Universidade do Minho
Escola de Engenharia

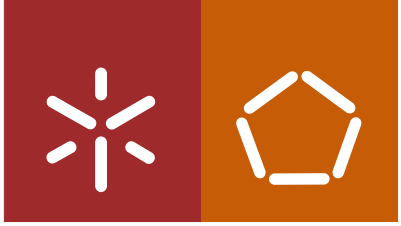
Joel Filipe Pereira Sousa

Frameworks no Desenvolvimento de Software: O caso da Elevation da PRIMAVERA BSS

outubro de 2015

Joel Filipe Pereira Sousa **Frameworks no Desenvolvimento de Software: O caso da Elevation da PRIMAVERA BSS**

UMinho | 2015



Universidade do Minho
Escola de Engenharia

Joel Filipe Pereira Sousa

Frameworks no Desenvolvimento de Software: O caso da Elevation da PRIMAVERA BSS

Dissertação de Mestrado
Mestrado Integrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação do
Professor Dr. Rui Dinis Sousa

outubro de 2015

DECLARAÇÃO

Nome: Joel Filipe Pereira Sousa
Endereço: joelfps@gmail.com
Telefone: 925801714
Número do Bilhete de Identidade: 13645519

Título:
Frameworks no Desenvolvimento de Software: O caso da Elevation da PRIMAVERA BSS

Orientador(es):
Professor Dr. Rui Dinis Sousa

Ano de conclusão: 2015

Designação do Mestrado: Mestrado Integrado em Engenharia e Gestão de Sistemas de Informação

Universidade do Minho, __/__/____
Assinatura: _____

AGRADECIMENTOS

Gostaria de agradecer às seguintes pessoas pela ajuda e apoio durante o decorrer desta dissertação, pois sem elas não seria possível finalizá-la.

Primeiramente gostaria de agradecer ao meu orientador Prof. Doutor Rui Dinis Sousa, pelo apoio e disponibilidade prestadas durante o decorrer desta dissertação de mestrado. Gostaria também de agradecer ao Dr. Luís Matos, o meu supervisor na PRIMAVERA BSS, pelo suporte dado na empresa, e pela ajuda na integração da mesma. Um muito obrigado por me ajudarem a crescer tanto pessoalmente como profissionalmente.

Aos meus pais, Manuel e Maria da Graça, à minha irmã, Sandra, por todo o seu apoio prestado, por terem acreditado em mim, e me possibilitarem estar a concluir esta etapa da minha vida. A quem estarei grato para sempre pelo seu apoio e carinho.

À minha namorada, Paula, que sempre me apoiou e me deu forças para continuar nos momentos mais difíceis.

À empresa PRIMAVERA BSS, pela oportunidade de trabalhar em projetos inovadores e aliciantes com uma equipa de grande excelência profissional e pela oportunidade de continuar com a minha carreira profissional integrando as suas equipas.

A todos os meus colegas de trabalho na PRIMAVERA BSS, em especial à equipa do EAM e ao Flávio Rodrigues, por todo o apoio prestado, tanto na disponibilidade para serem entrevistados e realizarem questionários mas também pela preocupação que tiveram em ajudar em tudo o que fosse possível.

A todos os meus colegas de mestrado e de licenciatura, que me acompanharam ao longo destes anos, principalmente o Rafael Ribeiro, Vítor Pinheiro, Rui Fernandes, João Cadime, Tiago Costa, João Gonçalves e Henrique Carvalho, por todos os grandes momentos que vivemos ao longo destes anos, tanto ao nível académico como pessoal.

Por último, gostaria também de agradecer a todos os meus professores que me acompanharam durante a minha passagem nesta universidade, o meu muito obrigado por tudo o que me ensinaram.

RESUMO

O desenvolvimento de software é cada vez mais complexo. De forma a permitir que a empresas continuem competitivas, as equipas de desenvolvimento utilizam *frameworks* de desenvolvimento de forma a reduzir o esforço e tempo necessários para as empresas darem uma reposta eficaz e eficiente às exigências do mercado.

Sendo a PRIMAVERA BSS uma empresa que gosta de estar sempre ao mais alto nível no que diz respeito à produção de *software*, ao longo dos últimos sete anos, uma das equipas de *software development* da PRIMAVERA BSS, desenvolveu a *framework* Elevation. No entanto, não é ainda evidente as mais valias para os colaboradores da PRIMAVERA BSS decorrente da adoção da Elevation no processo de desenvolvimento de software, pelo que este trabalho de dissertação propõe um estudo de caso que responda à seguinte questão: "Quais as vantagens e desvantagens da utilização da *framework* Elevation?".

Para além da resposta à questão evidenciando as vantagens e desvantagens, os resultados deste trabalho incluem ainda um conjunto de recomendações para a evolução da *framework* Elevation, as quais podem também ser tidas em consideração por outras empresas no contexto de outras *frameworks*.

ABSTRACT

The Software development is becoming more complex. The companies need to use software development frameworks, in order to reduce the effort and time needed to the companies be able to respond to the market requirements, so they can achieve the level of competitiveness required nowadays.

Being PRIMAVERA BSS a company that likes to be always on the highest level in the software production market, during the last seven years, their software development teams have been developing Elevation framework. However, is not yet clear that its a added value to the PRIMAVERA BSS employees in the software development process, so this dissertation will porpose a case study that anwsers the following question: "What are the advantages and disadvantages of using the framework Elevation?"

In addition to the anwsers to the question the results of this work include a set of recommendations for the development of Elevation framework which can also be taken into account by other companies in the context of other frameworks.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1

INTRODUÇÃO

Este capítulo introdutório ao trabalho de dissertação começa por descrever qual o propósito do documento, o enquadramento e qual a motivação para a sua realização. Por fim é descrita a estrutura do documento.

1.1 ENQUADRAMENTO E MOTIVAÇÃO

Desenvolver aplicações é cada vez mais exigente pois a complexidade é cada vez maior. Vão aparecendo assim problemas com a qualidade dos *software*, tornando a manutenção e evolução destes *softwares* muito cara.

Uma *framework* pode ser definida como algo reutilizável no todo ou em parte, de um sistema, representado por um conjunto de classes abstratas e e interações entre as suas instâncias. Outra definição muito comum é que uma *framework* é um esqueleto de uma aplicação que pode ser personalizada por um *developer* de aplicações (Johnson, 1997b; Wirfs-Brock and Johnson, 1990; Johnson and Foote, 1988). As *frameworks* de desenvolvimento de *software* são estruturas de classes que constituem implementações incompletas, que estendidas, permitem produzir diferentes aplicações de *software*. O uso de *frameworks* permite uma redução de esforço e tempo para o desenvolvimento de um *software* (Wentzel, 1994; Gaffney and Cruickshank, 1992). O uso de *frameworks* não é um complemento simples nos processos de desenvolvimento atuais. A implementação de uma *framework*, pode mudar dramaticamente os processos de criação de *software*. Toda a estrutura de uma *software factory* tem de ser adaptada ao uso da *framework*, pois os métodos, modelos e processos e tecnologias terão de ser adaptados (Griss and Wentzel, 1994). Os benefícios do uso de uma *framework* advêm da modularidade, reutilização de código e extensibilidade que esta oferece aos programadores (Fayad and C. Schmidt, 1997; Pree and Sikora, 1997) .

No contexto da PRIMAVERA BSS, a *framework* Elevation traz ferramentas e métodos para criar mais rapidamente aplicações de negócio, usando os princípios de design e tecnologias de estado de arte. Com o uso da *framework* Elevation, deverá ser possível obter melhorias no tempo de desenvolvimento de um *software*, desenvolvimento com menos erros, reduzir custos, melhorar a experiência do utilizador e acima de tudo aumentar a produtividade.

A PRIMAVERA BSS sempre olhou para as evoluções tecnológicas como oportunidades únicas para reinventar o negócio e se diferenciar. Com as exigências dos utilizadores a mudarem, de dia para dia, as empresas também têm que mudar, desenvolvendo-se e adaptando-se para conseguir acompanhar esta mudança.

A equipa técnica da consultoria é chamada com frequência a dar resposta a necessidades dos clientes não respondidas nos produtos base. No ERP PRIMAVERA existe um vasto conhecimento no desenvolvimento de soluções específicas, mas com o lançamento no mercado de soluções suportadas na *framework* Elevation, em particular o EAM *(Enterprise Asset Management)*, foi identificada a necessidade de desenvolver *know-how* nesta *framework*, para dar resposta a solicitações dos clientes não respondidas nos produtos base.

Realizando uma reflexão sobre a *framework* Elevation da PRIMAVERA BSS, será possível obter conclusões quanto a estes desenvolvimentos, qual o esforço necessário para fazer alguns desenvolvimentos, concluindo quais as vantagens e desvantagens da utilização da *framework*. Surge assim a seguinte questão de investigação: "Quais as vantagens e desvantagens da utilização da *framework* Elevation?".

1.2 PROBLEMA E OBJETIVO DA INVESTIGAÇÃO

A finalidade deste estudo é responder à questão: "Quais as vantagens e desvantagens da utilização da *framework* Elevation?".

Assim o principal foco do trabalho realizado estará ligado à realização de um estudo de caso sobre a *framework*.

Este estudo de caso poderá ser desdobrado nos seguintes objetivos:

- Identificar vantagens e desvantagens da *framework* através da leitura da documentação, realização de entrevistas e questionários aos programadores
- Refletir sobre a lista de vantagens e desvantagens da *framework* identificados.
- Tecer um conjunto de recomendações

1.3 ORGANIZAÇÃO DO DOCUMENTO

Após o enquadramento do trabalho e da explanação da motivação, concluímos o primeiro capítulo, Introdução (1), com a descrição da estrutura do documento. O segundo capítulo, Frameworks de Desenvolvimento de Software (2), é composto por duas partes. Na primeira parte, será abordado o conceito de *framework*, a sua caracterização, diferença entre *frameworks* e bibliotecas de classes, diferença entre *frameworks* e *design patterns*, vantagens e desvantagens de uma *framework* e por último qual o processo de aprendizagem de uma *framework*. Na segunda parte, é explicada a *framework*

Elevation, sendo descritos os seus princípios de design, arquitetura e o processo de desenvolvimento de aplicações sob a *framework*. No capitulo Abordagem Metodológica (3) é explicada a estratégia de pesquisa bibliográfica definida, qual a metodologia de investigação selecionada e descrita as seguintes fases do estudo: Planeamento e Design e Preparação e Recolha de dados No quarto capitulo(4) é apresentada a análise estatística dos resultados obtidos. No quinto e último capitulo são apresentadas as Conclusões (5) desta dissertação.

2

FRAMEWORKS DE DESENVOLVIMENTO DE SOFTWARE

No presente capitulo é clarificado o conceito de *framework*, começando pela sua definição, caracterização, diferença entre *frameworks* e bibliotecas de classes, para se abordar, depois a diferença entre *frameworks* e *design patterns*, benefícios e custos de uma *framework* e qual o seu processo de aprendizagem. Será também descrita a *framework* de desenvolvimento da PRIMAVERA BSS, descrevendo os seus princípios de design, a sua arquitetura e o processo de desenvolvimento.

2.1 REUTILIZAÇÃO DE SOFTWARE

O conceito de reutilização de *software* surgiu no seminário da NATO (*North Atlantic Treaty Organization* (NATO, 1949), quando foi considerada a possibilidade de desenvolvimento de aplicações através de partes previamente construidas, adaptáveis e com qualidade comprovada. Partes são consideradas blocos de construção (componentes), que se encaixam uns nos outros para desenvolverem novos sistemas (McIlroy, 1968). A reutilização foi pela primeira vez atingida por volta de 1972, através de sub rotinas e bibliotecas de funções. O conceito base de programação modular foi definido, e os engenheiros de *software* perceberam que esses módulos poderiam ser utilizados como componentes reutilizáveis (Szyperski, 2002).

Por volta dos anos 80 com o paradigma de programação orientada a objetos a tornar-se cada vez mais popular, a reutilização de *software* aumentou também a capacidade das classes herdarem métodos das classes pai, criando assim estruturas hierárquicas de classes, permitindo fazer alterações no código das classes sem necessitar de fazer alterações no código fonte (Johnson and Foote, 1988). Foi assim que surgiram as bibliotecas de classes, que tinham como função agrupar classes que poderiam ser reutilizadas noutras aplicações.

As bibliotecas de classes e o paradigma de programação orientada a objetos, não forneciam a reutilização necessária, sendo ainda necessário que os programadores escrevessem bastante código e definissem o fluxo de controlo da aplicação. Devido à falta de reutilização fornecida pelas bibliotecas de classes, começaram a surgir por volta dos anos 90 as *frameworks* orientadas a objetos, permitindo a reutilização não apenas de código como de design. Este foi um fator diferenciador para o aumento da popularidade do desenvolvimento de *software* orientado a componentes (Lopes, 2008). O principal objetivo deste tipo de desenvolvimento era reduzir a complexidade do paradigma de programação

orientada a objetos, facilitando a integração e personalização das aplicações (Nierstrasz and Dami, 1995).

A primeira *framework* largamente utilizada foi o Model-View-Controller (MVC), implementada em *SmallTalk-80* (Krasner and Pope, 1988). Mas a maioria das *frameworks* utilizadas são para domínios técnicos tais como interfaces com utilizadores ou distribuição, como por exemplo a *framework MacApp* (Schmucker, 1986), que é especifica para aplicações Macintosh. As *frameworks* para aplicações específicas são normalmente privadas.

A reutilização iniciou-se através da reutilização modular utilizando sub-rotinas e bibliotecas de funções, evoluindo depois para a reutilização de código, utilizando as bibliotecas de classes com o aparecimento do paradigma de programação orientada a objetos. E por último surgiu uma reutilização mais completa com o uso de *frameworks*, onde é reutilizado não somente o código como o design da aplicação, reduzindo bastante o código que é necessário ser escrito pelos programadores.

2.2 FRAMEWORKS

O termo *framework*, sendo muito abstrato, é utilizado em diferentes contextos, existindo portanto bastantes definições distintas. Nesta revisão de literatura focou-se as *frameworks* de desenvolvimento orientadas a objetos. Se compilarmos várias destas definições, poderemos obter uma definição de *framework* mais completa.

Uma das primeiras definições de *framework* surgiu em 1988 com a publicação do artigo *"Designing Reusable Classes"* onde Johnson and Foote (1988) em que definem que "uma *framework* é um conjunto de classes que constituem um projeto abstrato para uma família de problemas relacionados, e suporta a reutilização a uma maior granularidade do que classes. (...) À medida que as *frameworks* se tornaram mais refinadas, levaram à criação de componentes *"black box"* que podem ser utilizados sem a necessidade de conhecer as suas implementações".

Wirfs-Brock and Johnson (1990) definem que uma *framework* é "uma coleção de classes abstratas e concretas e as interfaces entre elas, e é o projeto para um subsistema". Taligent (1993) defende que "uma *framework* é um conjunto de blocos pré fabricados que os programadores podem usar, estender, ou personalizar para soluções computacionais especificas" e descreve que "Com o uso de *frameworks*, os programadores de *software* não necessitam de começar do zero todas as vezes que desenvolvem uma aplicação. Uma *framework* deriva de uma coleção de objetos, assim sendo tanto design e o código da *framework* podem ser reutilizados."

Johnson (1997b) resume que a definição mais utilizada é que "uma *framework* é um projeto reutilizável de todas ou apenas uma parte de um sistema que é representado por um conjunto de classes abstratas e a forma como elas interagem". Argumenta ainda que outra definição muito comum é que "uma *framework* é o esqueleto de uma aplicação que pode ser otimizado por um programador". Podemos encontrar ainda mais definições de *framework* em (Deutsch, 1989; Roberts and Johnson, 1996; Goldberg and Robson, 1983).

Idealmente as tecnologias de reutilização deveriam fornecer componentes, que poderiam facilmente serem integrados, de forma a criarem novos sistemas. O programador não necessitaria de saber como o componente era implementado e seria fácil de aprender. Originalmente esta seria a visão de reutilização de *software*, uma visão baseada em componentes. Mas as *frameworks* não se tornaram em componentes de *software* como era previsto inicialmente. As *frameworks* são mais personalizáveis do que qualquer componente, e têm interfaces mais complexas (Johnson, 1997b).

Basicamente uma *framework* é uma implementação incompleta de um projeto que tem fatores e variáveis que se assemelham a um domínio[1] especifico da aplicação. Contém elementos prontos a utilizar (*frozen spots* (Burnett et al., 1995)), e possibilita formas de personalização através de pontos de variação (figura 1). As *frameworks* são personalizadas utilizando *Inversion of Control*[2], um mecanismo onde na *framework* invoca as partes especificas da aplicação.

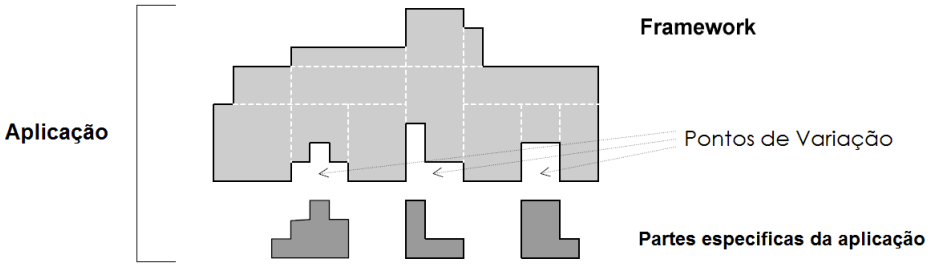


Figura 1.: Framework e aplicação: pontos de variação.
(fonte: (Lopes, 2008))

2.2.1 *Caraterização*

Johnson and Foote (1988) definem que uma característica importante das *frameworks* é que "os métodos definidos pelo utilizador para personalizar a *framework* vão ser chamados por ela própria, em vez da aplicação do utilizador. A *framework* normalmente tem o papel de coordenar e sequenciar as atividades da aplicação". Eles foram também os autores que mais influenciaram a caracterização das *frameworks*, e distinguiram-nas em dois tipos (Figura 2), as *frameworks* white-box e black-box.

