os livros, não conseguindo implementar a procura.

Assim, como seria de esperar, os participantes de informática conseguiram desenvolver mais etapas das tarefas atribuídas quando comparados com os restantes participantes.

Relativamente à experiência dos participantes no desenvolvimento de aplicações, verificou-se que, a percentagem de participantes que já tinha desenvolvido aplicações é igual à percentagem de participantes que nunca tinha desenvolvido uma aplicação (ver Figura 37).

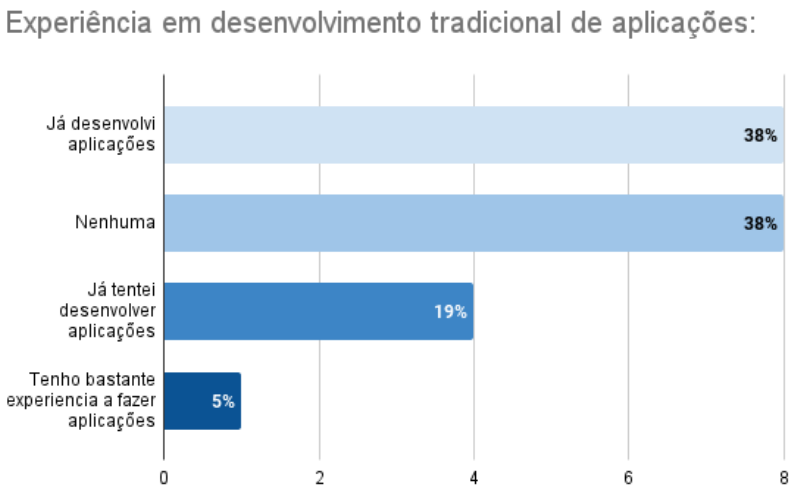


Figura 37: Contagem de participantes distribuídos por experiência no desenvolvimento de aplicações

Para além disso, foi avaliado o interesse dos participantes em utilizar este tipo de plataformas antes e depois da experiência, constatando que antes da experiência a maioria dos participantes não considerou a utilização de uma plataforma *low-code*. No entanto, após a experiência, foi notado um aumento do interesse dos participantes para a utilização destas plataformas para o desenvolvimento de aplicações. Este aumento demonstrou-se principalmente nos participantes de cursos de informática, enquanto nos participantes de outras áreas a alteração foi miníma (ver Figuras 38 e 39).

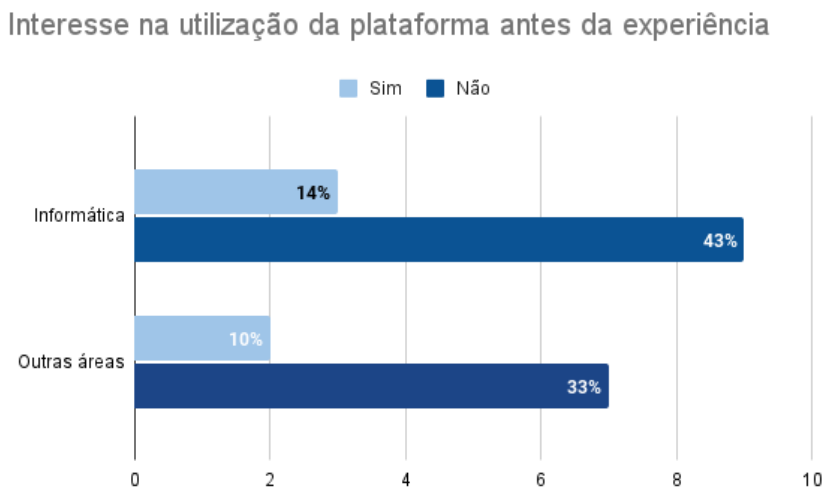


Figura 38: Contagem de participantes com e sem interesse na utilização de plataformas *low-code* antes desta experiência

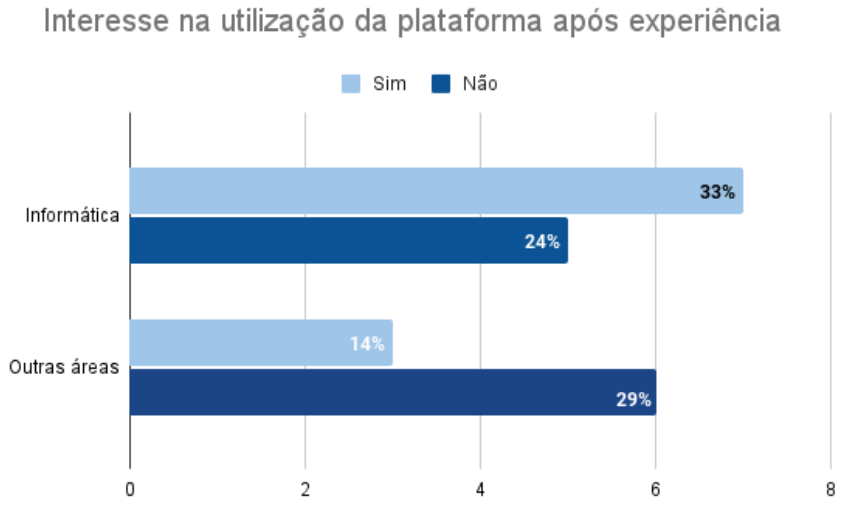


Figura 39: Contagem de participantes com e sem interesse na utilização de plataformas *low-code* após esta experiência

Podemos observar que, no caso dos participantes pertences a áreas não relacionadas com informática, o aumento de interesse para a utilização de plataformas *low-code* após a experiência ocorreu unicamente devido aos participantes da plataforma *MS PowerApps* (ver Figura 40).

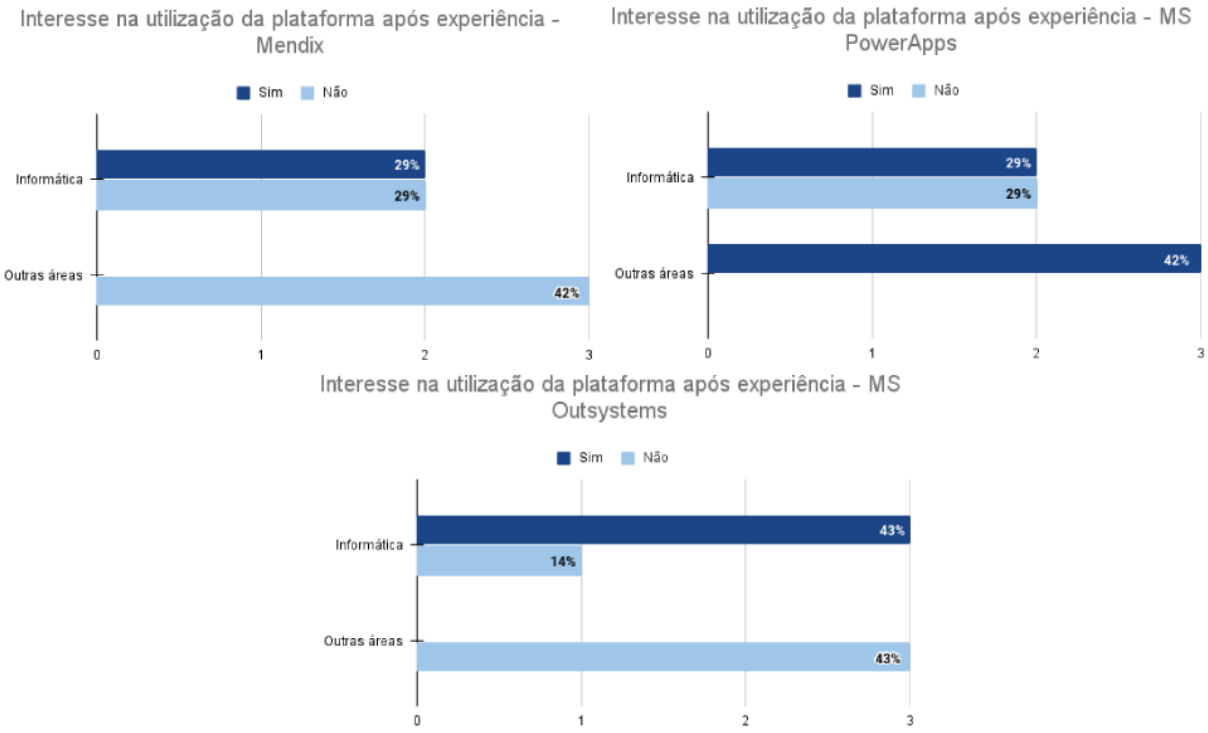


Figura 40: Contagem de participantes com e sem interesse na utilização de cada uma das plataformas *low-code* após esta experiência

De seguida, questionaram-se os participantes acerca da relação entre o aumento da divulgação de plataformas *low-code* e o aumento da sua utilização. A maioria dos participantes considera que o aumento da divulgação deste tipo de plataformas não aumentaria a sua utilização (ver Figura 41).