- *White-box* - Uma *framework* é denominada de *white-box*, quando a extensibilidade das aplicações desenvolvidas, através das *frameworks*, são definidas adicionando métodos a subclasses de uma ou mais classes geradas pela *framework*. Este tipo de *framework* exige que o utilizador saiba como funciona a implementação da *framework* de modo a conseguir utilizá-la (Johnson and Foote, 1988). Este tipo de *framework* é muitas as vezes chamado de *frameworks* "architecture-driven" ou "inheritance-focused" (Mattsson, 2000).

1 O Domínio é o campo de estudo que define um conjunto de requisitos, terminologias e funcionalidades para qualquer aplicação construida para resolver um problema na área de programação.
2 Também conhecido como Hollywood Principle: “Don’t call us, we’ll call you.”

- *Black-box* - Uma *framework* é caracterizada de *black-box*, quando são utilizados um conjunto de componentes que definem os comportamentos específicos. Todos ou quase todos os componentes são disponibilizados pela *framework* através de bibliotecas de componentes. Este tipo de *framework* apenas exige que o utilizador conheça a interface externa dos componentes (Johnson and Foote, 1988). Estas *frameworks* são muitas vezes referidas como "data-driven" *frameworks* (Mattsson, 2000).

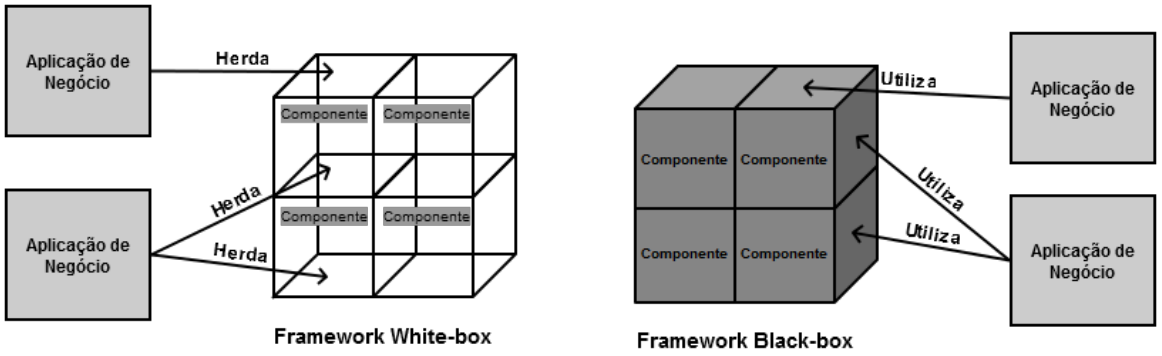


Figura 2.: Frameworks white-box e black-box

O principal problema das *frameworks white-box* é a necessidade de criar várias subclasses, dificultando os utilizadores a perceberem o design da aplicação de modo a conseguir altera-lo. Outro problema que surge é a as *frameworks white-box* têm um grau de dificuldade de aprendizagem mais elevado, visto ser necessário perceber como a *framework* foi implementada. Já por outro lado as *frameworks black-box* utilizam componentes já disponibilizados pela *framework*, geralmente representados graficamente, tornando mais fácil para os utilizadores perceberem o seu design. Para os novos utilizadores apenas precisam de perceber as interfaces externas dos componentes, não sendo preciso perceber a sua implicação. As *frameworks black-box* têm um grau de dificuldade de aprendizagem menor, mas são muito menos flexíveis quando se trata de fazer adaptações às aplicações geradas pela *framework* (Johnson and Foote, 1988). Para estes investigadores, Johnson and Foote (1988) uma maneira de caracterizar as diferenças entre *white-box* e *black-box* é "observando que nas *frameworks white-box*, o estado de cada instância está implicitamente disponível para todos os métodos da *framework*, assim como as varáveis globais estão disponíveis num programa em PASCAL. Numa *framework black-box*, qualquer informação passada para os constituintes da *framework* deve ser passada implicitamente. As *frameworks* white-box dependem das regras do âmbito intra-objeto para evoluírem, sem serem forçadas a subscrever um explicito e rígido protocolo, que pode limitar o design do processo prematuramente." *White-box* e *black-box* formam as fronteiras dos princípios de utilização e design de uma *framework*. A maioria das *frameworks* está situada entre estas duas fronteiras, que se denominam por *frameworks grey-box* (Figura 3). Este tipo de *frameworks* tenta usufruir das vantagens de design das *frameworks white-box* e *black-box*, tentando evitar as desvantagens de cada uma (Lopes, 2008).

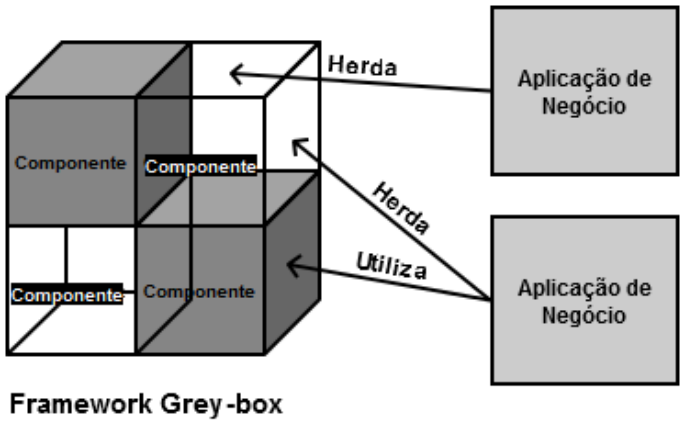


Figura 3.: Framework grey-box

De acordo com Taligent (1993), as *frameworks* podem ser caracterizadas de acordo com o domínio que são aplicadas, Taligent Inc. catalogou as *frameworks* em três categorias diferentes:

- *Frameworks de Aplicação* - agrupa conhecimentos que podem ser aplicados a um vasto leque de programas. Este tipo de aplicações fornecem funcionalidades horizontais, que podem ser aplicadas transversalmente em diversos domínios da aplicação. Um exemplo tipo deste tipo de *frameworks* são os *Graphical User Interface* (GUI) *frameworks*.
- *Frameworks de Domínio* - agrupa conhecimento e de um domínio especifico; este tipo de *frameworks* contém uma funcionalidade vertical, para um determinado domínio.
- *Frameworks de Suporte* - disponibilizam serviços que resolvem problemas computacionais específicos, como por exemplo acesso a ficheiros, drivers dos dispositivos ou computação distribuída.

2.2.2 *Frameworks vs Bibliotecas de Classes*

Frameworks e bibliotecas de classes são muitas vezes equiparadas, mas existem diferenças bastante claras entre elas. Embora algumas bibliotecas possam ter funcionalidades semelhantes a *frameworks*, e *frameworks* podem ser utilizadas como uma biblioteca de classes e as bibliotecas e as *frameworks* partilhem diversas caraterísticas, as *frameworks* podem ser distinguidas das bibliotecas de classes da seguinte forma (Taligent, 1993): uma biblioteca de classes, é um conjunto de classes relacionadas por herança, que têm como propósito uma funcionalidade geral na aplicação. Ao contrário das *frameworks*, as *bibliotecas de classes* não abordam um domínio da aplicação. Os fatores diferenciadores mais notáveis são: a capacidade de reutilização que é possibilitada, o conhecimento necessário para a sua utilização e a sua arquitetura. As *bibliotecas de classes* são consideradas formas passivas de reutilização, pois são invocadas pela aplicação. As classes das bibliotecas podem ser reutilizadas in-

dividualmente. Os programadores podem definir o fluxo de controlo, e a maioria da estrutura da sua aplicação, e apenas precisam de ter conhecimento das interfaces externas da *bibliotecas de classes*. Por outro lado, as *frameworks* são ativas, isto porque, já tem um fluxo de controlo definido que invoca componentes específicos da aplicação (*inversion of control*) nos pontos de variação. O fluxo de controlo passa e liga todas as classes, e essa é uma das razões para que as classes sejam reutilizadas como um todo. Com a utilização de *frameworks* a reutilização não limitada apenas ao código, como nas bibliotecas de classes, mas considera toda a sua estrutura e interações entre classes. Os programadores precisam de perceber o design da *framework*, e as suas interações de modo a conseguir estendê-la. É o fluxo de controlo que define o comportamento da *framework*, e das aplicações que são construidas através da *framework* (Lopes, 2008).

Bibliotecas de Classes	**Frameworks**
Conjunto de classes instanciadas pelo cliente	Fornece costumização através de subclasses
O cliente chama funções	Chama funcções dos clientes
Nenhum fluxo de controlo predefinido	Controla o fluxo de execução
Nenhuma interação predefinida	Define a interação entre objetos
Sem comportamento predefinido	Fornece comportamento predefinido

Tabela 1.: Bibliotecas de Classes vs Frameworks

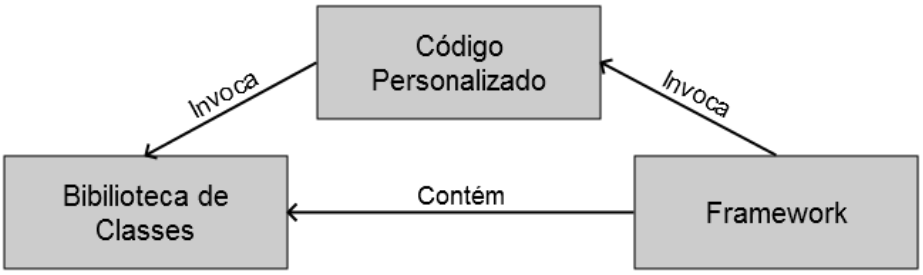


Figura 4.: Esquema *Frameworks* e Biblioteca de Classes.

2.2.3 *Frameworks vs Design Patterns*

Gamma et al. (1998) definem *design patterns* como "descrições de objetos comunicativos e classes que são customizadas para resolver um problema de design geral num contexto em particular". Se compararmos esta definição com a definição de *framework*, podemos salientar logo algumas evidências diferenciadoras. Independentemente de ambos serem compostos por classes e interfaces, *design patterns* são mais abstratos que *frameworks*, pois os *design patterns* descrevem soluções de design para um problema genérico sem terem um domínio em particular. Um *design pattern* geralmente não inclui código, ao contrário de uma *framework*, onde o código pode ser estudado e reutilizado diretamente. São menos especializados do que *frameworks*, pois não têm um domínio, nem uma arquitetura

de aplicação. Já por outro lado as *frameworks* incluem código, e esse código pode ser reutilizado (Gamma et al., 1998). Geralmente uma *framework* contém vários *design patterns*, por exemplo, os *design patterns* são utilizados para documentar *frameworks*.

2.2.4 *Benefícios e Custos*

Fayad and C. Schmidt (1997) apontam modularidade, reutilização, extensibilidade como os principais benefícios que uma *framework* oferece aos programadores. As *frameworks* aumentam a modularidade incorporando detalhes de implementações voláteis por trás de interfaces estáveis. Esta modularidade ajuda a melhorar a qualidade das aplicações, localizando o impacto de mudanças tanto no design como na implementação. Esta localização reduz o esforço necessário para perceber e realizar manutenções nas aplicações existentes. A reutilização, deve-se ao facto das *frameworks* fornecerem interfaces estáveis que aumentam a reutilização de código definindo componentes genéricos que podem ser reutilizados para criar novas aplicações. A reutilização fornecida pelas *frameworks* beneficia do conhecimento e perícia do domínio aplicado já na *framework*, evitando assim ter de recriar e revalidar soluções para aplicações cujo os objetivos e os requisitos são os mesmos; Extensibilidade é uma das vantagens que a *framework* possibilita, pois as *frameworks* oferecem formas para que seja possível estender as aplicações criadas pelas *frameworks*, permitindo personalizar as aplicações com funcionalidade especificas.

De acordo com Lopes (2008), as *frameworks* são ferramentas que podem ser utilizadas e que trazem diversas vantagens a quem as utiliza. Devido às *frameworks* possibilitarem a reutilização tanto a nível de código como a nível de design, existem dois tipos de vantagens. As vantagens relacionadas com reutilização de código são: redução de linhas de código que necessitam de ser escritas, que por sua vez facilitam a sua manutenção; redução de bugs nas aplicações, visto que o código será maioritariamente gerado pela *framework* que por sua vez já foi testado inúmeras vezes. Falando agora do segundo tipo de vantagens, as vantagens de reutilização do design são: transferência de conhecimento e especialização dos especialistas do domínio para os programadores das aplicações, simplesmente utilizando a *framework*; o domínio do problema já foi analisado, o design da solução já foi testado, e o seu funcionamento foi confirmado; existem interfaces estáveis que foram concebidas para aumentarem a extensibilidade da *framework*, permitindo a personalização das funcionalidades e serviços específicos da nova aplicação.

As *frameworks* não oferecem apenas benefícios e vantagens, também têm custos e pontos fracos. O desenvolvimento de uma *framework* é muito difícil e caro, porque desenvolver uma *framework* com qualidade, extensível e reutilizável, requer um maior conhecimento e especialização do domínio da aplicação. Uma *framework* é muito complexa o que faz com que a curva de aprendizagem seja muito grande, necessitando de um elevado investimento a nível de aprendizagem. Outra desvantagem é a durabilidade e alterações necessárias à *framework*. Como os requisitos das aplicações mudam frequentemente, as *frameworks* necessitam de algumas alterações de modo a poderem responder aos

novos requisitos. A correção de *bugs* é outro dos problemas existentes na *framework*. Mesmo com uma *framework* com um design bem desenvolvido, fazer *debug* às aplicações desenvolvidas pode ser difícil, porque os componentes genéricos tornam-se mais difíceis de validar, e quando o nível de abstração é muito grande, torna-se difícil de distinguir se o *bug* é da aplicação ou da *framework*. E com o uso do conceito *inversion of control (IoC)*, o fluxo de controlo oscila entre a infraestrutura da *framework* e a aplicação, o que faz com que os programadores não percebam ou não tenham acesso ao código da *framework*. Surgem também alguns problemas de integração, quando se tenta combinar duas ou mais *frameworks*. respeita à eficiência, é outro dos pontos negativos da *framework*. As *frameworks* potenciam a extensibilidade fornecendo vários formas ao programador para poder estender a sua aplicação, como por exemplo a criação de subclasses e personalizar as interfaces existentes. Esta extensibilidade faz com que a eficiência da *framework* seja reduzida (Fayad and C. Schmidt, 1997; Lopes, 2008; Taligent, 1993).

2.2.5 *Processo de Aprendizagem*

O processo de aprendizagem de uma *framework* é muito mais difícil, do que o processo de aprendizagem de uma biblioteca de classes, pelo simples facto de que numa *framework*, ao contrário de uma biblioteca, é impossível aprender uma classe de cada vez. As classes de uma *framework* estão implementadas de forma a cooperarem entre si, o que implica que o estudo destas deve ser em simultâneo. Estas classes não implementam toda a lógica da *framework*, é necessário estende-las, o que implica perceber o que pode mudar ou não na *framework* (Johnson, 1997a).

O que facilita bastante este processo de aprendizagem é a documentação disponível. Se a documentação for adequada, consegue agilizar todo o processo de aprendizagem. De acordo com Johnson (1997a) a melhor maneira de obter conhecimento sobre uma *framework* é através de exemplos. É muito mais fácil de aprender através de exemplos, pois os exemplos são mais concretos e facilitam a aprendizagem da *framework* como um todo. Os exemplos resolvem um problema especifico e podem ser estudados de forma a perceber o fluxo de controlo dentro da *framework*. Os exemplos não só mostram como o fluxo de controlo funciona, mas também o que os programadores precisam de fazer para utilizar a *framework*. O autor destaca que idealmente as *frameworks* deveriam conter uma série de exemplos, de várias dificuldades, e estes exemplos deveriam cobrir todas as funcionalidades da *framework*.

Geralmente a documentação existente das *frameworks* é pouco mais do que o código fonte. Idealmente, uma *framework* deveria ter uma documentação bastante completa que deveria explicar qual o seu propósito, como utilizar a *framework* e como a *framework* funciona. O propósito da *framework*, isto é, o domínio que para que foi projetada, deve ser a primeira coisa a ser documentada para que se perceba logo qual o domínio para que foi desenvolvida a *framework*. Seguidamente deve-se descrever como funciona a *framework*. Esta documentação deveria mostrar como desenvolver aplicações com esta *framework*. A maioria dos utilizadores da *framework* querem saber o mínimo possível acerca da

framework, só estão interessados em como fazer, e não querem saber o porquê de se fazer daquela maneira. A melhor documentação será do género de um *"cookbook"*[3] (Johnson, 1992; ParcPlace-Digitalk, 1995). Podemos ver alguns exemplos deste conceito em (ParcPlace-Digitalk, 1995; Reelsen, 2011; Nahavandipoor, 2014; Beazley and Jones, 2013). Esta documentação permite aos novos programadores possam desenvolver as suas primeiras aplicações, e os programadores mais avançados podem utilizá-la para verem soluções para um problema em particular, sem explicar as razões por trás dessas soluções. Os *"cookbooks"* não ajudam muito os programadores mais avançados na *framework*, mas têm grande valor didático para os programadores com menos conhecimento sobre a *framework*. Por ultimo, a documentação deve explicar a arquitetura da *framework*, descrevendo todas as classes, como colaboram, porque colaboram e como podem ser estendidas ou personalizadas. É importante que, com a documentação o programador fique com a visão geral da *framework* bem definida (Johnson, 1992, 1997a; Butler et al., 1998).