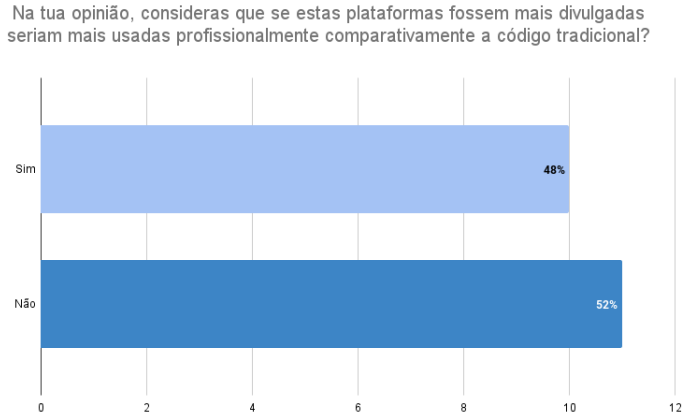


Figura 41: Contagem de participantes que consideram que caso as plataformas low-code fossem mais divulgadas passariam a ser mais utilizadas no mercado

Constatou-se que para as plataformas *MS PowerApps* e *Outsystems* existe uma relação positiva entre o aumento do uso de plataformas low-code e o aumento da divulgação destas (ver Figura 42).

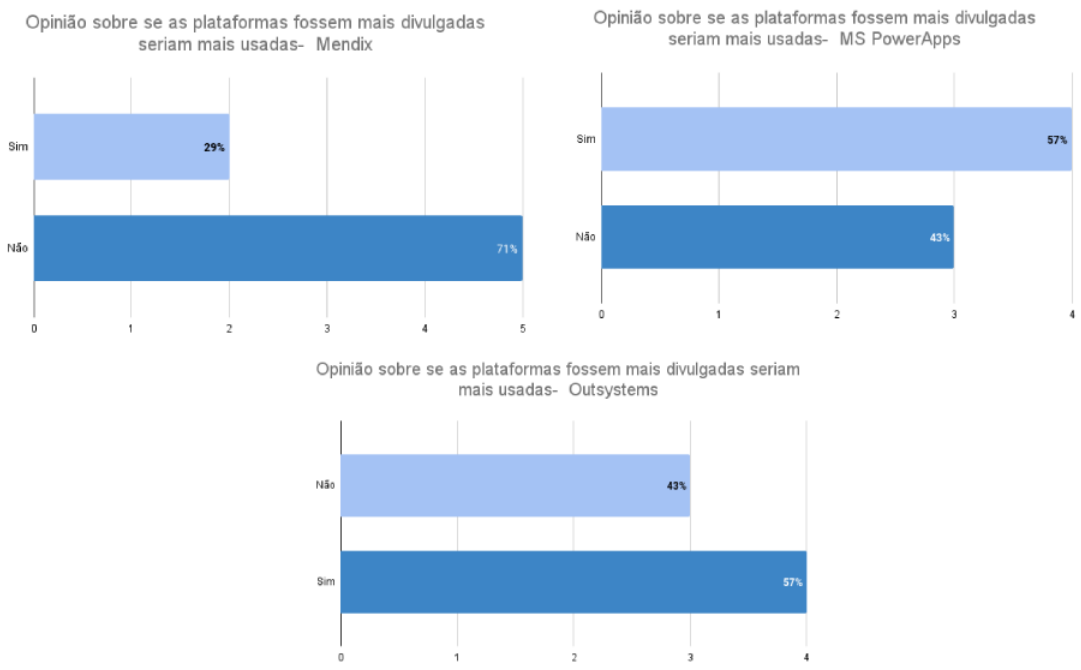


Figura 42: Contagem de participantes que consideram que caso as plataformas low-code fossem mais divulgadas passariam a ser mais utilizadas no mercado, tendo em conta a plataforma escolhida

Independentemente da tarefa realizada, cada participante foi também questionado acerca da facilidade de utilização da plataforma atribuída. A maioria dos participantes considerou que a sua plataforma era de difícil utilização, embora, tenha sido no *Mendix* onde esta dificuldade foi mais sentida. Além disso, é ainda importante realçar que a plataforma considerada de mais fácil utilização foi a *MS PowerApps* (ver Figuras 43 e 44).

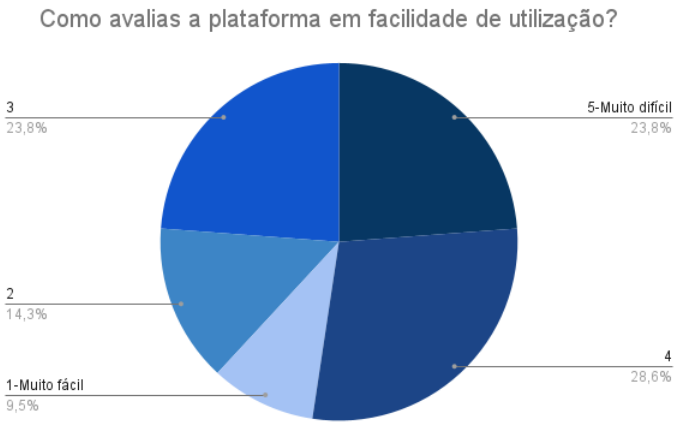


Figura 43: Distribuição de participantes por grau de facilidade de utilização da plataforma atribuída

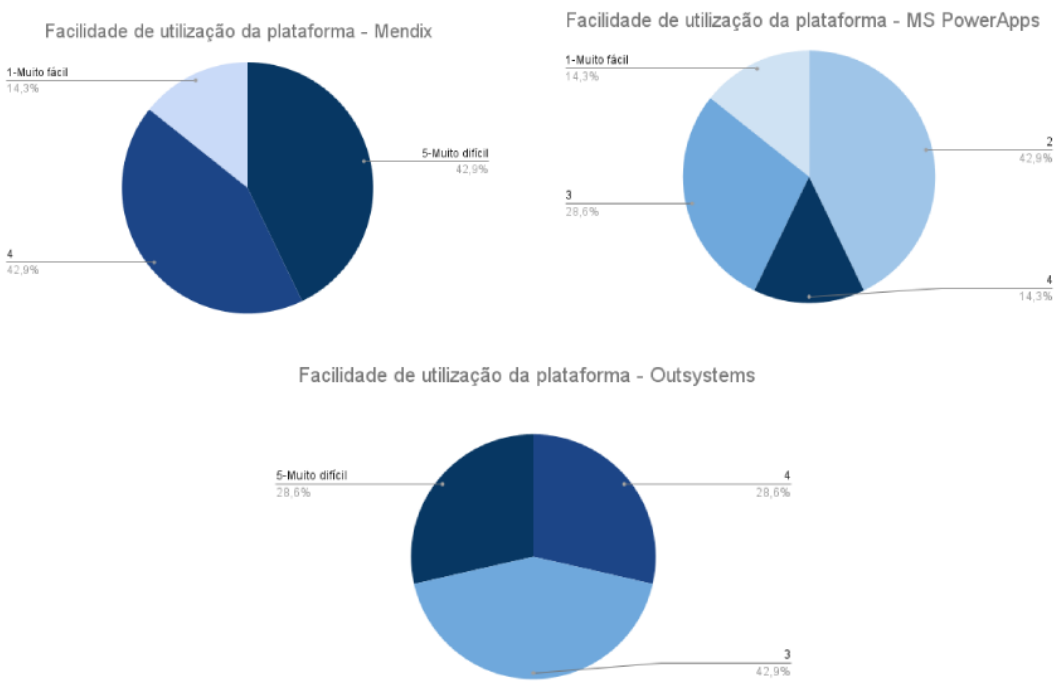


Figura 44: Distribuição de participantes por grau de facilidade de utilização da plataforma atribuída

Ao longo da realização da tarefa proposta, cada participante recorreu à documentação da plataforma utilizada, na tentativa de aprender alguns conhecimentos necessários para a realização da tarefa. Tendo em conta esta necessidade, cada participante foi questionado acerca da utilidade da documentação acedida (ver Figura 45).

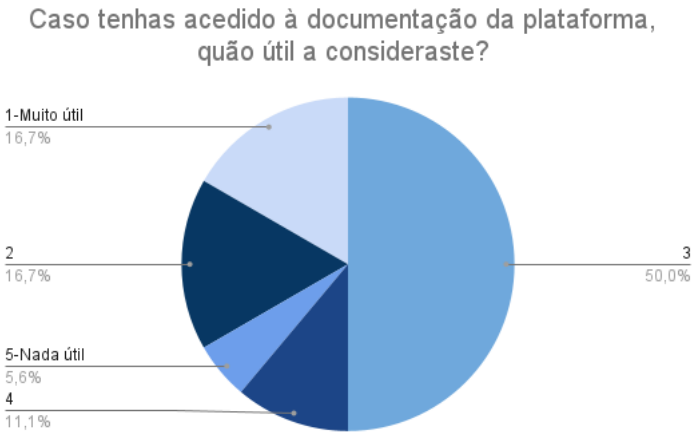


Figura 45: Distribuição de participantes por grau de utilidade da documentação da plataforma

Ao observar este fator organizado por plataforma atribuída, pode constata-se que os participantes que consideraram a documentação mais útil são aqueles cuja tarefa pertence à *MS PowerApps* (ver Figura

46).

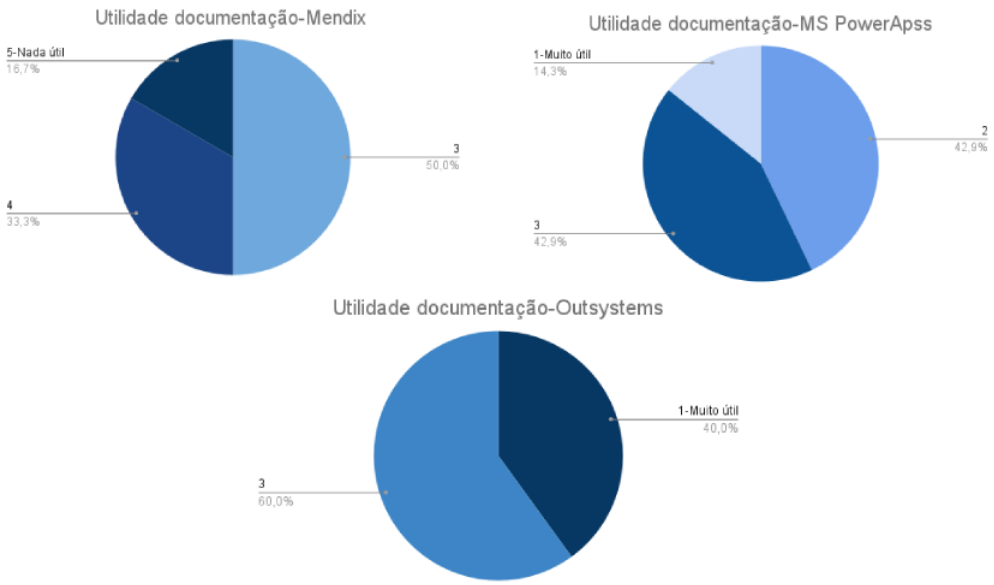


Figura 46: Distribuição de participantes por grau de utilidade da documentação da plataforma da tarefa atribuída

Para além da documentação, foi pedido a cada participante que incluísse também uma customização da interface da aplicação desenvolvida. Quando questionados acerca da facilidade de customização das aplicações criadas, as opiniões dos participantes mostraram-se dispersas, notando-se um número semelhante de participantes que consideraram esta uma tarefa fácil com os participantes que consideraram uma tarefa difícil (ver Figura 47).

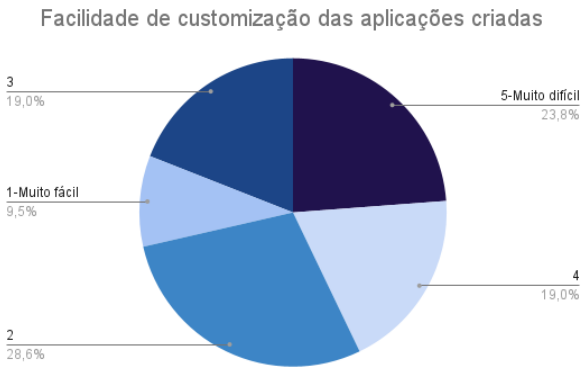


Figura 47: Distribuição de participantes por grau de facilidade de customização da aplicação desenvolvida

No entanto, ao analisar as respostas a esta questão, organizadas por plataforma utilizada, pôde

constatar-se que para utilizadores do *Mendix*, esta foi considerada uma tarefa especialmente difícil (ver Figura 48).

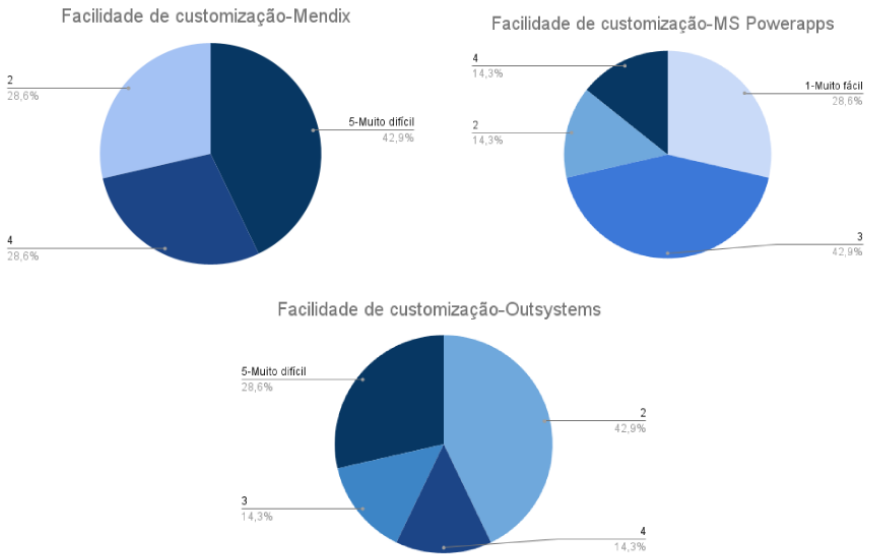


Figura 48: Distribuição de participantes por grau de facilidade de customização da aplicação desenvolvida para cada plataforma

4.8.2 Dados específicos recolhidos para cada tarefa

Como cada tarefa tem as suas dificuldades e necessidade de aprendizagem da plataforma em questão, para cada tarefa elaboraram-se questões que visam avaliar a plataforma atribuída tendo em conta a tarefa e facilidade no seu desempenho.

Autenticação

Como referido anteriormente, onze participantes ficaram atribuídos à tarefa de autenticação das aplicações criadas, sendo estes distribuídos pelas três plataformas *low-code* escolhidas.

Para desempenhar esta tarefa, é necessário introduzir e utilizar os dados relativos aos utilizadores da aplicação desenvolvida. Relativamente à facilidade desta tarefa, as respostas gerais desta questão foram muito dispersas, uma vez que, a percentagem de participantes que avaliaram a plataforma com dificuldade elevada foi semelhante à percentagem de participantes que a consideraram fácil (ver Figura 49).

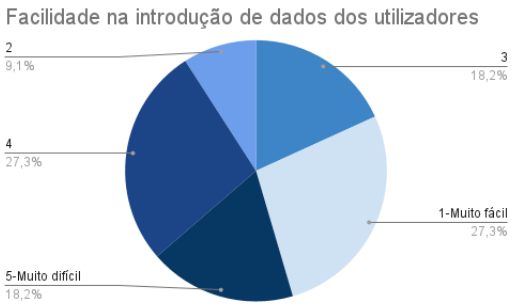


Figura 49: Distribuição de participantes por grau de facilidade de introdução de dados de utilizadores

Tal como os resultados gerais, também os resultados para a plataforma *Mendix* são inconclusivos. No entanto, para a plataforma *Outsystems*, os participantes revelaram facilidade na realização desta tarefa, enquanto os participantes da plataforma *MS PowerApps* revelaram ter dificuldade na execução desta tarefa (ver Figura 50).