A primeira utilização de uma *framework* é sempre uma experiência de aprendizagem. Deve-se desenvolver uma aplicação simples, recorrendo a exemplos, utilizando o *"cookbook"* para perceber como implementar determinadas funcionalidades. Depois de desenvolvida a primeira aplicação toda a documentação fará mais sentido. As *frameworks* são bastante complexas, o que faz com que a curva de aprendizagem seja maior do que outras formas de desenvolvimento, por isso é bastante importante que exista documentação e métodos de aprendizagem bem elaborados para a *framework*, de forma a perceber com o utilizá-la (Johnson, 1997a).

2.3 FRAMEWORK ELEVATION

Elevation é uma *framework* de negócio com ferramentas e métodos para criar rapidamente aplicações de negócio de grande valor, desenvolvidos para durar, utilizando os princípios e tecnologias de design baseado em estado da arte. Foi lançada em 2011 pela PRIMAVERA BSS, como uma plataforma que permite desenvolver os seus produtos para a *Cloud* e *on-premises* em metade do tempo até agora necessário e com um número de problemas muito inferior ao atualmente conseguido (PRIMAVERA, 2011). Tem como seus objetivos, o apoiar o desenvolvimento dos produtos da PRIMAVERA da próxima geração, aumentar a produtividade de desenvolvimento das aplicações e acima de tudo aumentar a qualidade dos produtos. Para uma melhor perceção podemos descrever a *framework* como:

- **Biblioteca de Desenvolvimento:** contém um conjunto de tipos reutilizáveis, que fornecem um vasto leque de funcionalidades standard.
- **Fábrica de Desenvolvimento:** personaliza o *IDE (integrated development environment) Visual Studio* para fornecer um ambiente de desenvolvimento especifico para o domínio.
- **Tecnologia:** integra tecnologias de estado da arte para desenvolvimento de alto nível.

3 Conceito metafórico utilizado para descrever um livro que fornece instruções para algo em particular.

- **Design de aplicação:** define uma arquitetura e design para desenvolver aplicações de negócio orientadas a serviços.
- **Framework de Negócio:** fornece conceitos de negócio reutilizáveis e padrões que aceleram o desenvolvimento de aplicações.
- **Plataforma de aplicação PRIMAVERA:** é a plataforma que suporta as aplicações PRIMAVERA da nova geração.

2.3.1 *Princípios de Design*

A *framework* Elevation assenta num conjunto de princípios de design, que caracterizam a *framework* e todas as aplicações geradas pela *framework*. Estes princípios são:

- *Programação Declarativa:* Quase todo o comportamento da *framework* pode ser configurado, personalizado ou modificado através de ficheiros de configuração XML (*eXtensible Markup Language*)
- *Padrões de Design:* A arquitetura da *framework* implementa os boas práticas e padrões de design conhecidos.
- *Reutilização de Código:* A *framework* reutiliza bibliotecas externas de forma a aumentar a produtividade.
- *Orientado a Serviços:* O design orientado a serviços promove a independência entre componentes.
- *Orientado a Processos:* O núcleo da *framework* é projetado para suportar operações orientadas a processos de negócio e fluxos de trabalho.
- *Extensibilidade:* Os componentes da *framework* foram projetados de forma a serem facilmente estendidos e personalizados.
- *Independência Tecnológica:* Os componentes da *framework* foram desenvolvidos de forma a terem o mínimo de dependências de outras tecnologias.

2.3.2 *Arquitetura*

A *framework* Elevation é um projeto e um sistema, que funciona na Internet, em múltiplas plataformas de clientes, e está implementado na *Cloud*. Melhora o *TCO (total cost of ownership)*. É altamente escalável, isto é, tanto pode ser aplicada no desenvolvimento para uma pequena organização como numa organização de maior dimensão e complexidade. Está baseada em tecnologias de estado da arte.

Reduz a complexidade e a variação do desenvolvimento, fazendo que todos os produtos desenvolvidos através da *framework* tenham um fluxo operacional semelhante. Acima de tudo, uma das principais características da *framework* é que é altamente extensível.

Na figura 5 podemos ver uma visão geral do design da arquitetura da *framework* Elevation. Podemos ver que as aplicações geradas são compostas por camadas, e essas camadas pertencem a grupos diferentes.

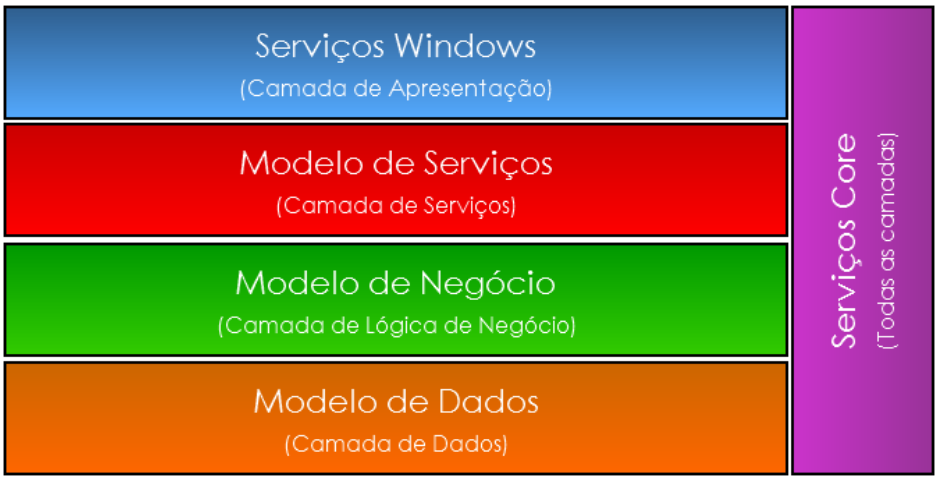


Figura 5.: Camadas da *framework* Elevation

A *framework* Elevation é composta por quatro camadas, em que a camada de apresentação faz a gestão da interface e experiência do utilizador. A camada de serviços, lida com os serviços que são utilizados pela camada de apresentação (cliente) para comunicar com a camada mais abaixo (servidor). Por baixo da camada de apresentação está a camada de lógica de negócio, que é responsável por gerir toda a lógica da aplicação. Por ultimo, temos a camada de dados, que lida com a representação e persistência da dados. Todas estas camadas estão também suportadas na *framework* .NET (Microsoft, 2002).

Estrutura das Aplicações

O ambiente de desenvolvimento, para gerar uma aplicação com a *framework* Elevation, é o *Visual Studio*. As aplicações desenvolvidas com esta *framework* são compostas por dois componentes: a Aplicação e Módulo. O componente "Aplicação" agrega todos os módulos como podemos ver nas figuras 6 e 7. O componente "Módulo" é qualquer projeto que represente componentes lógicos e de negócio do produto da aplicação criada, ver figura 8.

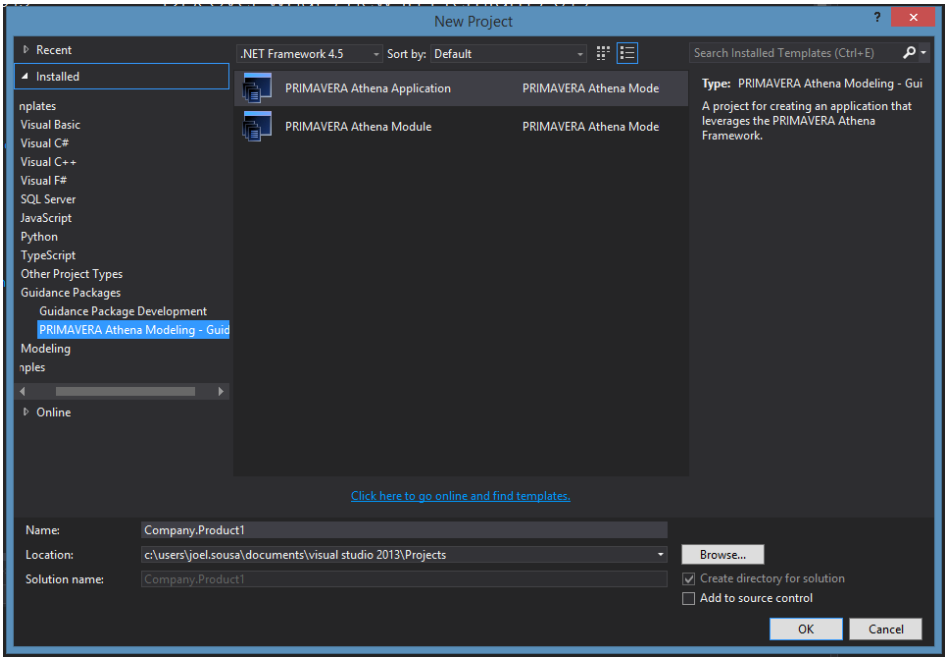


Figura 6.: Componente "Aplicação"

2.3.3 *Desenvolvimento*

O processo de desenvolvimento sobre a *framework* Elevation inicialmente passa por criar um projeto do tipo aplicação, depois de criado podemos então criar ou adicionar módulos à aplicação. O processo de desenvolvimento é composto por duas fases. Na 1.ª fase é desenvolvida a modelação que

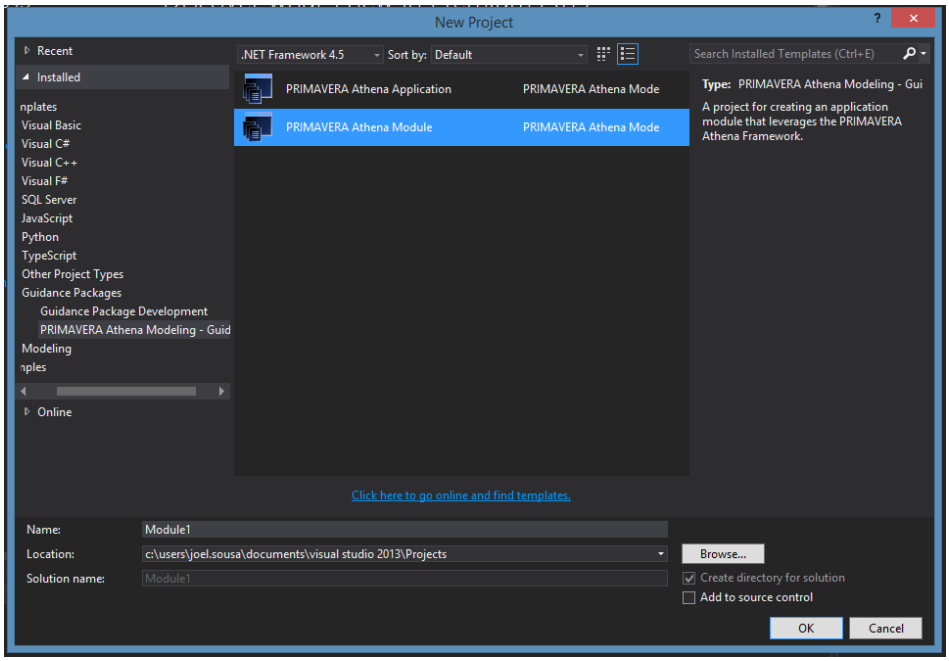


Figura 7.: Componente "Módulo"

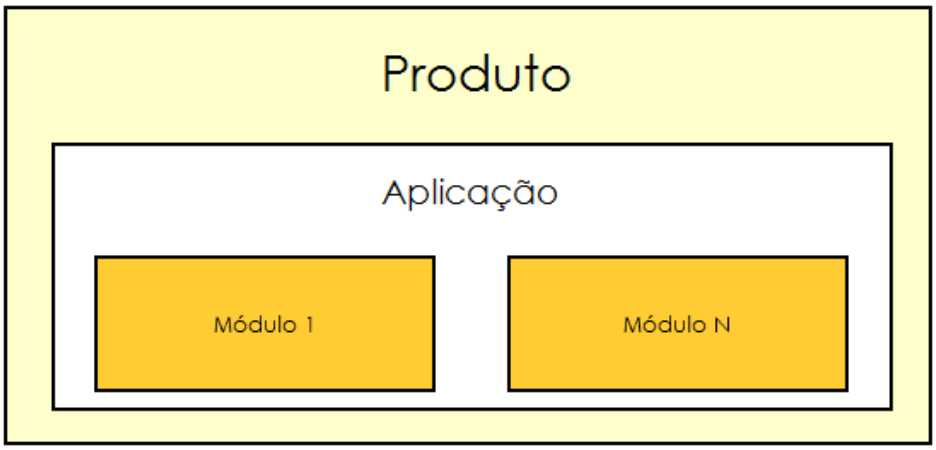


Figura 8.: Produto desenvolvido na *framework* Elevation

irá estruturar a aplicação a desenvolver, esta modelação irá gerar código automático que poderá ser personalizado na segunda fase. Posteriormente na segunda fase é então realizada a personalização do código gerado. É nesta fase que são implementadas as regras de negócio, validações e outros tipos de lógica de negócio. Ao fim da criação do módulo da aplicação são gerados automaticamente vários projetos do *Visual Studio*, estes projetos estão divididos por pastas, ver figura 9

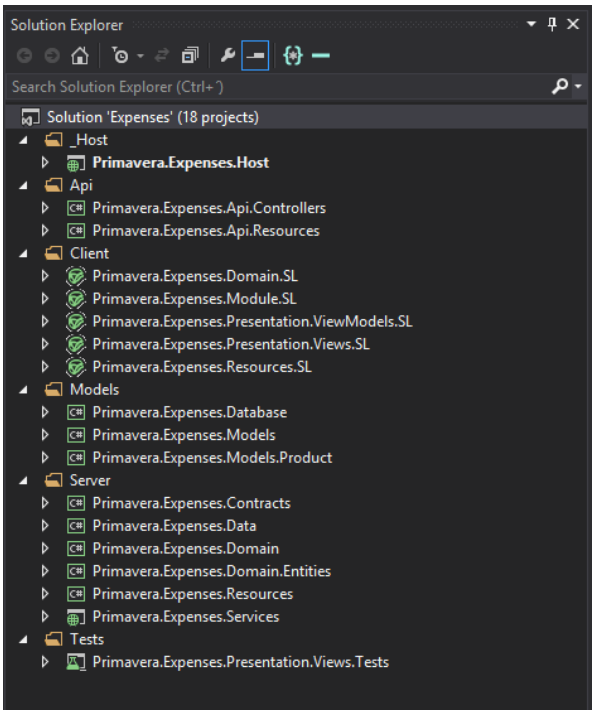


Figura 9.: Estrutura de um Módulo no Visual Studio

Os projetos gerados pela *framework* são:

- "Host" - É composto pelo projeto que que contém as configurações para a execução do módulo.
- "Client" - Agrega os projetos que lidam com a parte de apresentação da aplicação, do lado de cliente.
- "Models" - É composto por dois projetos, um que gera todas as estruturas das bases de dados necessárias para suportar a lógica de negócio, e outro que contém os designers da *framework*.
- "Server" - É um conjunto de projetos que lidam com a lógica do lado do servidor.
- "Tests" - Este módulo em particular, é utilizado para efetuar diversos testes ao módulo.

Modelação

O processo de desenvolvimento começa com o processo de modelação, onde é modelada toda a estrutura da aplicação. Esta modelação é realizada num designer, que é uma extensão do *Visual Studio*, é uma superfície de design gráfica onde são criados objetos a utilizar posteriormente na geração de código automático. A modelação é dividida em 6 componentes: product designer, entities designer, services designer, presentation designer, list designer, menu designer. Estes *designers* serão explicados em detalhe seguidamente, onde será representado todo o fluxo de desenvolvimento da modelação.

1.ª FASE - PRODUCT DESIGNER

Neste designer é onde são definidos todos os módulos[4], submódulos[5] e funcionalidades[6] que irão ser utilizados no produto[7]. Os módulos definidos neste designer podem ser criados de raiz, ou importados de outros produtos já desenvolvidos (figura 10).

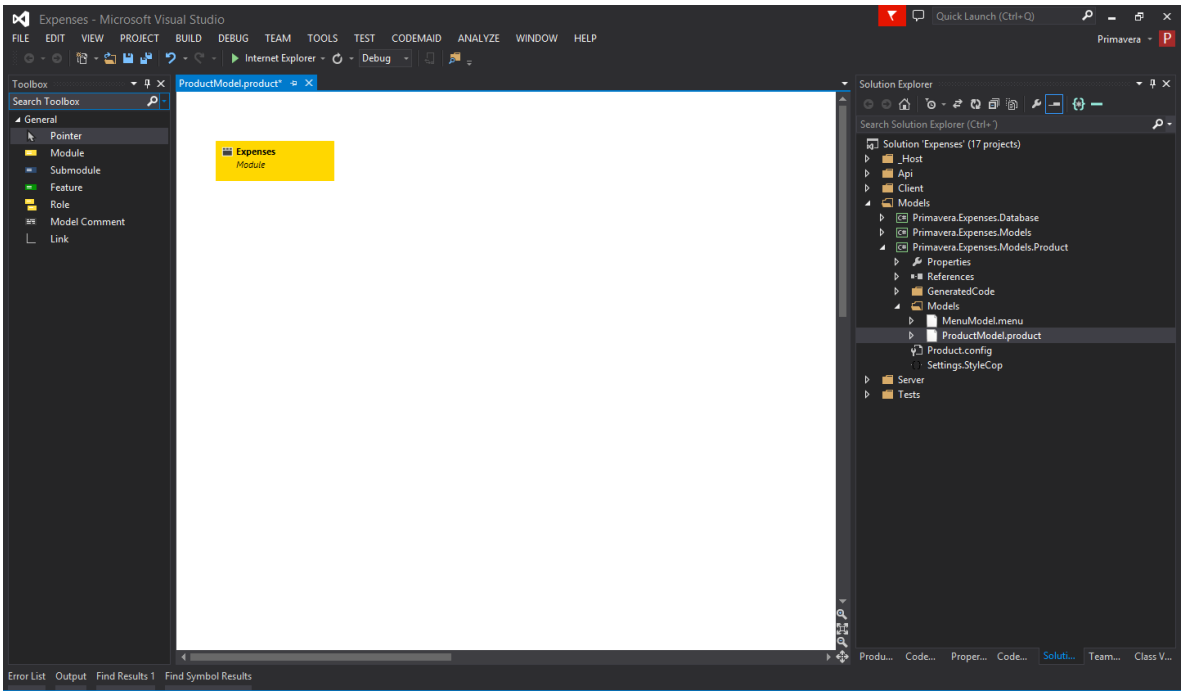


Figura 10.: Product Designer e toolbox no Visual Studio

2.ª fase - ENTITIES DESIGNER

Nesta secção é onde são definidas as entidades que irão fazer parte do produto. São definidas as relações entre as entidades e especificam-se dos campos das entidades (figura 11).