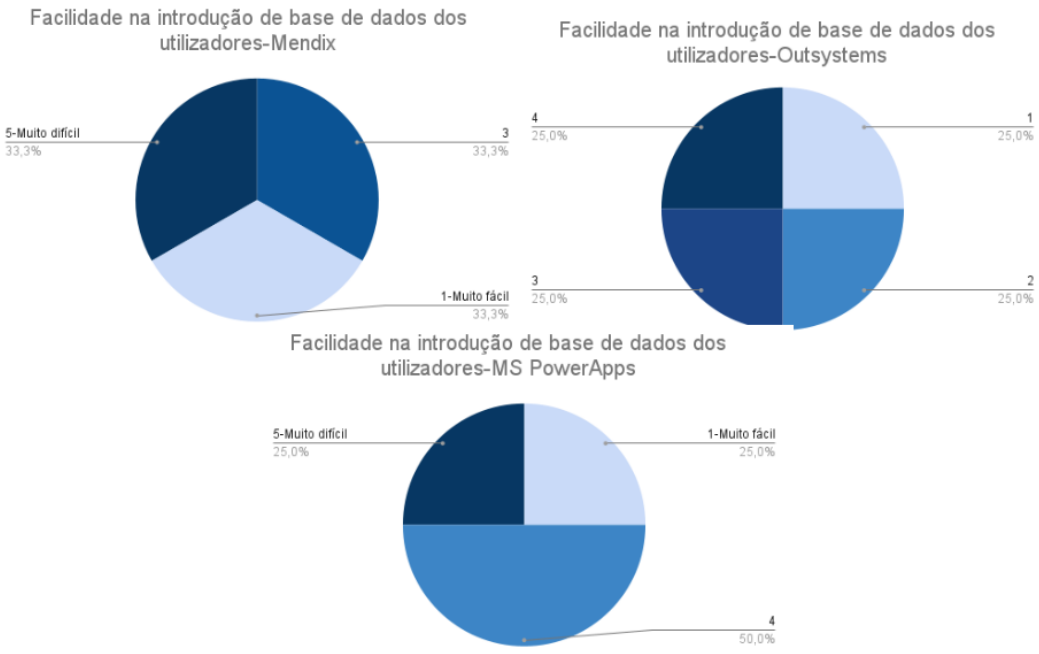


Figura 50: Contagem de participantes por grau de facilidade de introdução de dados de utilizadores para cada plataforma

Outra parte necessária para o desenvolvimento desta tarefa, é referente à lógica dos eventos das páginas de *login* e de registo, que permite tratar de todas as validações, necessárias para o funcionamento dos botões de registo e *login* (ver Figura 51). Os dados permitem inferir que a maioria dos alunos considerou esta tarefa dificil.

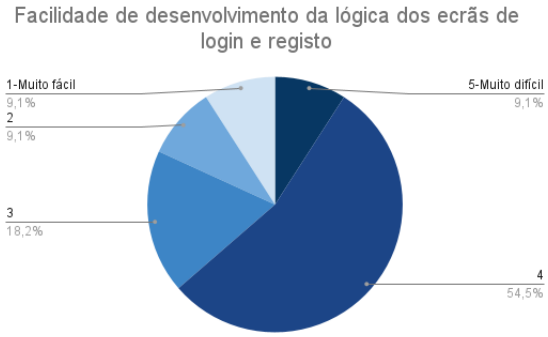


Figura 51: Distribuição de participantes por grau de facilidade de desenvolvimento da lógica necessária par os ecrãs de *login* e registo

Ao distribuir este fator por cada uma das plataformas, pode concluir-se que o desenvolvimento da lógica para os eventos dos ecrãs de *login* e registos foram avaliados com especial dificuldade para a plataforma *Mendix* (ver Figura 52).

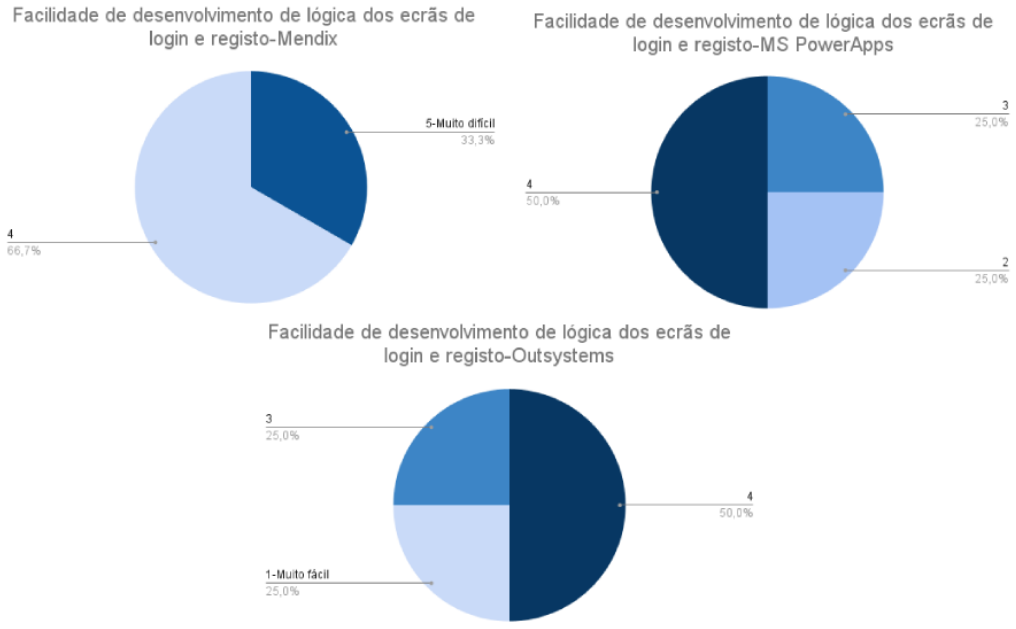


Figura 52: Distribuição de participantes por grau de facilidade de desenvolvimento da lógica necessária par os ecrãs de *login* e registo para cada uma das plataformas

Chamada à API

Para os participantes cuja tarefa atribuída consiste na utilização da *API* de livros da *Google* e desenvolvimento das páginas de lista de livros e detalhes de livros, também foram elaboradas questões específicas para a sua tarefa.

Assim, primeiramente estes foram questionados acerca da facilidade do consumo da *REST API*. A maioria dos participantes considerou esta tarefa de dificuldade elevada (ver Figura 53).

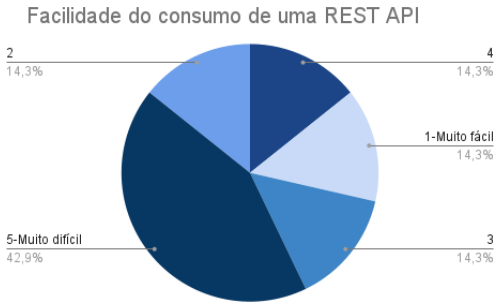


Figura 53: Distribuição de participantes por grau de facilidade de consumo da *REST API*

No entanto, ao observar a distribuição destes dados por ambas as plataformas utilizadas para esta tarefa, conclui-se que de uma forma geral, esta tarefa foi mais complexa para os participantes da *Outsystems* (ver Figura 54).

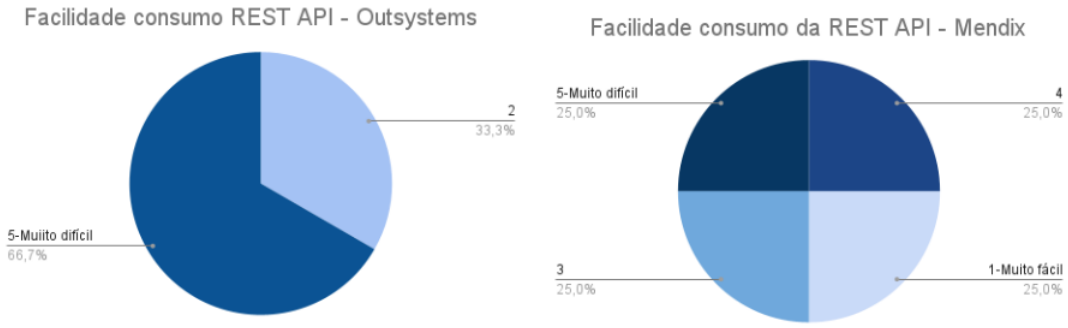


Figura 54: Distribuição de participantes por grau de facilidade de consumo da *REST API* para as plataformas destinadas a esta tarefa

Depois do consumo da *API* é necessário tratar de fazer o *resquest* à *API*, utilizar os dados resultantes e criar os ecrãs de livros e detalhes de livros e a respetiva lógica necessária por detrás dos eventos destes ecrãs.

Relativamente à utilização dos dados de livros retirados a partir ao *request* à *API* foi, de uma forma geral, avaliado como uma tarefa difícil segundo a experiência dos participantes (ver Figura 55).

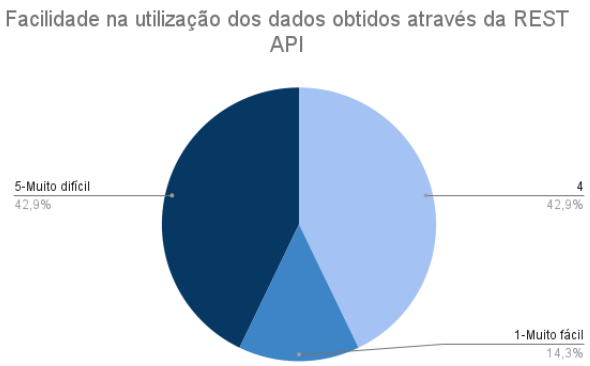


Figura 55: Distribuição de participantes por grau de facilidade na utilização dos dados retirados do pedido à *REST API*

Esta dificuldade foi notava nas duas plataformas usadas para a realização desta tarefa, embora na *Outsystems* se tenha notado que para todos os participantes esta foi uma tarefa de dificuldade elevada (ver Figura 56).

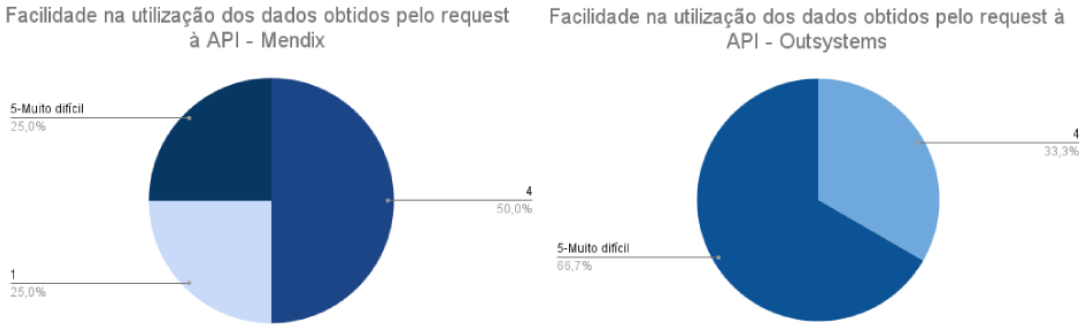


Figura 56: Distribuição de participantes por grau de facilidade na utilização dos dados retirados do pedido à *REST API* por plataforma

Para a criação da lógica necessária para os eventos referentes às páginas de lista de livros e detalhes de livro, pode concluir-se que esta foi considerada uma tarefa difícil para grande parte dos participantes. A dificuldade demonstrada no desenvolvimento desta tarefa foi notada em ambas as plataformas usadas para esta tarefa (ver Figuras 57 e 58).

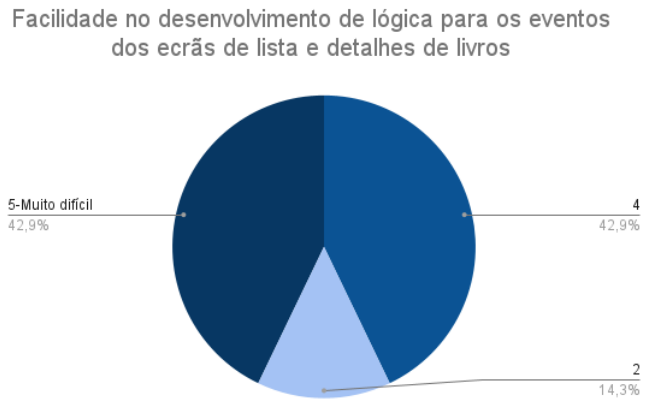


Figura 57: Distribuição de participantes por grau de facilidade na criação de eventos da página de lista de livros e detalhes de livros

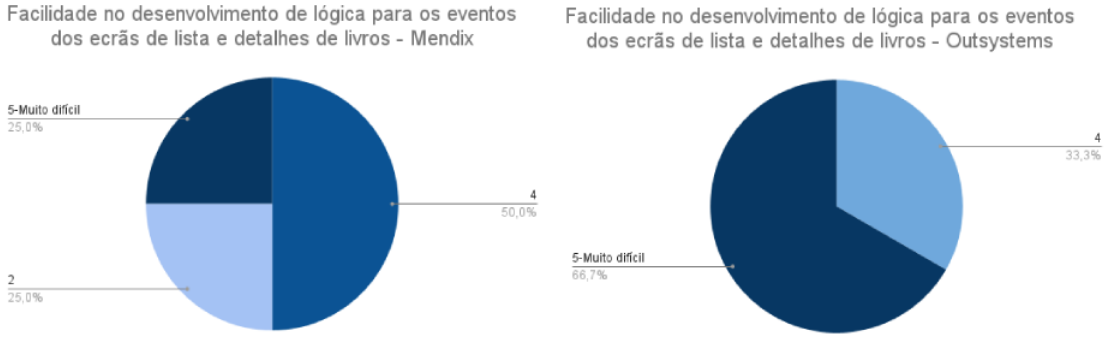


Figura 58: Distribuição de participantes por grau de facilidade na criação de eventos da página de lista de livros e detalhes de livros para casa plataforma

Uso de dataset

Relativamente ao uso de um *dataset* com a lista de todos os livros e as suas respetivas características, tarefa atribuída à *MS PowerApps*, os participantes foram questionados acerca da facilidade de incluir este *dataset* na plataforma e ainda sobre a facilidade de utilização destes mesmos dados. Para os participantes desta tarefa foi unânime que, a introdução de um *dataset* é uma tarefa simples de realizar (ver Figuras 59). Relativamente à facilidade de utilização dos dados dos livros, a maioria dos participantes considerou esta uma tarefa de simples execução 8ver Figura 60).

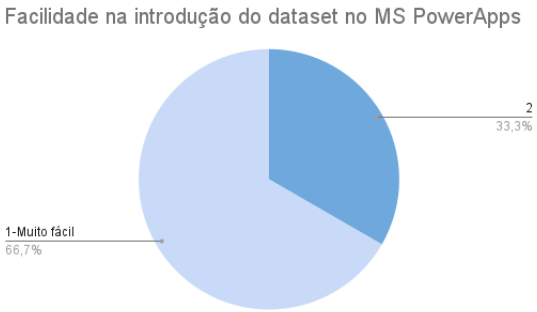


Figura 59: Contagem de participantes por grau de facilidade de introdução de um *dataset* na *MS PowerApps*

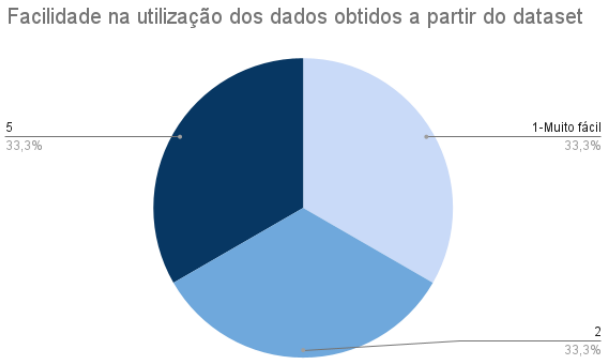


Figura 60: Contagem de participantes por grau de facilidade na utilização de dados do *dataset* na *MS PowerApps*

Na Tabela 5, encontra-se representada, para algumas das perguntas do questionário, a distribuição das respostas dos participantes tendo em conta o meio envolvente deste participante, isto é, se ele pertence ou não a um curso de informática. Assim, pode verificar-se que, no caso do *Mendix*, para os participantes sem conhecimento em informática, o desenvolvimento de todas as etapas requeridas revelou-se, unanimemente, de difícil execução. Por outro lado, alguns participantes desta mesma plataforma, mas com conhecimento em informática, revelaram ter facilidade na execução de algumas etapas. Ao observar os mesmos dados para a plataforma *MS PowerApps*, infere-se que, esta diferença de opiniões para ambos os grupos de participantes não é tão drástica, havendo participantes de ambos os grupos a considerar várias etapas tanto fáceis como difíceis. O mesmo se observa para a plataforma *Outsystems*, em que com exceção de uma etapa, relativa à introdução de dados dos utilizadores, existem participantes de ambos os grupos a considerar as várias etapas de dificuldade baixa e elevada.