4 É a uma unidade funcional do produto e de desenvolvimento de aplicações (tipicamente cada módulo é desenvolvido por uma equipa autónoma). Um módulo é composto por um ou mais submódulos.
5 Corresponde a uma subdivisão funcional de determinado módulo. Um submódulo tem uma ou mais funcionalidades.
6 São funcionalidades de um determinado submódulo, tipicamente licenciável como uma unidade.
7 Uma aplicação que agrega um conjunto de módulos.

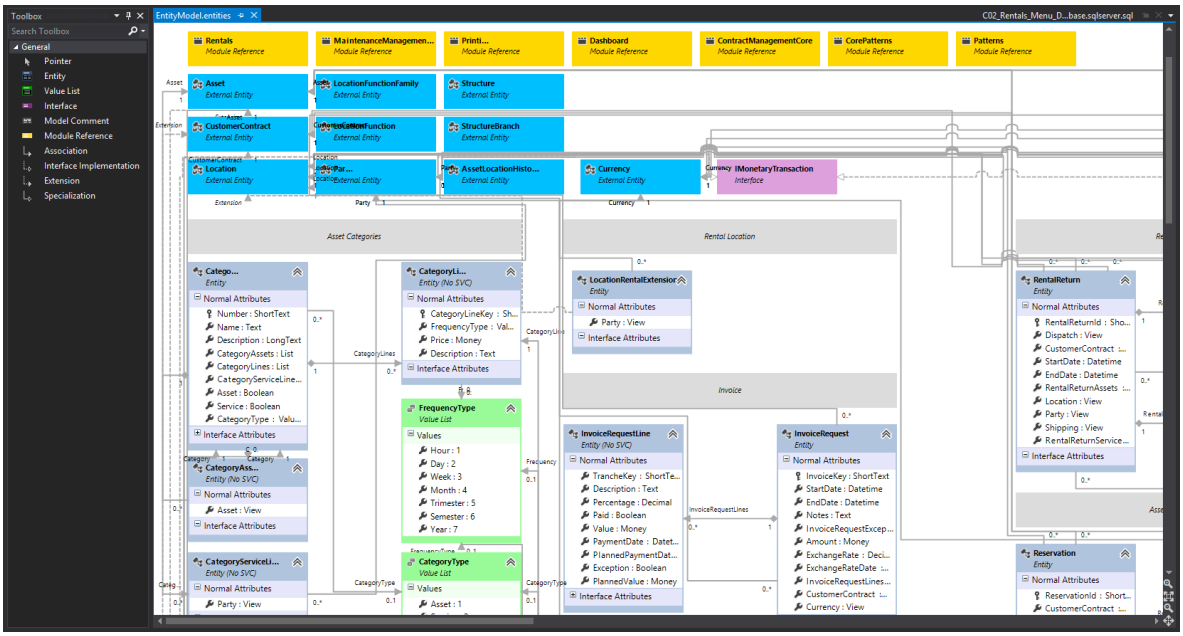


Figura 11.: Entities Designer e toolbox no Visual Studio

3.ª *fase* - LIST DESIGNER

Permite efetuar a modelação das perspetivas de exploração de dados (e construção de listas) de um módulo. Uma perspetiva define uma "estrela" de exploração de dados, relacionando entidades de um ou mais módulos (figura 12).

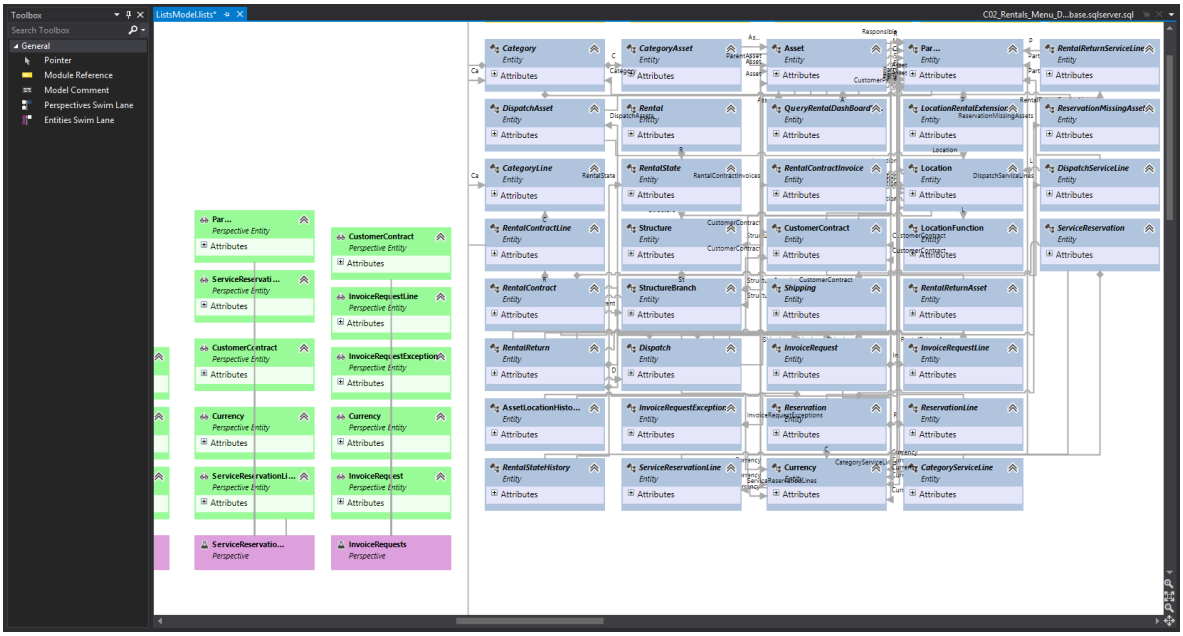


Figura 12.: List Designer e toolbox no Visual Studio

4.ª fase - PRESENTATION DESIGNER

O *Presentation Designer* permite a modelação das *views*[8] das aplicações Elevation. Apesar do meta-modelo deste designer introduzir restrições e orientações implícitas na modelação dessas *views*, o grau de liberdade do analista continua a ser elevado e a permitir que sejam construídas *views* estruturalmente e esteticamente muito diferentes (figura 13).

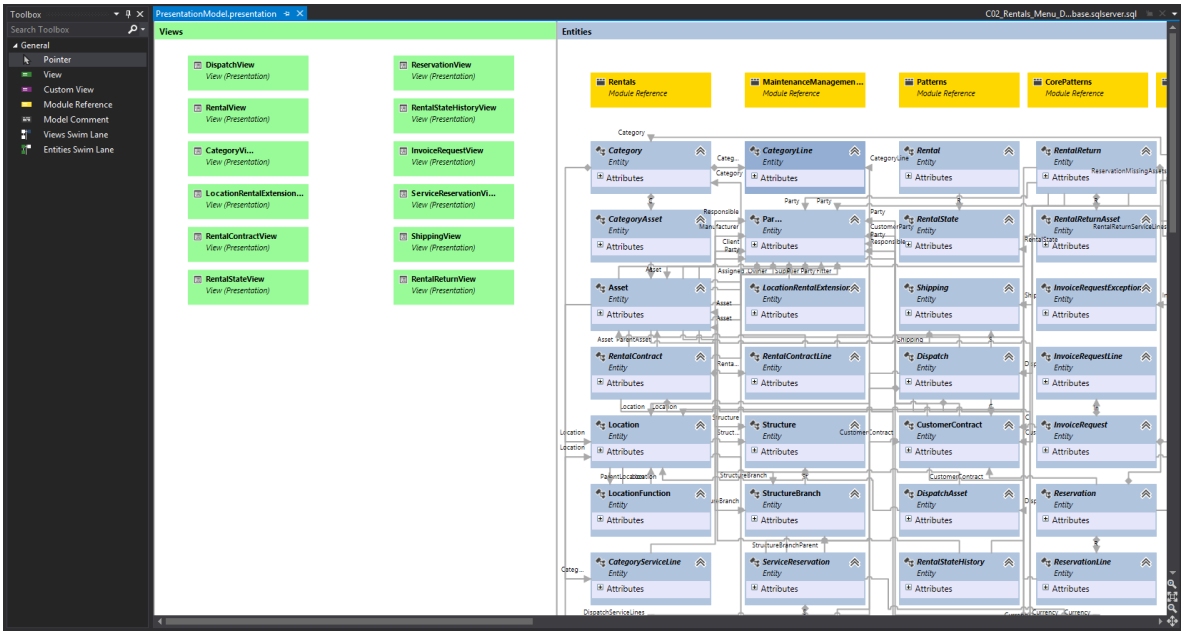


Figura 13.: Presentation Designer e toolbox no Visual Studio

5.ª fase - SERVICES DESIGNER

O *Services Designer* permite efetuar a modelação dos serviços sobre entidades definidas no Entities Designer. Um serviço corresponde a um serviço (Web service) que pode ter as operações *CUD (Create, Update e Delete)* sobre a entidade correspondente e zero ou mais operações específicas, ver figura 14 (figura 14).

8 Corresponde a um *layout* de apresentação de uma entidade modelada seja para visualização ou para impressão. Uma determinada entidade pode ter tantas *views* quantas as necessárias.

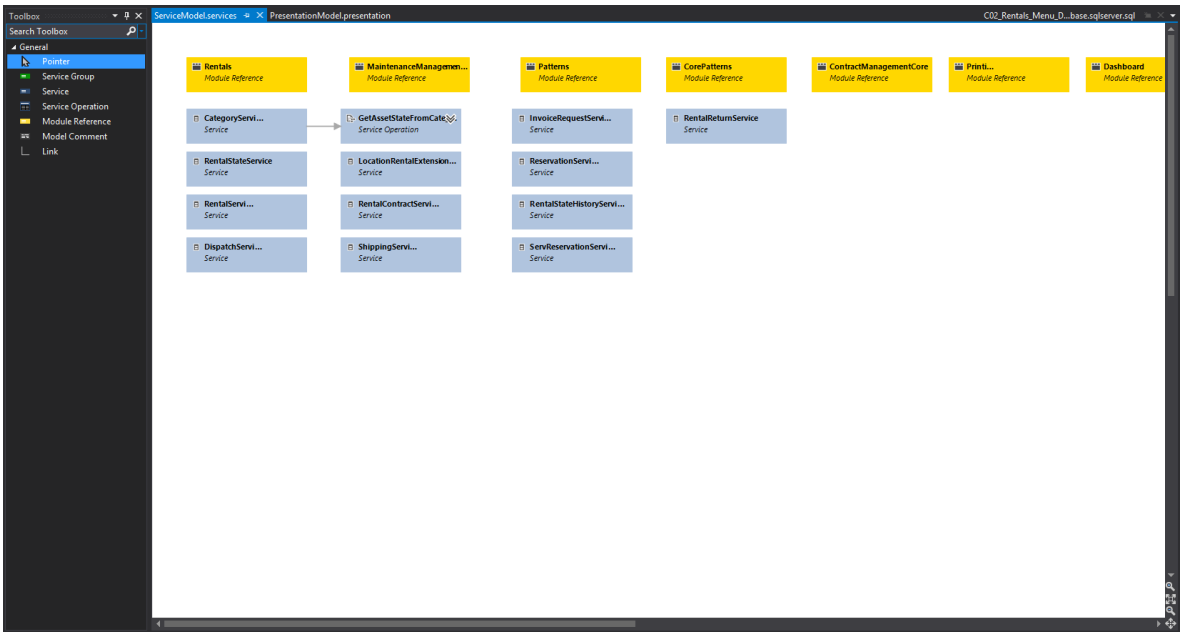


Figura 14.: Services Designer e toolbox no Visual Studio

6.ª fase - MENU DESIGNER

O *Menu Designer* permite a modelação do menu da aplicação a partir das tarefas definidas nos vários módulos. O menu é a estrutura hierárquica de opções (tarefas) que compõem o menu da aplicação (figura 15).

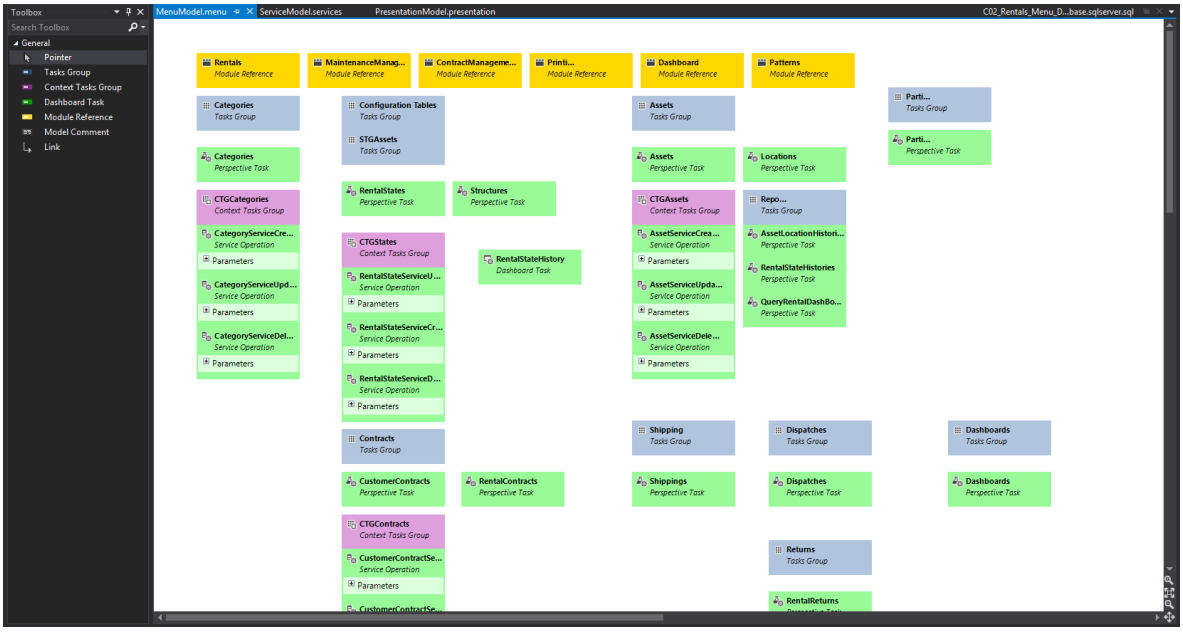


Figura 15.: Menu Designer e toolbox no Visual Studio

Personalização

Através da modelação é possível criar uma aplicação completa em algumas horas sem introduzir uma linha de código, mas essa aplicação não iria fazer mais que mostrar formulários, e listas e permitir gravar dados na base de dados. É então ai que entram os programadores, que têm de programar a lógica de negócio. Esta personalização consiste em estender classes que foram geradas pelo código automático, adicionando novos métodos ou substituir métodos existentes. A linguagem de programação utilizada neste tipo de personalizações é o c# .NET.

É possível produzir lógica de negócio do lado do cliente, ou do lado do servidor, geralmente do lado do cliente é produzida lógica de negócio nas *views*, são efetuadas validações a campos, preenchimento automático de campos entre outras operações que não envolvem comunicação com a base de dados.

Por outro lado, no servidor, é desenvolvida lógica de negócio que faz validações dos campos na base de dados entre outro tipo de operações que requerem informação que apenas está disponível do lado do servidor.

3

ABORDAGEM METODOLÓGICA

Neste capítulo dedicado à abordagem metodológica utilizada na dissertação, apresenta-se a estratégia de pesquisa utilizada na realização de literatura a que se segue a descrição do método de investigação adotado, o estudo de caso, cujos primeiros passos (planeamento, design, preparação, recolhas de dados) são explicados neste capítulo e os últimos passos (análise de dados e conclusões) são considerados nos capítulos seguintes.

3.1 REVISÃO DE LITERATURA

De modo a realizar a revisão de literatura relacionada com o tema desta dissertação, foi realizada a pesquisa em alguns portais com bastante credibilidade nos seus conteúdos científicos, nomeadamente "RepositoriUM" (RUM, 2003), "Web of Knowledge" (WoK), "Scopus" (Scopus, 2004), "Google Scholar" (Google, 2005), "IEEE Xplore Digital Library" (IEEE, 2005) e "ACM Digital Library" (ACM, 1996).

Esta pesquisa foi realizada através de diversas combinações de palavras chave, tais como: "software development framework", "object oriented framework", "software reuse framework", "rad frameworks", "Rapid application development frameworks", "design reuse frameworks". De seguida foram também cruzados várias palavras chave, de forma a ter uma pesquisa mais consistente.