	Mendix		MS PowerApps		Outsystems	
	Informática (4 participantes)	**Outras áreas (3 participantes)**	**Informática (4 participantes)**	**Outras áreas (3 participantes)**	**Informática (4 participantes)**	**Outras áreas (3 participantes)**
Facilidade de utilização da plataforma	1-Muito fácil(25%) 4(25%) 5-Muito difícil(50%)	4(67%) 5.Muito difícil (33%)	1-Muito fácil(25%) 2(25%) 3(50%)	2(67%) 4(33%)	3(50%) 4(50%)	3(33%) 5.Muito difícil (67%)
Facilidade de customização das aplicações	2(50%) 4(25%) 5-Muito difícil(25%)	4(33%) 5-Muito difícil(67%)	1-Muito fácil(25%) 2(25%) 3(50%)	1-Muito fácil(33%) 3(33%) 4(33%)	2(50%) 4(25%) 5-Muito difícil(25%)	2(33%) 3(33%) 5-Muito difícil(33%)
	Informática (2 participantes)	**Outras áreas (1 participante)**	**Informática (2 participantes)**	**Outras áreas (2 participante)**	**Informática (2 participantes)**	**Outras áreas (2 participante)**
Facilidade na introdução de dados dos utilizadores	1-Muito fácil(50%) 3(50%)	5-Muito difícil(100%	1-Muito fácil(50%) 5-Muito difícil(50%)	4(100%)	1-Muito fácil(50%) 2(50%)	3(50%) 4(50%)
Facilidade de desenvolvimento da lógica dos ecrãs de login e registo	4(50%) 5-Muito difícil(50%)	4(100%)	3(50%) 4(50%)	2(50%) 4(50%)	1-Muito fácil(50%) 4(50%)	3(50%) 4(50%)
	Informática (2 participantes)	**Outras áreas (2 participantes)**	**Informática (2 participantes)**	**Outras áreas (1 participantes)**	**Informática (2 participantes)**	**Outras áreas (1 participantes)**
Facilidade de utilização dos dados da REST API	1-Muito fácil(50%) 4(50%)	4(50%) 5-Muito difícil(50%)	N/A	N/A	4(50%) 5-Muito difícil(50%)	5-Muito difícil(100%)
Facilidade de desenvolvimento da lógica para os eventos dos ecrãs de lista e detalhes de livros	2(50%) 4(50%)	2(50%) 5-Muito difícil(50%)	N/A	N/A	4(50%) 5-Muito difícil(50%)	5-Muito difícil(100%)
Facilidade de uso dados dataset	N/A	N/A	1-Muito fácil(50%) 2(50%)	5-Muito difícil(100%)	N/A	N/A

Tabela 5: Distribuição de respostas dos participantes tendo em conta o curso

4.8.3 Outras questões sobre o uso da plataforma

No final de questionados sobre as suas tarefas específicas, os participantes foram ainda submetidos a uma questão de escolha múltipla sobre a sua experiência na utilização da plataforma *low-code*.

As opções disponíveis são apresentadas na Tabela 6.

Questão 1	Percentagem de respostas	Percentagem de respostas por plataforma
Eu acho que gostaria de utilizar esta plataforma mais frequentemente	5%	5% Outsystems
Eu achei esta plataforma mais complexa que o necessário	52%	14% PowerApps 14% Outsystems 24% Mendix
Eu considerei a plataforma fácil de utilizar	10%	10% PowerApps
Eu acho que iria precisar de ajuda de um profissional nesta plataforma para ser capaz de utilizar este sistema	43%	10% PowerApps 14% Mendix 19% Outsystems
Eu encontrei nesta plataforma várias componentes bem integradas	19%	14% PowerApps 5% Menidx
Eu considerei haver demasiada inconsistência na plataforma	15%	5% Outsystems 10% PowerApps
Eu considero que a maior parte das pessoas aprenderia esta plataforma rapidamente	19%	5% Mendix 14% PowerApps
Considerei esta plataforma muito incomoda de utilizar	38%	14% Mendix 24% Outsystems
Eu senti-me muito confiante a usar esta plataforma	10%	10% PowerApps
Eu precisei de aprender muitas coisas antes de conseguir utilizar a plataforma	62%	29% Outsystems 33% Mendix
As capacidades da plataforma foram de encontro as minhas necessidades	15%	5% PowerApps 10% Outsystems
A utilização da plataforma foi uma experiência frustrante	71%	14% PowerApps 24% Outsystems 33% Mendix
Com esta plataforma, tive de perder muito tempo a corrigir problemas	47%	14% PowerApps 14% Outsystems 19% Mendix

Tabela 6: Respostas à última questão realizada aos participantes

Ao analisar as respostas dadas a estas questões pôde constatar-se que, a maioria dos participantes demonstrou que a execução destas tarefas recorrendo a plataformas *low-code* constituiu uma experiência frustrante pela sua complexidade de aprendizagem, sendo esta resposta mais notada em participantes atribuídos ao *Mendix*. Para além disso, a maioria dos participantes considerou a plataforma atribuída mais complexa do que o necessário, sendo necessária uma grande aprendizagem prévia antes do desenvolvimento da tarefa proposta. Adicionalmente, grande parte dos participantes considerou que ao realizar as suas tarefas, precisou mais tempo do que o necessário para resolver problemas, sendo esta problemática mais notada no *Mendix*.

Por outro lado, também foram selecionados alguns aspetos positivos, principalmente em parte dos participantes que utilizaram o *Ms PowerApps*. Apenas participantes da *MS PowerApps* consideram que a

sua plataforma foi de fácil utilização, considerando ainda a sua utilização futuramente. Além disso, utilizadores desta plataforma foram de uma forma geral aqueles que apresentaram menos aspetos negativos. No entanto, segundo os participantes, esta plataforma foi a que apresentou maiores inconsistências.

4.8.4 Opinião dos participantes sobre as plataformas em estudo

No final deste estudo, ainda foi questionado a cada um dos participantes sobre se pretendiam acrescentar alguma opinião negativa ou positiva sobre as plataformas utilizadas para o desenvolvimento da sua tarefa.

Mendix

Os participantes que utilizaram a plataforma *Mendix*, apontaram algumas desvantagens tais como, o facto da interface da plataforma ser pouco intuitiva e confusa. Indicaram que a necessidade de criação de fluxos para qualquer ação torna a sua utilização demasiado complexa. Foi apontado ainda a dificuldade de utilização da nomenclatura por não ser semelhante a nenhuma linguagem de programação mais conhecidas. Por fim, notaram que a plataforma apresenta uma quantidade reduzida de templates, com baixa possibilidade de personalização (ver Anexo A1 com todas as respostas).

Relativamente às vantagens, os participantes referiram a facilidade de criação da base de dados e o *autocomplete* de expressões (ver Anexo A2).

MS PowerApps

Na *PowerApps* uma das desvantagens apresentadas foi o facto de ao longo do desenvolvimento das aplicações, o progresso não ser gravado automaticamente. Adicionalmente, foi referido a falta de ligação direta à documentação na plataforma, a ligação à bases de dados ser pouco intuitiva, permitir apenas o desenvolvimento de aplicações básicas e, por último, apresentar dificuldades exageradas na sua utilização (ver Anexo A3).

Relativamente às vantagens, os participantes apontaram a facilidade na utilização de fórmulas, facilidade na colocação de imagens quando comparadas às outras plataformas e não haver necessidade de instalação da plataforma (ver Anexo A4).