De maneira a filtrar a seleção dos documentos obtidos, foram tidos em consideração aspetos como o titulo, o abstract, o autor e o número de citações dos documentos.

3.2 ESTUDO DE CASO

Para perceber quais as vantagens e desvantagens da utilização da *framework* Elevation é necessário encontrar evidências das melhorias sentidas com a utilização da *framework*, do que os programadores das equipas de desenvolvimento e da consultoria acham da *framework* e do que realmente é importante para eles numa *framework* de desenvolvimento. Relembra-se a questão inicialmente colocada: "**Quais as vantagens e desvantagens da utilização da *framework* Elevation?**". A abordagem metodológica escolhida para a elaboração desta dissertação foi o estudo de caso.

Tabela 2.: Palavras-Chave/*Keywords* utilizadas na pesquisa.

Palavras-Chave	**Keywords**
Frameworks de desenvolvimento de software	Software development framework
Frameworks orientadas a objetos	Object oriented framework
Frameworks reutilização de software	Software reuse framework
Frameworks reutilização de designs	Design reuse frameworks
Reutilização de software	Reutilização de software
Programação orientada a componentes	Component Orienteted Programming
Frameworks modulares	Modular Frameworks
Frameworks de desenvolvimento rápido	Rapid application development frameworks
	RAD frameworks

Esta abordagem metodológica é adequada pois um *case study* é utilizado em diversas ocasiões para contribuir para o nosso conhecimento do individuo, grupo, organização, social, político e fenómenos relacionados. Os dados são recolhidos através de documentação, arquivos, observações diretas, observações de participantes e de artefactos (Yin, 2009). De forma a perceber quais as vantagens e desvantagens da utilização da *framework*, será necessário a utilização de vários métodos de recolha de dados, desde entrevistas e questionários, analise de documentação e de casos práticos, este tipo de métodos enquadra-se nas características desta metodologia.

A execução de um estudo do caso, pode ser dividida em seis passos os quais, excetuando o primeiro, podem ser executados iterativamente (ver figura 16): planeamento, design, preparação, recolha de dados, análise de dados e conclusões (Yin, 2009).

No passo de planeamento decidiu-se qual a questão ou questões de investigação.

De seguida foi elaborado um design da investigação, porque este design serviu de base para toda a investigação. Definiu-se que dados são relevantes, que dados se deviam recolher e como os analisar.

As técnicas de recolha de dados que são utilizadas neste estudo incluem: análise de documentação para perceber melhor as vantagens e desvantagens da *framework*. Entrevistas e questionários de forma a ser possível perceber qual a visão dos programadores sobre a *framework*, e perceber quais as vantagens e desvantagens que cada programador identifica.

Por último foram analisados os dados obtidos de forma a formalizar uma resposta à questão proposta nesta dissertação

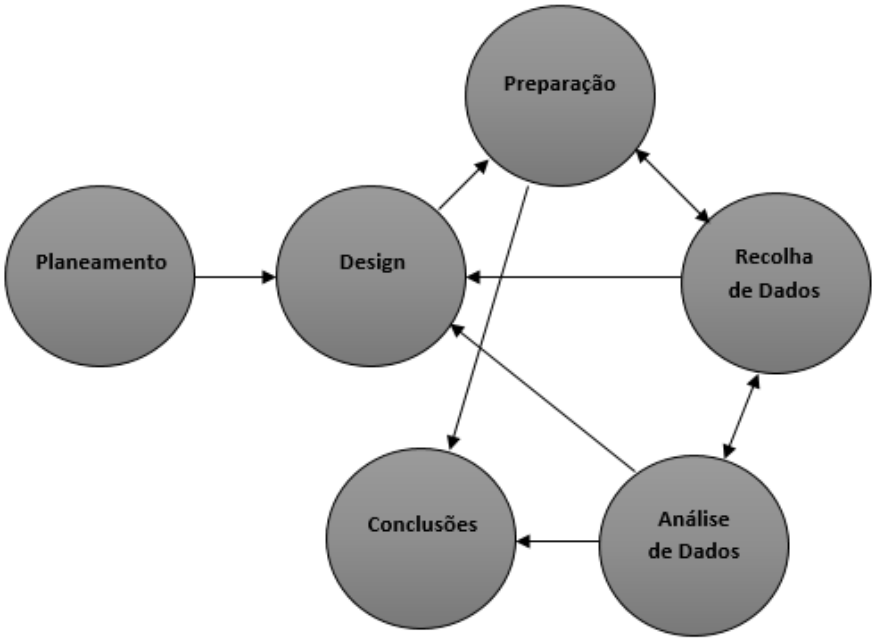


Figura 16.: Processo de investigação em estudo de caso
(Fonte:(Yin, 2009) (traduzido pelo autor))

3.3 PLANEAMENTO E DESIGN

De forma a definir o trabalho de investigação desta dissertação, foi realizada uma reunião na sede da Primavera BSS com o orientador na universidade e o supervisor na empresa.

Foram discutidas várias questões que poderiam ser relevantes para investigação, tendo ficado definida a seguinte questão: "Quais os vantagens e desvantagens da utilização da *framework* Elevation?"

Ficou acordado que no âmbito da recolha de dados, seriam realizados questionários aos colaboradores da PRIMAVERA BSS que trabalham diretamente com a *framework* Elevation, o objeto de estudo nesta investigação.

Ficou ainda planeada, foi a realização de um protótipo para uma melhor compreensão da *framework*. , permitindo a recolha de variáveis para uma posterior análise e enriquecendo o investigador com alguma experiência na *framework* que lhe facilitasse a elaboração de questões.

3.4 PREPARAÇÃO E RECOLHA DE DADOS

De forma a preparar o estudo, os colaboradores da PRIMAVERA BSS foram informados que iriam ser inquiridos quanto às vantagens e desvantagens da *framework* Elevation. Foi também pedido acesso à documentação relevante ao desenvolvimento com a *framework* Elevation.

As técnicas de recolha de dados que foram utilizadas neste estudo foram: análise de documentação, onde foi possível perceber melhor a *framework* Elevation e perceber as práticas de desenvolvimento das equipas de desenvolvimento; entrevistas onde foi possível recolher informação a utilizar na elaboração de questionários. Os questionários permitiram obter uma a visão dos colaboradores da PRIMAVERA BSS sobre a *framework* e identificar as vantagens e desvantagens na sua utilização.

3.4.1 *Preocupações Éticas*

Quando é realizado um estudo de caso, devemos ter em atenção algumas considerações éticas. Três conceitos éticos basilares de um estudo de caso são: confidencialidade, negociação e acessibilidade. Nesta dissertação tais conceitos resultam nas seguintes considerações:

- O objetivo do estudo e o seu envolvente será bem explicito.
- Será pedida autorização para consultar documentos, e não será realizada nenhuma cópia sem a autorização necessária.
- Haverá um consenso das perguntas às quais o entrevistado quer responder.
- As entrevistas e questionários serão conduzidos de forma confidencial.

3.4.2 *Fases da Recolha de Dados*

A análise de dados foi dividida em várias fases, de forma a conseguir obter dados das mais variadas fontes. Foi necessário dividir o estudo em cinco fases.

1ª Fase - *Revisão de literatura* - Foi elaborada uma pesquisa bibliográfica, onde foram identificadas e estudadas fontes de informação relevantes para o estudo.

2ª Fase - *Seleção de documentação* - Envolveu a realização de um estudo e análise da documentação, selecionada de entre a fornecida pela PRIMAVERA BSS, de modo a perceber o funcionamento dos desenvolvimentos sobre a *framework*.

3ª Fase - *Avaliação de documentação* - Foi avaliada toda a documentação fornecida, e foi filtrada toda a informação pertinente para o estudo.

4ª Fase - *Realização de entrevistas* - Foram realizadas algumas entrevistas aos elementos das equipas de desenvolvimento da PRIMAVERA BSS, onde foi possível identificar variáveis que seriam avaliadas no questionário final.

5ª Fase - *Realização de questionários na PRIMAVERA BSS* - Foram realizados vários questionários aos elementos das equipas de desenvolvimento da PRIMAVERA BSS, neste caso tentando envolver

o maior número possível. Este tipo de abordagem contou com a colaboração de elementos com papéis distintos.

6ª Fase - *Avaliação dos resultados* - Foram analisadas estatisticamente as respostas obtidas, preparando gráficos para serem discutidos na fase de análise de resultados.

3.4.3 *Entrevistas*

As entrevistas foram marcadas previamente e realizadas na sede da PRIMAVERA BSS. Foram entrevistados quatro elementos. Três *developers* das equipas de desenvolvimento que utilizam a *framework* Elevation, sendo dois deles *developers* seniores e outro júnior e um *Product Owner*. Ao entrevistar estes elementos foi possível obter uma visão mais técnica, de um *developer* com mais e outro com menos experiência, conseguindo obter respostas que só quem utiliza a *framework* diariamente as consegue responder. Entrevistando o *product owner*, foi possível obter respostas de alguém que planeia o projeto, e tem que definir qual a visão e o caminho que o produto deve levar.

Antes de se realizarem as entrevistas, foi acordado que as entrevistas seriam anónimas e que os seus nomes não seriam utilizados nas citações retiradas das entrevistas.

A entrevista foi realizada de acordo com um guião previamente definido. As perguntas deste guião foram elaboradas com o objetivo de fundamentar a resposta à questão de investigação.

O guião da entrevista foi composto pelas seguintes questões:

1. Quais as características que mais valoriza numa *framework* de desenvolvimento?
2. Quais as vantagens da utilização da *framework* Elevation?
3. Quais as desvantagens da utilização de *framework* Elevation?
4. Já houve funcionalidades que tiveram de ser repensadas e implementadas talvez com menor performance devido as limitações da arquitetura da *framework*?
5. Acha que a utilização da *framework* encurta o tempo de desenvolvimento, permitindo assim entregar soluções com mais qualidade em menor tempo?
6. Como classifica a documentação existente sobre a *framework* ELEVATION? Existe a documentação necessária? Qual a estrutura que a documentação deveria ter?
7. Como classifica as ferramentas de apoio à *framework* existentes? Se existirem, indique algumas.
8. Teve formação? Considera que existe formação adequada?
9. Qual o tempo de aprendizagem da framework?
10. Qual o seu nível de satisfação para com a framework?

11. Defina 5 vantagens, ordenada da mais importante, para a menos importante que vê com o uso da framework Elevation

12. Defina 5 desvantagens, ordenada da mais importante, para a menos importante que vê com o uso da framework Elevation

13. Sugestões para o futuro da framework

3.4.4 *Variáveis*

Através das entrevistas foi possível identificar um conjunto de varáveis, que foram ser posteriormente utilizadas num questionário final, distribuído por todos os colaboradores que trabalham diretamente com a *framework* Elevation, de modo a serem validados e então analisar as respostas obtidas.

Podemos agrupar as varáveis identificadas em três grupos: as características importantes numa *framework* de desenvolvimento, vantagens da *framework* Elevation e desvantagens da *framework* Elevation.

Características importantes numa framework de desenvolvimento

Tabela 3.: Características importantes numa framework de desenvolvimento

Variável	**Citação**
Design das aplicações	"Devem produzir aplicações com um design bem estruturado."
Formação	"A formação inicial é muito importante, de forma a colocar o programador a produzir de forma mais rápida o mais rápido possível."
Modularização	"Deve ter uma arquitetura modular, de forma a ser possível utilizar esses módulos individualmente, tornando a framework mais leve." "... se a framework estiver dividida em módulos é uma mais valia"
Documentação	"Valorizo bastante a documentação, principalmente nas frameworks desenvolvidas internamente, visto que não é possível procurar soluções na internet."
Manutenção	"Convém que a manutenção seja fácil".
Uniformização	"Faz com que os programadores trabalhem todos da mesma forma, tornando assim o desenvolvimento mais uniforme."
Segurança	"É importante que crie aplicações seguras, retirando algum trabalho aos programadores".

continua na próxima página

Tabela 3 – continuação da pagina anterior

Característica	Citação
Compatibilidade	"deve ser compativel em vários sistemas operativos"
Rapidez de Implementação	"Definição de uma estrutura e de uma base para um desenvolvimento rápido e com resultados"
Curva de Aprendizagem	"O tempo de aprendizagem deve ser reduzido, porque enquanto se aprende a produção é menor"
Extensibilidade	"Uma framework deve ser extensível de forma a podermos integrar produtos diferentes"
Exemplos Existentes	"Os exemplos na documentação são bastante importantes para quem começa"
Eficiência	"Deve ser o mais eficiente possível, de forma a reduzir o tempo desperdiçado a programar."
Estruturação	"A estruturação da framework é importante, e ajuda na aprendizagem da mesma."
Reutilização	"Deve ser possível reutilizar o maior número de componentes possíveis."
Pré-Customização	"Disponibilize meios acessíveis de customização e parametrização".
Escalabilidade	"Devem ser altamente escaláveis visto que com o decorrer do tempo a exigência das aplicações é maior"
Requisitos de hardware	"Não convém ser muito pesada"
Performance	"A performance também é importante, visto que o objetivo é ajudar o programador e não dificultar o desenvolvimento."
Abstração	"Abstração e simplificação de código"; "Valorizo a abstração de tarefas habituais no desenvolvimento de software."

Vantagens das framework Elevation

Tabela 4.: Quais as vantagens da utilização da framework Elevation

Variável	Citação
Facilita o design das aplicações	"... é mais fácil fazer o design de aplicação"

continua na próxima página

Tabela 4 – continuação da pagina anterior

Vantagem	Citação
Aumenta a rentabilidade	"... as equipas tornam-se mais rentáveis para a empresa"
Reduz o investimento	"No ponto de vista de produtos finais reduz o investimento necessário."
Aumenta a performance das aplicações	"... de uma maneira geral aumenta a performance das aplicações"
Aumenta a segurança das aplicações	"As aplicações desenvolvidas com a *framework* Elevation são bastante seguras, pois houve muita preocupação nesse aspeto durante o desenvolvimento da *framework*."
Reduz o custo de licenciamento	"Como as produtos são desenvolvidas mais rápido tem um impacto direto no licenciamento das mesmas."
Cria regras e boas práticas nas equipas	"Ajuda a que a qualidade do código seja melhor, o *code analysis* e *source analysis* são ferramentas muito úteis, pois formatam as pessoas a escrever código muito semelhante, para evitar que cada um tenha o seu estilo de programar e torna o desenvolvimento bastante consistente."
Facilita o deployment	"O deployment para on premises é rápido"
Redução de recursos humanos	"... dependendo dos produtos pode reduzir o pessoal necessário"
Facilita os ciclos de teste	"os padrões ajudam facilita o trabalho dos testers"
Cria padrões nas linhas das aplicações	"Uso de padrões semelhantes em diferentes produtos" "... as views e funcionalidades são todas semelhantes"
Reduz o time-to-market	"Tenho a experiencia de fazer uma aplicação ainda com a versão anterior da framework que foi feito num mês e que o produto ainda está a ser vendido"

continua na próxima página

Tabela 4 – continuação da pagina anterior

Vantagem	Citação
Aumenta a produtividade	"Os modelos garantem que mesmo tendo migrações de tecnologia temos sempre uma parte do desenvolvimento garantido, tornando as equipas mais produtivas nestes casos"
Facilita as decisões de arquitetura	"a arquitetura base já é gerada pela *framework*, isso facilita-nos algumas decisões"
Facilita o desenvolvimento dos parceiros	"os parceiros deveriam utiliza-la de forma a facilitar a produção de software específicos"
Diminui o tempo de desenvolvimento	"A modelação em conjunto com a geração de código reduz os tempos de desenvolvimento"

Desvantagens da framework Elevation

Tabela 5.: Quais as desvantagens da utilização de framework Elevation

Desvantagem	Citação
Dificuldade de suporte	"o suporte existente é chegar ao lado da equipa que desenvolve a *framework* e tirar dúvidas estando dependendo da dificuldade dos mesmos."
Complexidade	"A *framework* é muito complexa para quem começa a trabalhar. O primeiro contacto pode ser um pouco assustador" "... há conceitos que são muito complexos, foram feitos de forma muito complexa"
Curva de aprendizagem	"Período de aprendizagem e configuração inicial é bastante difícil." "Curva de aprendizagem lenta e difícil."
Dificuldade de manutenção	"... e as manutenções às vezes são difíceis, porque temos de descobrir se o problema é nosso produto ou da *framework* em sí."
Limitações da *framework*	"... a framework tem as suas limitações"

continua na próxima página

Tabela 5 – continuação da pagina anterior

Variável	Citação
Custos de implementação	"Implementar pequenas funcionalidades ás vezes demora demasiado tempo, e isso aumenta os custos do projeto."
Requisitos de sistema das aplicações	"As aplicações precisam de um browser compatível com silverlight"
Licenciamento dos produtos	"Como a produção é maior conseguimos reduzir o custo de licenciamento dos produtos."
Tecnologias utilizadas	"Dependência das tecnologias em que se assenta..."
Dificuldades de alteração do core da *framework*	"Alterar o core da framework é difícil, porque se criarmos algum bug numa release de *framework* todas as aplicações terão o mesmo problema."
Falta de formação	"Apenas obtive uma pequena formação on-the-job, este tipo de formação não é suficiente."
Rigidez da *framework* em casos mais específicos	"... por vezes, alterações muito simples conseguem demorar muito tempo a se desenvolver devido à rigidez da própria estrutura."
Documentação limitada	"A documentação existente não é suficiente e é difícil de consultar."
Custo de introdução de novas funcionalidades	"Às vezes implementar novas funcionalidades é caro, porque implica alterações em vários produtos."

3.4.5 *Questionário Final*

Após as entrevistas, foi realizado um questionário a vários elementos de várias equipas de desenvolvimento da PRIMAVERA BSS. Este questionário foi direcionado a todos as pessoas que têm experiência na utilização da *framework* Elevation, sendo estes elementos, *testers* ou *developers*, com vários níveis de experiência, tanto a nível profissional, como na utilização da *framework*.

As respostas obtidas foram de grande importância, pois com base nestas respostas, e em conjunto com as entrevistas foi possível realçar um conjunto de vantagens e desvantagens que os *developers* e *testers* identificam quando utilizam a *framework*.

Universo

O universo desta pesquisa contemplou elementos de várias equipas de desenvolvimento da PRIMAVERA BSS, que trabalham, ou já trabalharam, com a *framework* Elevation.

Deste universo foi possível inquirir/entrevistar um total de 4 equipas.

- Entreprise Asset Management (EAM) - É a equipa que desenvolve um *software* de manutenção, utilizando a *framework* Elevation.
- Inovação e Novas Tecnologias (INT) - Esta equipa é responsável pelo desenvolvimento da *framework* Elevation.
- Business Suite (BS) - É a equipa responsável pelo desenvolvimento de uma aplicação de gestão, também esta desenvolvida utilizando a *framework* Elevation.
- Business Analytics (BA) - Esta equipa desenvolve a aplicação de análise de dados, utilizando a *framework* Elevation.

Perfis dos Inquiridos

Foram identificados três perfis essenciais, para o processo de desenvolvimento de *software* utilizando a *framework*. Estes perfis identificados são essenciais para obter respostas às questões sobre as dificuldades e benefícios no desenvolvimento de *software* utilizando a *framework* Elevation.

Developer - São os elementos que fazem desenvolvimento de software utilizando a *framework* e também desenvolvem a *framework* em si.

Product Owner - É o principal *stakeholder* nos desenvolvimentos dos projetos, sendo ele que interage diretamente com o cliente e define os requisitos dos projetos. O Product Owner tem de ter bastante conhecimento sobre a *framework*, pois tem de conhecer quais as suas limitações e mais valias de forma a realizar o projeto com o máximo de qualidade possível e dentro dos parâmetros definidos pelo cliente.

Tester - O *tester* também tem um papel importante no processo de desenvolvimento, pois tem de conseguir testar não só o *software* de desenvolvimento, como também validar atualizações constantes à *framework*. Isto exige que os *testers* tenham bastante conhecimento sobre a *framework*, de forma a conseguirem perceber qual o impacto que essas atualizações têm nos outros componentes da *framework* e aplicações desenvolvidas.

Questionário

De seguida é possível observar as questões realizadas no questionário final, agrupadas pelas seguintes áreas:

Caracterização do respondente

- Qual a sua equipa?
- Qual a sua função na empresa/equipa?
- Quantos anos tem de experiência profissional?

Experiência com *frameworks*

- Já trabalhou com a *framework* Elevation?
- Já trabalhou com outro tipo de *framework* de desenvolvimento?

Importância das *frameworks*

- Classifique a importância dos seguintes características de uma *framework* para quem as utiliza.
- Concorda com os seguintes vantagens da utilização da *framework* Elevation?
- Concorda com os seguintes desvantagens da utilização da *framework* Elevation?

Apoio à utilização da Elevation

- Como classifica a documentação existente sobre a framework Elevation?
- Considera que existe formação adequada?

Apreciação da Elevation

- Qual o tempo de aprendizagem da *framework* Elevation?
- Qual o seu nível de satisfação para com a *framework* Elevation?
- Quais as vantagens que sente com a utilização da *framework* Elevation?
- Defina 5 vantagens ordenada da mais importante para a menos importante que vê com o uso da *framework*.
- Quais as desvantagens que sente com a utilização da *framework* Elevation? Defina 5 vantagens ordenada da mais importante para a menos importante que vê com o uso da *framework*.
- Se pudesse realizar algumas alterações na *framework* Elevation o que sugeria?

3.4.6 *Desenvolvimento em Elevation*

Durante a escrita desta dissertação o investigador efetuou desenvolvimentos na *framework* Elevation, fazendo parte da equipa do *Enterprise Asset Management* onde conseguiu obter experiência e aptidões técnicas que contribuíram muito para o desenvolvimento desta dissertação.

Durante estes desenvolvimentos, o investigador conseguiu abordar várias características da *framework*, tais como, a sua extensibilidade, desenvolvimentos do lado do cliente, desenvolvimentos do lado do servidor e criação de *scripts* SQL para casos específicos. Desta forma, foi possível constatar também quais as vantagens e desvantagens que foi sentindo ao longo dos desenvolvimentos, assim como validar os dados recolhidos através de outras formas.

Sendo membro integrante da equipa do EAM, foi possível passar por todo o processo que um programador menos experiente passa quando começa a trabalhar com a *framework*. Este tipo de experiência teve grande impacto na altura de selecionar e identificar vantagens e desvantagens para a criação do questionário.

As principais vantagens que o investigador conseguiu identificar foram:

- **Redução do tempo de criação da estrutura base das aplicações** - É possível criar uma aplicação base sem grandes regras de negócio em poucas horas.
- **A modelação facilita o desenvolvimento e aumenta a personalização das aplicações** - através da modelação é possivel criar uma aplicação base sem regras de negócio complexas em poucas horas. Através da modelação é possivel criar as entidades que devem ser geradas, *views* (*front end*) a listagem das entidades e todas as conexões à base de dados.
- **O *front-end* das aplicações bem organizada e com um design bem estruturado** - o front-end gerado através da modelação é bastante apelativo, simples e bem estruturado.
- **Extensibilidade** - É possível importar módulos de outras aplicações já desenvolvidas de modo a reutilizar funcionalidades já desenvolvidas noutros produtos.

As principais desvantagens que o investigador conseguiu identificar foram:

- **Complexidade** - foi um grande impacto entrar em contacto com uma *framework* tão complexa como o Elevation, com bastantes projetos dependentes uns dos outros. Perceber o fluxo de criação de uma aplicação é também bastante complexo.
- **Falta de documentação** - A falta de documentação é um grande ponto negativo, principalmente para quem começa. Existe um pequeno *walkthrough* que explica o básico de criação de uma aplicação através de modelação que, no entanto, é bastante incompleto e insuficiente para facultar as bases de desenvolvimento da aplicação. Existem outros documentos que explicam algumas funcionalidades, mas alguns estão obsoletos e/ou contêm demasiada informação técnica que torna difícil a absorção do conhecimento.

- **Falta de formação** - A formação prestada foi apenas uma sessão de uma hora, onde um programador experiente efetua um *walkthrough* simples sobre o processo de desenvolvimento. Todo o resto do processo de aprendizagem exigiu que o investigador fosse perguntando e expondo dúvidas aos programadores mais experientes.
- **A *framework* exige bastantes recursos da máquina de desenvolvimento** - a *framework* Elevation exige um mínimo de 8GB de RAM na máquina de desenvolvimento, e um CPU competente.
- **O front-end pouco eficiente em termos de performance** - a tecnologia Silverlight faz com que os formulários fiquem um pouco lentos quando se trata do carregamento de grandes quantidades de dados.

4

ANÁLISE DE DADOS

Neste capítulo o investigador apresenta os dados obtidos com o questionário na PRIMAVERA BSS e consequente análise dos mesmos.

4.1 QUESTIONÁRIO AOS COLABORADORES DA PRIMAVERA BSS

Foi possível obter uma amostra de 28 respostas referentes a cinco equipas distintas. As equipas envolvidas foram:

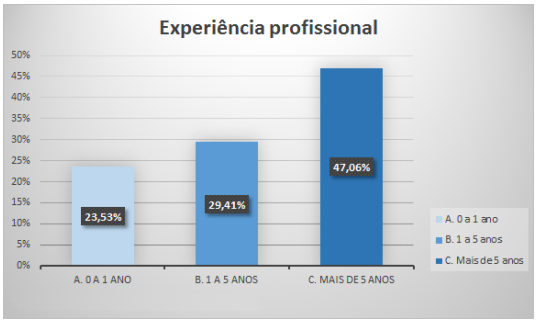
- Entreprise Asset Management (EAM)
- Inovação e Novas Tecnologias(INT)
- Business Suite (BS)
- Business Analytics (BA)

4.1.1 *Experiência profissional e com frameworks de desenvolvimento*

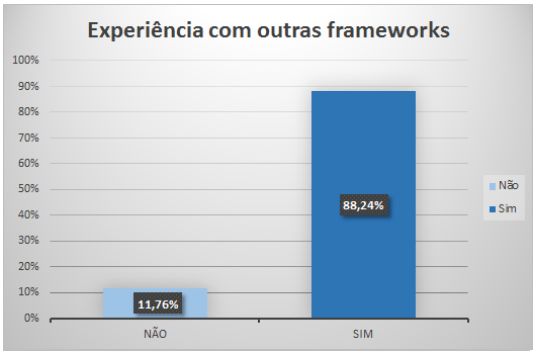
Na figura 17a pela resposta à pergunta: "Quantos anos tem de experiência profissional?", constata-se que a maioria dos inquiridos tem mais de 5 anos de experiência profissional.

Na resposta à questão: "Já trabalhou com outro tipo de *framework*?", verifica-se que 88% dos inquiridos já utilizaram outra *framework* para além da Elevation (figura 17a).

Em suma, podemos verificar que as equipas de desenvolvimento da PRIMAVERA BSS que trabalham com a *framework* são maioritariamente constituídas por colaboradores já com alguma experiência profissional e que grande parte já trabalhou com outras *frameworks* de desenvolvimento. Esta experiência com *frameworks* terá contribuído para uma opinião mais bem fundamentada, na apreciação das vantagens e desvantagens da *framework* Elevation.



(a) Experiência Profissional



(b) Experiência com outras frameworks

Figura 17.: Análise de competências

4.1.2 *Características importantes de uma framework*

A questão: "Classifique a importância das seguintes características de uma *framework* de desenvolvimento?", que questão tem como objetivo, perceber qual a importância dada pelos inquiridos às características previamente identificadas, ordenada do mais para o menos importante. Permite verificar que as cinco características mais importantes são: a escalabilidade, performance, extensibilidade, reutilização de código e eficiência. Já as características menos importantes são: requisitos de hardware, pré-customização, abstração, compatibilidade e estruturação (figura 18).

Pela resposta a esta questão conseguimos perceber quais os fatores que os inquiridos acharam importantes numa *framework*, os quais podem ser confrontados com os benefícios encontrados para melhor perceber se as principais vantagens esperadas duma *framework* estão a ser correspondidas pela *framework* Elevation.

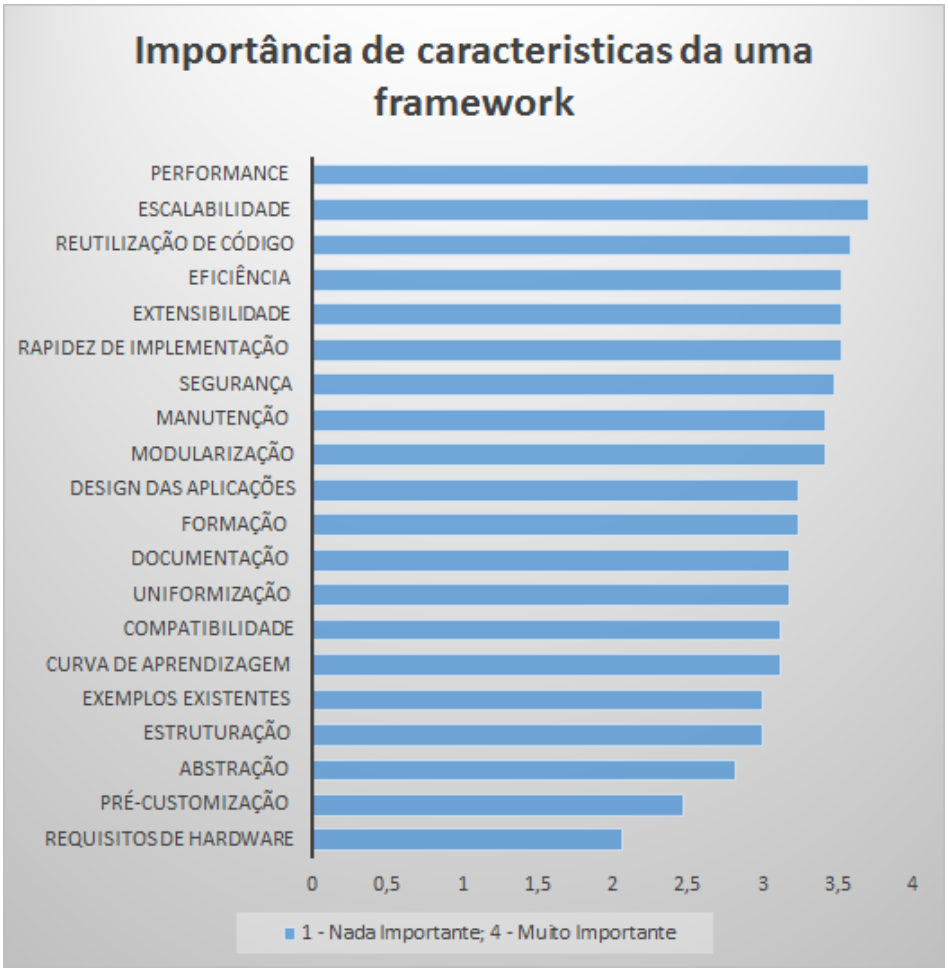


Figura 18.: Características importantes na utilização da frameworks

Podemos constatar que as principais características que têm mais relevância são fatores técnicos e orientados ao aumento da qualidade dos aplicações e diminuição de tempos de desenvolvimento. Fatores como a performance, escalabilidade, reutilização de código, eficiência e extensibilidade são tudo fatores que deverão permitir aumentar a performance das equipas de desenvolvimento.

4.1.3 *Vantagens de utilização da framework Elevation*

A questão: "Concorda com os seguintes benefícios da utilização da framework Elevation?", para cada uma das vantagens previamente identificadas permite verificar para cada uma qual o nível de concordância que lhe é atribuido na escala "Concordo totalmente"... "Discordo totalmente" (Figura 19).

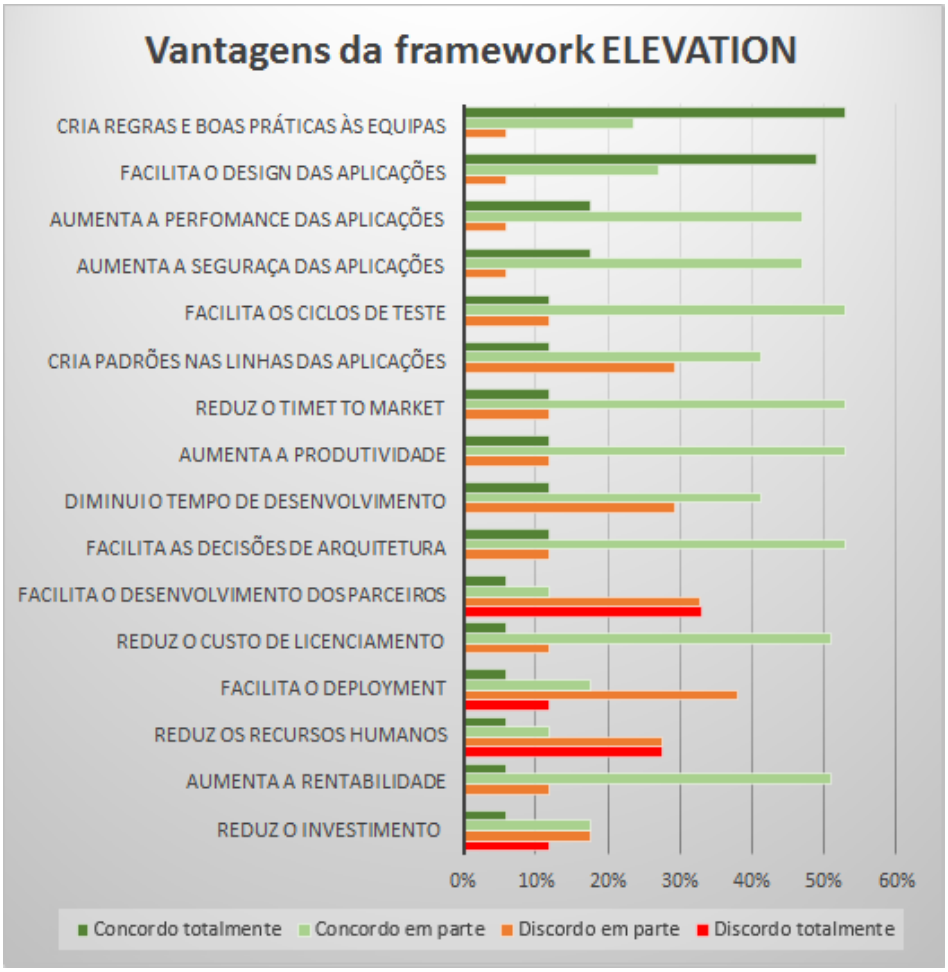


Figura 19.: Vantagens de utilização da framework Elevation

A figura 20 apresenta de uma forma mais clara os dados estruturados em função do nível de concordância.

Os dados obtidos com a realização de questionários corroboram algumas vantagens que já tinham sido evidenciadas pelo investigador durante a revisão de literatura (consultar secção 2.2.4 onde foram descritas os benefícios e os custos esperados de uma *framework*). Fazendo agora uma análise aos dados obtidos destaca-se que Fayad and C. Schmidt (1997) já haviam definido que os principais benefícios que uma *framework* fornece a um *developer* são: a modularidade, reutilização e extensibilidade. Estas vantagens foram identificadas na *framework* Elevation, à excepção da modularidade que não foi referenciada diretamente. No entanto, uma vez que a arquitetura da framework Elevation é divida por módulos, este benefício também se enquadra.