Outsystems

Tal como seria de esperar, os participantes apontaram algumas desvantagens na *Outsystems*. Entre estas desvantagens, mencionaram a interface pouco intuitiva, a fraca documentação, a dificuldade de perceber e corrigir erros, não ter suporte para ambiente *Linux*, ser pouco prático, necessitar de registo

para aceder à plataforma e, por fim, de uma forma geral ser mais difícil de utilizar comparativamente a código tradicional (ver Anexo A5).

No entanto, foram apresentadas vantagens, todas referentes à facilidade de customização visual (ver Anexo A6).

4.8.5 Interpretação dos dados recolhidos

Ao analisar as opiniões recolhidas pelos participantes, a utilização deste tipo de plataformas com pouco tempo de aprendizagem demonstrou ser uma experiência frustrante para a maioria dos participantes. Para além disso, estudando os dados recolhidos dos estudos com os participantes, pôde concluir-se que a *MS PowerApps*, demonstrou ser das plataformas mais intuitivas, sendo esta a única cujo o tempo disponível foi suficiente para desenvolvimento total da tarefa, no entanto, é importante notar que, uma vez não ser possível, nesta plataforma, realizar a tarefa de utilização da *API* gratuitamente, os participantes referentes a esta parte viram a sua tarefa mais facilitada. Os alunos distribuídos pela *Outsystems* e pelo *Mendix* mostram maior dificuldade geral para realização de tarefas. No entanto, foi no *Mendix* que os participantes dos dois grupos, isto é, pertencentes a informática ou a outros cursos, demonstraram mais dificuldade na execução geral das etapas necessárias ao desenvolvimento final.

Adicionalmente, foi notado que alunos com experiência em desenvolvimento de aplicações, aprenderam mais facilmente os conhecimentos necessários para o desenvolvimento das tarefas, enquanto, parte dos alunos que nunca desenvolveram uma aplicação, se mostraram muito confusos, não conseguindo começar a tarefa sem ajuda. Antes da execução das tarefas propostas, foi notado que os participantes com conhecimento em programação estavam mais hesitantes em relação ao uso deste tipo de plataformas, no entanto, após a experiência, alguns destes admitiram que, com um maior conhecimento sobre a plataforma, voltariam a utilizá-la para o desenvolvimento de *software*.

Capítulo 5

Conclusões e trabalho futuro

5.1 Conclusões

O desenvolvimento desta dissertação permite-me retirar algumas ilações. De facto, a utilização de plataformas *low-code* facilita uma grande parte do processo de desenvolvimento de aplicações, logo pelo facto de permitir que o mesmo desenvolvedor trate do *backend* e *frontend* sem grandes divisões. No entanto, a utilização deste tipo de plataformas não me pareceu algo tão intuitivo como descrito pelos seus proponentes. Sendo muitas vezes muito confuso o entendimento dos passos necessários a realizar para a inclusão de certas partes na aplicação, como, por exemplo, a utilização de certos *widgets* ou colocação de fluxos de eventos nos ecrãs. Sendo assim necessária uma grande aprendizagem para que estas plataformas sejam utilizadas de forma correta o suficiente para desenvolver todos os aspetos pretendidos numa aplicação.

No entanto, para um desenvolvedor que não apresente nenhum conhecimento sobre programação, acredito que seja mais fácil a aprendizagem necessária para a utilização deste tipo de plataformas do que aprender várias linguagens de programação que seriam inevitavelmente necessárias para proceder ao desenvolvimento do *frontend* e *backend* de uma aplicação. É importante referir que, para um desenvolvedor não experiente em programação, este tipo de plataformas exije um grau de conhecimento elevado, não pela aprendizagem da plataforma em si, mas também pelo facto de ser necessário apresentar alguns conhecimentos externos de informática, como bases de dados, simples fluxos de usabilidade de aplicações e conhecimentos sobre que tipo de dados externos usar, como a utilização de uma *API*.

Por outro lado, para desenvolvedores profissionais, existem alguns aspetos na utilização deste tipo de plataformas que acabam por se tornar de maior complexidade do que o uso de código tradicional, como, por exemplo, a utilização de uma simples *API*. Em contrapartida, algumas fases de desenvolvimento de uma aplicação são facilitadas utilizando este tipo de plataformas, até mesmo para um desenvolvedor profissional, como, por exemplo, a inclusão de base de dados ou a customização das interfaces das

aplicações.

Fazendo uma análise da utilização das três plataformas escolhidas para esta dissertação, isto é, o *Mendix* , *MS PowerApps* e *Outsystems*, posso afirmar que tanto pela minha experiência como ao observar os participantes do estudo, a *MS PowerApps* revelou ser a plataforma mais intuitiva e fácil de utilizar. No entanto, esta plataforma, na minha opinião, apresenta muitos entraves ao desenvolvimento de aplicações maiores como, por exemplo, aplicações empresariais. Assim, na minha opinião, das plataformas estudadas, as mais eficientes e completas para o desenvolvimento *low-code* são o *Mendix* e a *Outsystems*, sendo que o *Mendix* apresentou uma maior complexidade de utilização, revelando ser muito confuso de aprender inicialmente, apresentando também algumas limitações de customização e ecrãs sem recurso a *css*.

5.2 Perspetiva de trabalho futuro

Futuramente, seria interessante repetir o estudo realizado anteriormente de uma forma mais alargada, isto é, usando mais participantes e, em vez de colocar cada participante em contacto com a plataforma atribuída apenas durante duas horas, seria interessante colocar estes participantes em aulas de ensino sobre a plataforma, permitindo que estes realizem a tarefa após já terem conhecimentos suficientes sobre esta, possibilitando uma opinião mais consistente. Desta forma seria possível que a tarefa atribuída fosse uma pequena aplicação em vez de apenas uma parte desta. Seria ainda interessante adicionar mais objetos de estudo, permitindo ter mais opiniões.

Bibliografia

[1] Microflows-mendix documentation, . URL https://docs.mendix.com/refguide/microflows/.

[2] Nanoflows, . URL https://docs.mendix.com/refguide/nanoflows/.

[3] O que é um contentor? URL https://azure.microsoft.com/pt-pt/overview/what-is-a-container/.

[4] Data action - outsystems documentation. URL https://success.outsystems.com/Documentation/11/Reference/OutSystems_Language/Interfaces/Adding_Data_and_Logic/Data_Action.

[5] Technology churn, Nov 2014. URL https://wiki.c2.com/?TechnologyChurn.

[6] O que é orquestração?, Oct 2019. URL https://www.redhat.com/pt-br/topics/automation/what-is-orchestration.

[7] Entenda as vantagens e desvantagens do low-code, Jul 2020. URL https://blog.cronapp.io/vantagens-e-desvantagens-low-code/.

[8] Citizen developer vs. professional developer, Dec 2021. URL https://quixy.com/blog/citizen-developer-vs-professional-developer/.

[9] O que é o low-code? uma introdução ao desenvolvimento de aplicações low-code, Feb 2022. URL https://www.creatio.com/page/pt-pt/o-que-e-o-low-code.

[10] Md Abdullah Al Alamin, Gias Uddin, Sanjay Malakar, Sadia Afroz, Tameem Haider, and Anindya Iqbal. Developer discussion topics on the adoption and barriers of low code software development platforms. *Empirical Software Engineering*, 28(1):1–59, 2023.

[11] Hardit Bhatia. Power apps primer: Canvas vs. model-driven apps, Aug 2022. URL https://global.hitachi-solutions.com/blog/canvas-vs-model-driven-apps/.

[12] Alexander C Bock and Ulrich Frank. Low-code platform. *Business & Information Systems Engineering*, 63(6):733–740, 2021.

[13] Llewellyn Britz. Canvas app vs model-driven app: Differences, benefits and more, Jul 2021. URL https://www.columbusglobal.com/en-gb/blog/canvas-app-vs-model-driven-app-differences-benefits-and-more.

[14] Noleto Cairo. Rad: Conheça o desenvolvimento rápido de aplicação!, Aug 2020. URL https://blog.betrybe.com/tecnologia/rad/.

[15] Lenino Manuel Lima Dias. *Outsystems Logic Previewer*. PhD thesis, NOVA University of Lisbon, 2021.

[16] Cem Dilmegani. Citizen developers in 2022: Who are they & why do they matter, Feb 2022. URL https://research.aimultiple.com/citizen-developer/.