Lopes (2008) realçou que as *framework* trazem vantagens porque permitem não só reutilizar o código mas também fazer uma reutilização nível arquitetura. Isto permite criar menos bugs nas aplicações, visto que o código que é necessário escrever é reduzido, o que por sua vez facilita a

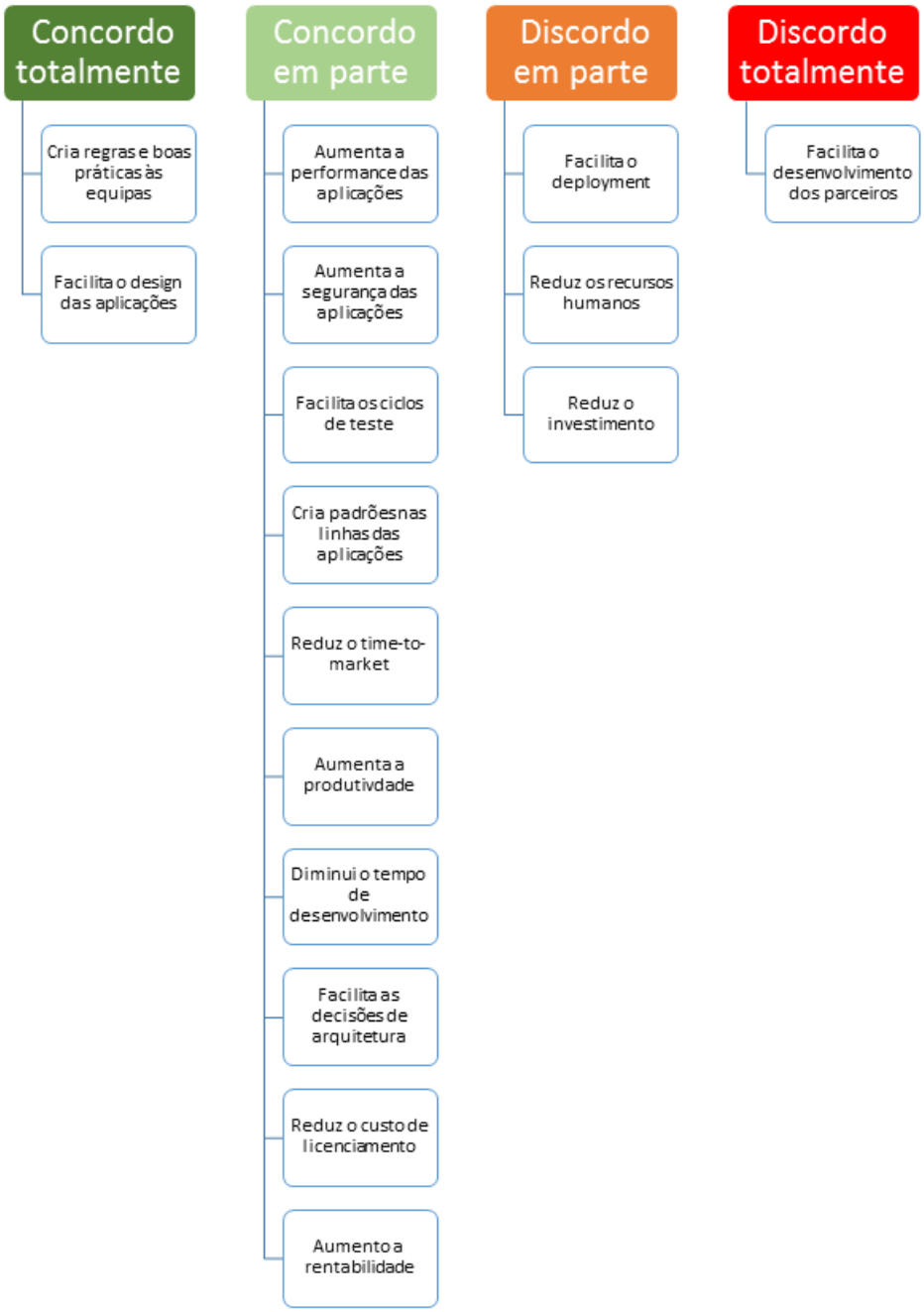


Figura 20.: Vantagens de utilização da framework Elevation

manutenção, já que o código gerado já foi testado inúmeras vezes algo que é suportado pelos dados obtidos.

As vantagens que a *framework* Elevation proporciona estão relacionadas entre si. O aumento da qualidade do código e redução do tempo de desenvolvimento, faz com que a produtividade seja aumentada e que o tempo de desenvolvimento das aplicações seja reduzido. A manutenção é assim muito mais fácil devido a menor quantidade de bugs, e as aplicações muito mais seguras e com uma melhor performance. A modelação existente na *framework* faz com que o processo de desenvolvimento seja acelerado, e sejam criados standards dentro das equipas, o que leva a fazer com que as linhas das aplicações sejam mais semelhantes, visto que estão de certa forma delimitadas pelo potencial da modelação. Por sua vez, isto leva a que seja desperdiçado menos tempo a definir as regras de desenvolvimento para cada aplicação, focando-se assim no negócio, reduzindo o *time-to-market* e potencializando assim menores custos de licenciamento.

4.1.4 *Desvantagens de utilização da framework Elevation*

A questão: "Concorda com as seguintes desvantagens da utilização da framework Elevation?". para cada uma das desvantagens previamente identificadas permite verificar para cada uma qual o nível de concordância que lhe é atribuído na escala "Concordo totalmente"... "Discordo totalmente"

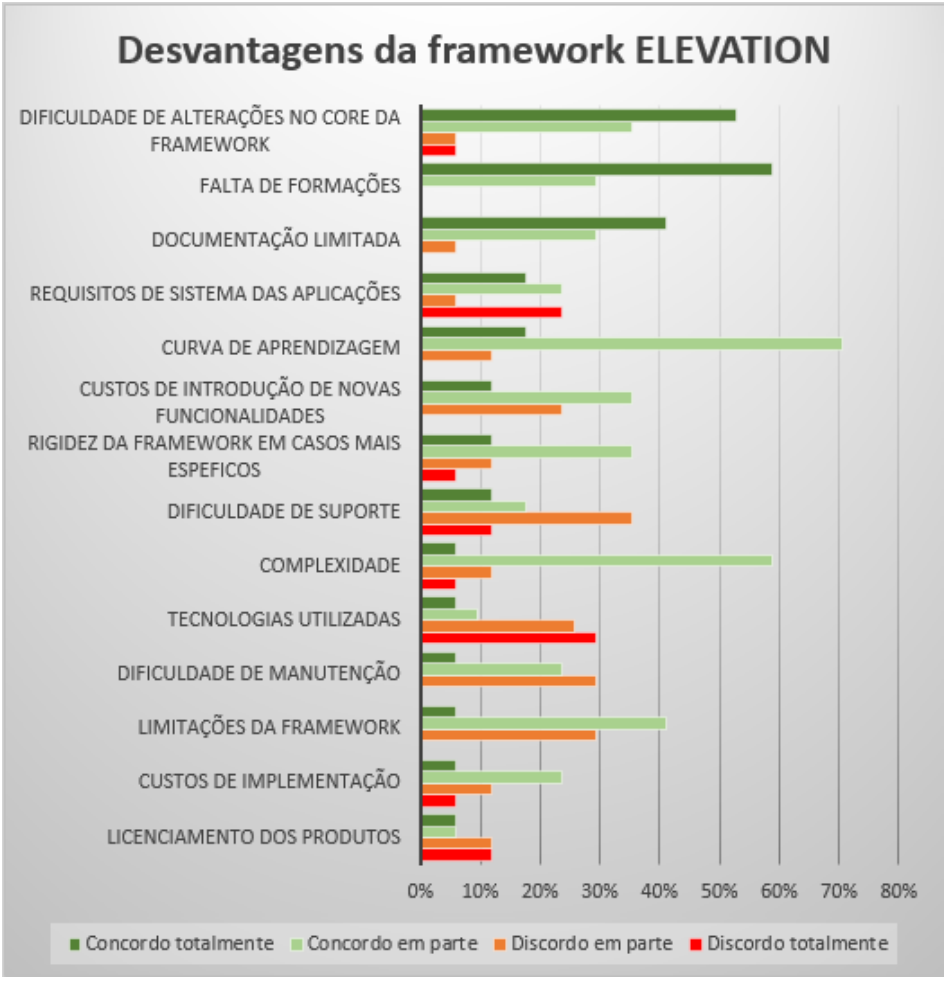


Figura 21.: Desvantagens de utilização da framework Elevation

A figura 22 apresenta de uma forma mais clara os resultados obtidos.

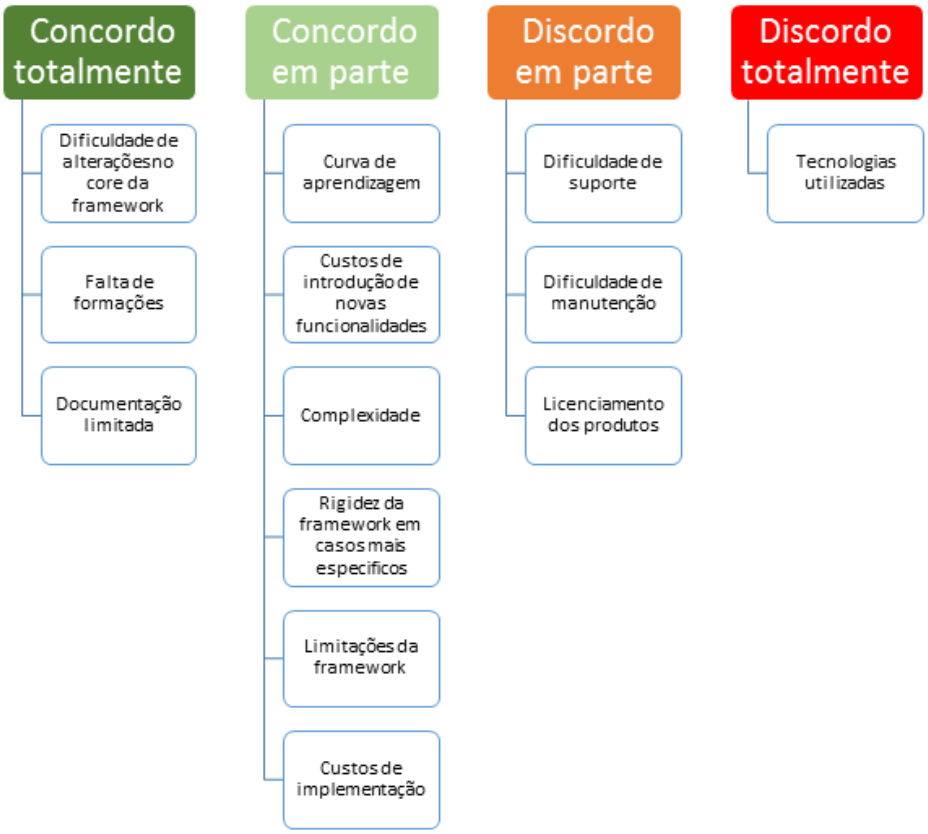


Figura 22.: Desvantagens de utilização da framework Elevation

Na revisão de literatura foram identificadas algumas desvantagens, entre as quais se destacam a complexidade e a curva de aprendizagem. Um ponto também abordado foi a durabilidade e alterações à *framework*. Como os requisitos das aplicações estão sempre a mudar, então é necessário fazer alterações no core da *framework*. Estas alterações nem sempre são fáceis, pois exigem um enorme conhecimento por parte dos programadores.

A correção de bugs pode ser um problema pois a *framework* está desenvolvida de forma a criar componentes genéricos e quando aparece um *bug* é difícil validar se o bug é realmente da *framework* ou da aplicação em si. Com o uso de *inversion of control (IoC)*, o fluxo de controlo oscila entre a infraestrutura da *framework* e da aplicação tornando ainda mais difícil a tarefa de fazer *debug*.

Outra das desvantagens das *frameworks* é a perda de eficiência, uma contradição com o objetivo da *framework*, porque as *frameworks* potenciam a extensibilidade fornecendo várias formas ao programador para poder estender a sua aplicação. Essa extensibilidade é fornecida, por exemplo, através da criação de subclasses e pela personalização das interfaces existentes. Este tipo de extensibilidade faz com que a eficiência da *framework* seja reduzida. (Fayad and C. Schmidt, 1997; Lopes, 2008; Taligent, 1993)

Podemos constatar que as desvantagens encontradas foram também identificadas na revisão de literatura. O problema de documentação limitada, está diretamente relacionado com documentação mal estruturada e falta de formação, que por consequência faz com que a curva de aprendizagem seja aumentada, tornando a *framework* ainda mais complexa para utilização (não que a *framework* seja complexa em si mesma). Esta falta de conhecimento e informação, faz com que seja difícil adicionar novas funcionalidades ao core da *framework*, aumentando assim o seu custo de implementação de funcionalidades. A versatilidade, rigidez e a dificuldade de alterações no core também estão relacionadas porque como a *framework* abstrai grande quantidade de código do programador, quando chega a altura de implementar novas funcionalidades nas *frameworks* (neste caso concreto, na framework Elevation), é necessário alguém muito experiente, tornando estes tipos de desenvolvimento um trabalho apenas para seniores.

Outra desvantagem realçada foi a tecnologia de *front-end*, que está relacionada com um ponto levantado na revisão de literatura: a durabilidade da *framework*. O *front-end* da *framework* Elevation foi desenvolvido à sete anos atrás em *SilverLight* da *Microsoft*. No entanto, esta tecnologia está agora a ser descontinuada, o que faz com que apareçam dificuldades no desenvolvimento, obrigando as equipas a efetuarem esforços extra para conseguirem satisfazer necessidades (ou cumprir com os requisitos) do mercado.

4.1.5 *Documentação da framework Elevation*

A questão: "Como classifica a documentação existente da *framework* Elevation?". para aferir a sua qualidade, permite verificar que apenas 3% dos inquiridos considera a documentação excelente, 18% considera-a boa, 20% considera-a suficiente, 41% respondeu que é insuficiente e 18% dos inquiridos considera a documentação muito insuficiente.

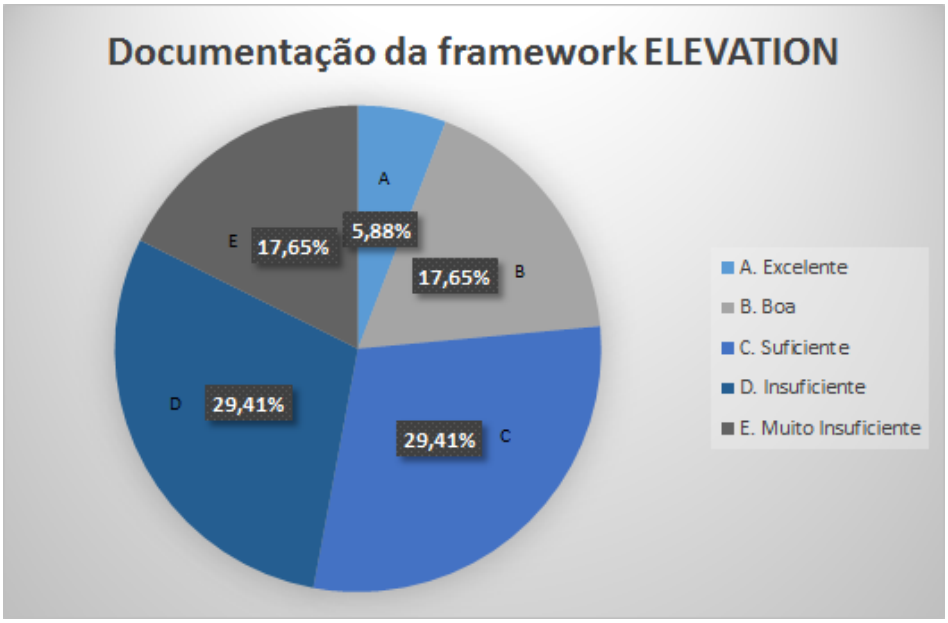


Figura 23.: Documentação da framework Elevation

Podemos concluir que os colaboradores consideram que a documentação existente não é suficiente, sendo este um aspeto pouco satisfatório para os colaboradores. Este ponto tem um impacto direto na curva de aprendizagem, fazendo que esta seja aumentada, tornando a *framework* ainda mais complexa.

4.1.6 *Formação existente sobre a framework Elevation*

Na figura 24 é possível observar se existe formação adequada para a utilização da *framework*, através das respostas à pergunta: "Considera que existe formação adequada?". Observando o gráfico podemos constatar que 88% dos inquiridos considera que não existe formação adequada para a utilização da *framework*.