[17] Fulya Gürcan and Gabriele Taentzer. Using microsoft powerapps, mendix and outsystems in two development scenarios: An experience report. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 67–72. IEEE, 2021.

[18] George F Hurlburt. Low-code, no-code, what's under the hood? *IT Professional*, 23(6):4–7, 2021.

[19] Jenni Lehman. Magic quadrants and marketscopes: How gartner evaluates vendors within a market. *Gartner. com, available at: www. gartner. com/doc/486094*, 2008.

[20] Robin Lichtenthäler, Sebastian Böhm, Johannes Manner, and Stefan Winzinger. A use case-based investigation of low-code development platforms. In *ZEUS*, pages 76–83, 2022.

[21] Jerry Liptak and Lauren Horwitz. What is citizen development, and what is a citizen developer?, Feb 2021. URL https://searchsoftwarequality.techtarget.com/definition/citizen-development.

[22] James Martin. *Rapid application development*. Macmillan Publishing Co., Inc., 1991.

[23] Ricardo Martins, Filipe Caldeira, Filipe Sa, Maryam Abbasi, and Pedro Martins. An overview on how to develop a low-code application using outsystems. In *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pages 395–401. IEEE, 2020.

[24] Lionel Mew and Daniela Field. A case study on using the mendix low code platform to support a project management course. In *Proceedings of the EDSIG Conference ISSN*, volume 2473, page 3857, 2018.

[25] Duarte Oliveira. Quais são as vantagens das plataformas low-code?, Feb 2022. URL https://pt.primaverabss.com/pt/blog/low-code-programacao/.

[26] Justice Opara-Martins, Reza Sahandi, and Feng Tian. Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. *Journal of Cloud Computing*, 5(1): 1–18, 2016.

[27] Inês Margarida Monteiro de Moura Pinto. *Desenvolvimento Rápido de Aplicações-Comparação de soluções em Outsystems e Mendix*. PhD thesis, 2021.

[28] Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio, and Alfonso Pierantonio. Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 171–178. IEEE, 2020.

[29] Raquel Sanchis, Óscar García-Perales, Francisco Fraile, and Raul Poler. Low-code as enabler of digital transformation in manufacturing industry. *Applied Sciences*, 10(1):12, 2019.

[30] Naeem Ahmad Sattar. Selection of low-code platforms based on organization and application type. 2018.

[31] Joe Stangarone. Pros and cons of low-code development platforms, Mar 2019. URL https://www.mrc-productivity.com/blog/2019/03/pros-and-cons-of-low-code-development-platforms/.

[32] Khushi Talesra and GS Nagaraja. Low-code platform for application development. *International Journal of Applied Engineering Research*, 16(5):346–351, 2021.

[33] Paul Vincent, Kimihiko Iijima, Mark Driver, Jason Wong, and Yefim Natis. Magic quadrant for enterprise low-code application platforms. *Gartner report*, 2019.

[34] Kumar Vivek. O que é o power apps? - power apps, Feb 2022. URL https://docs.microsoft.com/pt-pt/powerapps/powerapps-overview.

[35] Robert Waszkowski. Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, 52(10):376–381, 2019.

[36] Jason Wong, Kimihiko Iijima, Adrian Leow, Akash Jain, and Paul Vincent. Magic quadrant for enterprise low-code application platforms, Sep 2021. URL https://www.gartner.com/doc/reprints?id=1-295WSI86&ct=220217&st=sb.

Apêndice A
Opiniões dos Participantes

Desvantagens do Mendix apresentadas pelos participantes cuja tarefa atribuída pertence a esta plataforma
Sendo uma aplicação low-code, seria de esperar que apresentasse uma interface intuitiva e de fácil utilização. Contudo é demasiado complexa de utilizar e acima de tudo é necessário criar fluxos para tudo. A nomenclatura usada é extremamente complicada e podia ser parecida às linguagens mais conhecidas como Java ou Python, em vez de uma nomenclatura própria não explicada em lado nenhum.
Os templates fornecidos podiam ser melhores e dar para personalizar mais. Para mudar a cor de um botão ou até do fundo é necessário começar uma página em branco em vez de utilizar o template. Até CSS puro é mais fácil, visto só ser necessário mudar uma cor.
A plataforma não providencia um uso fácil dos menus nem é de fácil interpretação inicial. Após a consulta da documentação foi fácil consumir a REST API. O grande problema desta plataforma é a tentativa de modificação dos componentes já criados uma vez que as dependências entre ”microflows”e ”diagrams” não são bem explicadas na documentação e qualquer mínima alteração causa uma enorme dificuldade em manter um programa coerente e coeso.
É muito confusa

Tabela A1: Opiniões negativas registadas pelos participantes sobre a utilização de Mendix para o desenvolvimento de aplicações

Vantagens do Mendix apresentadas pelos participantes cuja tarefa atribuída pertence a esta plataforma
Pelo menos não é necessário criarmos nós a base de dados.
Autocomplete nas expressões

Tabela A2: Opiniões positivas registadas pelos participantes sobre a utilização de Mendix para o desenvolvimento de aplicações

Desvantagens do MS PowerApps apresentadas pelos participantes cuja tarefa atribuída pertence a esta plataforma
Não grava automaticamente
A funcionalidade de pesquisa podia dar resultados de funções, e podia haver ligações diretas entre a plataforma e a sua documentação.
Acredito que a Plataforma tem bastantes desvantagens, sobretudo para pessoas com prévia experiência em desenvolvimento de software. As Bases de Dados(BDs) são pouco intuitivas a sua criação tal como a criação de funções/código para a gestão da mesma.
A plataforma é muito simples, o que é muito bom para fazer aplicações, mas parece limitada em alguns aspetos e por isso parece só servir para aplicações mais pequenas ou rápida.
Apresentou dificuldades exageradas para a realização de tarefas que deveriam ser simples.

Tabela A3: Opiniões negativas registadas pelos participantes sobre a utilização de MS PowerApps para o desenvolvimento de aplicações

Vantagens do MS PowerApps apresentadas pelos participantes cuja tarefa atribuída pertence a esta plataforma
O código usado é bastante simples. É bastante fácil colocar imagens comparativamente a outras plataformas.
O facto de não ser preciso instalar é perfeito

Tabela A4: Opiniões positivas registadas pelos participantes sobre a utilização de MS PowerApps para o desenvolvimento de aplicações

Desvantagens do Outsystems apresentadas pelos participantes cuja tarefa atribuída pertence a esta plataforma
A interface não é intuitiva para quem usa nas primeiras vezes, e a documentação não ajuda muito (pelo menos nos passos que tive de fazer precisei de auxílio ao criador do estudo, uma vez que a documentação era insuficiente). Alguns passos foram mais trabalhosos ou complicados do que se fossem feitos em código normal.
Leaky abstractions - a interface exposta ao programador parece, ao princípio, simples, mas para resolver alguns problemas é preciso procurar profundamente a causa/solução do problema, enquanto que numa linguagem tradicional isto seria mais óbvio, já que o programador estaria, em princípio, mais familiarizado com os detalhes do código. A atração por plataformas low-code vem da aparente facilidade na aprendizagem e fácil desenvolvimento de aplicações.No entanto, sinto que a complexidade que existe em programação tradicional continua a existir em plataformas low code, apenas escondida por debaixo de camadas de abstração (enquanto que em programação tradicional esta complexidade é óbvia). Registo necessário - é preciso registar uma conta para utilizar esta plataforma. Se fosse desenvolvido noutra linguagem tradicional, e.g. Python/JS, não seria necessário. Desenvolvimento não é cross-platform - não há suporte em Linux. Não é facilmente integrável num browser - com programas tradicionais (feitos em texto), o código pode ser visualizado facilmente na web, e.g. Github. Também é fácil enviar secções do código por mensagens e embeber código tradicional. O mesmo não é possível com esta plataforma.
Menos prático do que utilizar um editor de texto.

Tabela A5: Opiniões negativas registadas pelos participantes sobre a utilização de Outsystems para o desenvolvimento de aplicações

Vantagens do Outsystems apresentadas pelos participantes cuja tarefa atribuída pertence a esta plataforma
Facilidade na edição da interface gráfica.
Facilidade de costumização visual

Tabela A6: Opiniões positivas registadas pelos participantes sobre a utilização de Outsystems para o desenvolvimento de aplicações