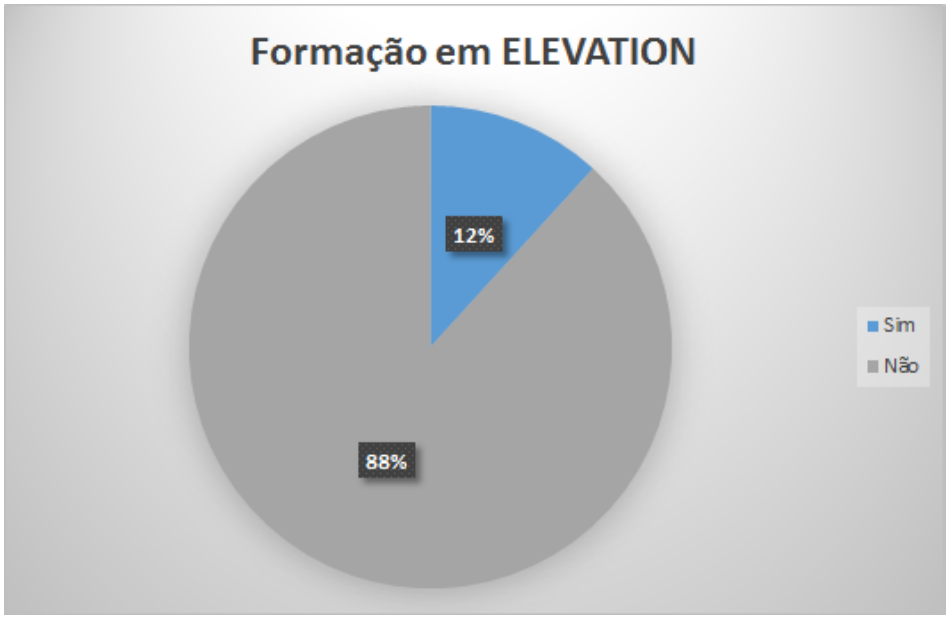


Figura 24.: Formação existente sobre a framework Elevation

Outra desvantagem que foi apresentada foi a falta de formação, e podemos ver que efetivamente os colaboradores consideram que não existe formação adequada. Começa já a ser possível observar que um dos principais aspetos negativos da *framework* é mesmo a falta de formação ou até mesmo informação disponível para os colaboradores. A maioria das formações existentes são sessões rápidas, onde é efetuada uma passagem por todo o processo de desenvolvimento, explicando os princípios muito básicos do desenvolvimento em Elevation.

4.1.7 *Ferramentas de apoio à framework Elevation*

O gráfico representado na figura 25, demonstra os resultados à questão: "Como classifica as ferramentas de apoio à *framework*?". Com esta questão é possível concluir se existem ferramentas de apoio à *framework*. Observando o gráfico constatamos que 67% dos inquiridos consideram que existem algumas ferramentas de apoio à *framework* Elevation, 28% considera que não existem nenhumas e 5% considera que existem bastantes ferramentas.

Em termos de ferramentas de apoio a *framework* Elevation, os colaboradores consideram que existem algumas. Através da análise documental e investigação, constatou-se que existem várias ferra-

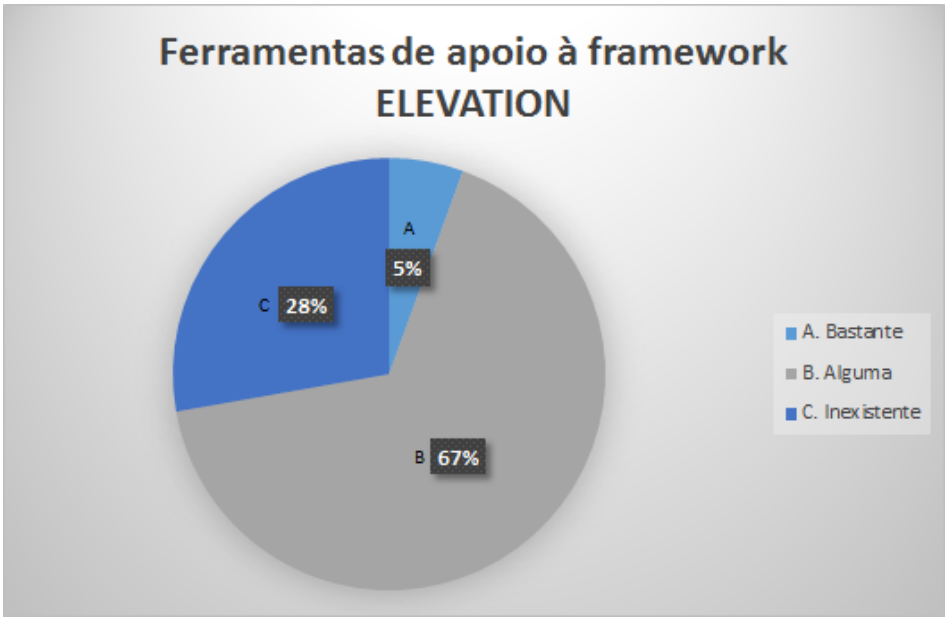


Figura 25.: Ferramentas de apoio à framework Elevation

mentas que facilitam as tarefas dos programadores. Existem algumas ferramentas de *upgrades* das bases de dados das aplicações desenvolvidas na *framework* que facilitam a manutenção e atualização das mesmas, tais como: ferramentas de upgrades das bases de dados das aplicações desenvolvidas na *framework*, que facilitam a manutenção e atualização das mesmas; e ferramentas foram todas desenvolvidas internamente, para apoiarem o processo de desenvolvimento de *software*.

4.1.8 *Tempo de aprendizagem*

A resposta à questão: "Qual o tempo de aprendizagem da *framework*?", permite observar que 59% dos inquiridos consideram que o tempo de aprendizagem atual é entre seis meses a um ano, 35% considera que o tempo de aprendizagem é menos de seis meses e apenas 6% consideram que é mais de um ano.

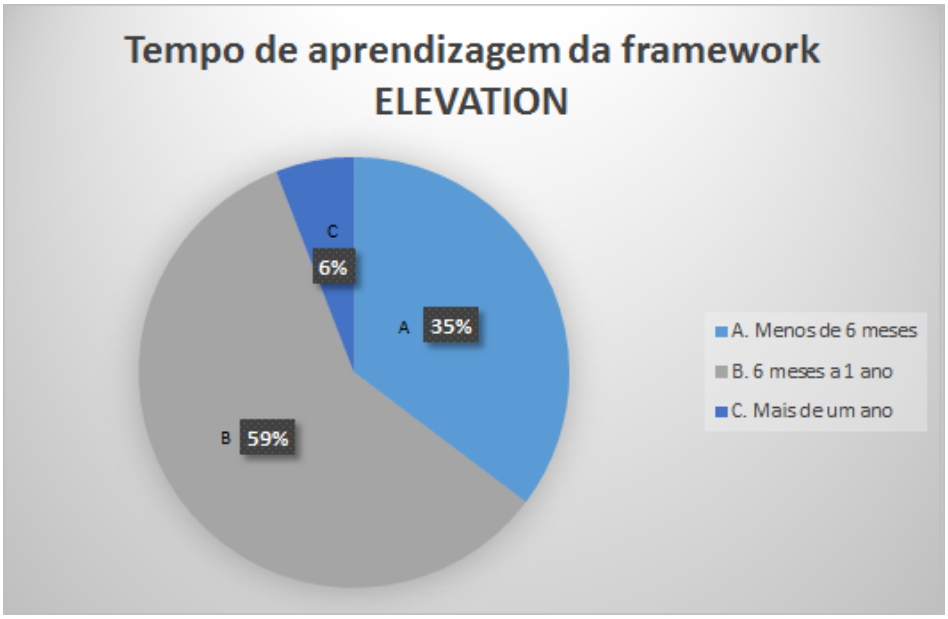


Figura 26.: Tempo de aprendizagem

A curva de aprendizagem de uma *framework* é uma desvantagem já levantada na revisão de literatura, sendo que Fayad and C. Schmidt (1997) já referiam que geralmente as *frameworks* são bastante complexas, fazendo que a sua curva de aprendizagem seja grande. Esta curva de aprendizagem vai sempre afetar a performance das equipas de desenvolvimento, sendo que é vantajoso que esta seja o mais curta possível. Como podemos ver a grande maioria dos colaboradores considera que esta curva de aprendizagem tem uma duração de seis meses a um ano. Este período deve ser um aspeto que deve receber alguma atenção de forma a aumentar a produtividade das equipas.

4.1.9 *Nível de satisfação dos utilizadores*

A questão: "Qual o seu nível de satisfação para com a *framework*?", permite concluir que 53% dos inquiridos estão satisfeitos, 35% estão satisfeitos e 12% estão insatisfeitos.

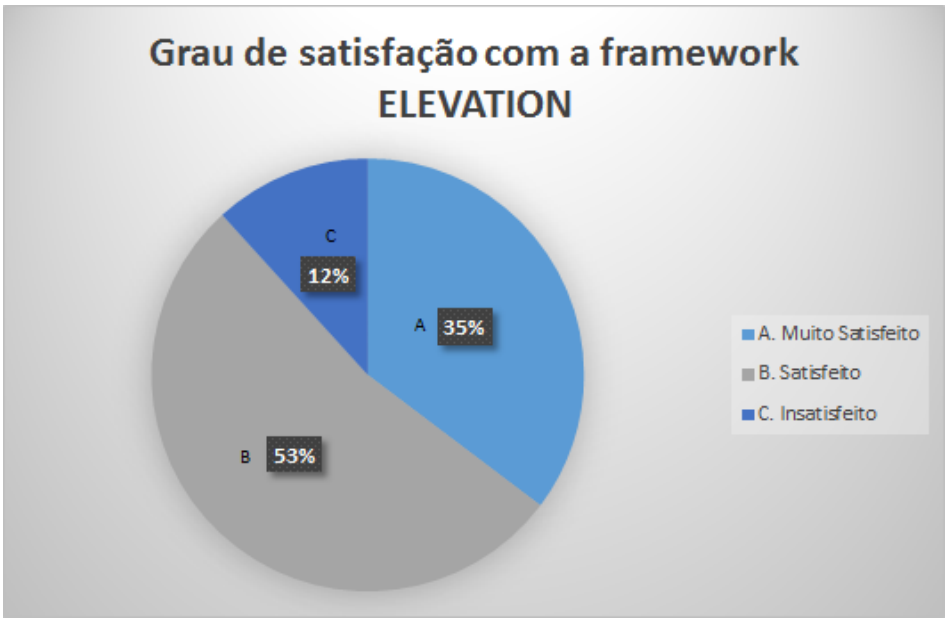


Figura 27.: Nível de satisfação dos utilizadores

De uma forma geral embora existam alguns colaboradores que não estão satisfeitos com a *framework*, o panorama geral é positivo. A satisfação advém das vantagens que a *framework* traz às equipa de desenvolvimento. No entanto, dentro do leque dos 88% dos colaboradores satisfeitos, apenas 35% estão muito satisfeitos, o que não deixa de ser um fator que merece atenção para que sejam tomadas algumas medidas para aumentar o nível de satisfação dos colaboradores.

4.1.10 *Lista de Vantagens*

Através da questão "Quais as vantagens que sente com a utilização da *framework* Elevation? Defina 5 vantagens ordenada da mais importante para a menos importante que vê com o uso da *framework*." foi possível identificar mais vantagens que não estavam presentes no questionário. Estas vantagens são apresentadas na tabela 6.

Tabela 6.: Lista de vantagens

Variável	Vantagens
Segurança das aplicações	"As aplicações são mais seguras"
Arquitetura das aplicações	"A arquitetura base é toda definida pela *framework* "
Estruturação	"Centralizar em menos gente decisões estruturais dos produtos"
Redução de custos	"Custos menores devido a tempos reduzidos no tempo de desenvolvimento"
Extensibilidade	"Através da modelação é muito fácil de estender as aplicações"
Consistência	"Garantia da consistência das regras de negócio em todas as camadas dos produtos."
Qualidade do código	"Garantia de qualidade do código"
Performance das aplicações	"Maior performance das aplicações"
Maior produtividade a longo prazo	"As equipas são mais produtivas a longo prazo"
Manutenção	"Facilidade de manutenção das aplicações"
Modelação	"A modelação e geração de código facilita o desenvolvimento"

4.1.11 *Lista de Desvantagens*

Através da questão "Quais as desvantagens que sente com a utilização da *framework* Elevation? Defina 5 desvantagens ordenada da mais importante para a menos importante que vê com o uso da *framework*." foi possível identificar desvantagens que não estavam presentes no questionário (tabela 7).

Tabela 7.: Lista de desvantagens

Vantagem	Resposta no questionário
Complexidade	"O nível de complexidade inicial é muito alto"
Curva de aprendizagem	"Curva de aprendizagem que pode ser acentuada" e "Dificuldade na previsão de todos os cenários possíveis aquando da implementação da FW "
Documentação	"Falta de documentação/informação para os *developers*"
Dificuldade de alterações no core	"Dificuldade de alterações no core da framework"
Formação	"Inexistência de formações para as equipas de produto"
Rigidez	"Limitações no desenvolvimento"
Falta de automatização nos testes	"Falta de aposta nos testes unitários de base na FW"
Tecnologia *front-end*	"A tecnologia silverlight utilizada para o front-end é demasiado pesada"

4.1.12 *Alterações para o Futuro*

Em relação à questão "Se pudesse realizar algumas alterações na *framework* Elevation o que sugeria?", foi possível obter as seguintes respostas dos inquiridos:

- "Criar documentação que seja facilmente consultada."
- "Alterações das interfaces para algo mais leve"
- "Criação de exemplos para apoio ao desenvolvimento"
- "Maior proximidade entre a equipa da plataforma e os programadores que a usam assim como destes 2 últimos em relação aos clientes finais"
- "Alteração das tecnologias usadas (Silverlight para HTML5)"
- "How to's sobre como desenvolver certos componentes de forma a standardizar o processo e aumentar o grau de conhecimento."
- "Maior aposta na área de Testes Unitários na base da FW e desenvolvimento do Core com exemplos de produto."
- "A FW deveria focar-se na geração da camada de dados e de serviços e não da apresentação/UI."
- "Maior flexibilidade da modelação"
- "Evolução tecnológica"
- "Permitir modelar mais cenários de negócio 'out of the box'"
- "Permitir reutilização de componentes"
- "Permitir modelar regras de UI"
- "Permitir modelar regras de negócio."
- "Existirem documentos com 'how to' decrevendo implementações de vários cenários de negócio"
- "Formação às equipas de DEV."

4.1.13 *Resumo de Resultados*

Embora as respostas tenham sido maioritariamente positivas, em relação às vantagens da utilização da *framework* Elevation foram encontradas também algumas desvantagens e ao longo destes nove meses trabalhando diretamente com a *framework*, realizando questionários, conversando com os *developers*

da PRIMAVERA BSS, foi elaborada um conjunto de benefícios e desvantagens que a *framework* proporciona às equipas de desenvolvimento da PRIMAVERA BSS.

Com base nos dados obtidos anteriormente é possível observar na tabela 8 um resumo das vantagens e desvantagens da *framework* Elevation.

Tabela 8.: Resumo dos resultados

Vantagens	Desvantagens
Aumenta a Segurança das aplicações	Complexidade
Facilita as decisões de arquitetura	Curva de aprendizagem
Estruturação	Documentação limitada
Reduz o tempo de desenvolvimento	Dificuldade de alterações no core
Extensibilidade	Falta de formações
Consistência	Rigidez da *framework* em casos mais específicos
Qualidade do código	Falta de automatização nos testes
Aumenta a performance das aplicações	Custos de implementação
Maior produtividade a longo prazo	Tecnologia *front-end*
Manutenção	Custos de introdução de novas funcionalidades
Modelação	Deployment das aplicações
Cria regras e boas práticas nas equipas	
Facilita o design das aplicações	
Reduz o *time-to-market*	
Reduz o custo de licenciamento	
Facilita os ciclos de teste	
Cria padrões nas linhas das aplicações	
Aumenta a rentabilidade	

Fazendo agora uma análise comparativa das vantagens e desvantagens identificadas, podemos ver claramente que a *framework* Elevation traz bastantes benefícios para a PRIMAVERA BSS, embora existam algumas falhas que deveriam ser revistas. As principais vantagens destacadas foram ao nível de produção de *software*, e as principais desvantagens foram ao nível de aprendizagem e formação sobre a *framework*. A falta de documentação e formação inicial na *framework* é uma desvantagem referida por bastantes elementos das equipas de desenvolvimento. Este problema pode estar a ser um pouco mitigado uma vez que as equipas que trabalham com a *framework* são bastantes experientes. No entanto, à medida que se for recrutando novos programadores sem experiência na *framework* os custos de desenvolvimento de aplicações podem aumentar, uma vez que o tempo médio de aprendizagem da *framework* Elevation é superior a seis meses e durante este período médio de tempo a produção dos programadores vai ser reduzida, reduzindo consequentemente a produtividade das equipas.

5

CONCLUSÕES

5.1 DISCUSSÃO DE RESULTADOS

Procura-se com este estudo analisar a *framework* de desenvolvimento da Elevation, que é um produto muito importante para a PRIMAVERA BSS uma vez que desenvolver uma *framework* de desenvolvimento de *software* é algo que exige bastante esforço por parte das empresas, e por isso é importante perceber quais os benefícios e desvantagens desta *framework* que a PRIMAVERA BSS tem vindo a desenvolver ao longo destes sete anos.

Com este estudo pretende-se responder a uma questão sobre quais as vantagens e desvantagens da *framework* Elevation. Este estudo foi realizado utilizando um estudo de caso estruturado em seis atividades: o planeamento, design, preparação, recolha de dados, análise de dados e por último, conclusões.

5.1.1 *Validação de Resultados*

A validação de resultados foi efetuada utilizando o método de grupo de discussão focalizada (Um grupo de discussão focalizada é um grupo que se reúne para uma discussão aberta sobre um tema ou problema. Fornecendo um conjunto de observações (Eliot, 2005)). Para este grupo, foram convidados quatro elementos das equipas de desenvolvimento aos quais foram apresentados os resultados obtidos. De seguida foi-lhes solicitado que discutissem os mesmos, referindo com quais concordam ou discordam, assim como os que consideram estar em falta (caso se aplique).

Nas vantagens apresentadas houve concordância geral por parte dos membros do grupo de discussão focalizada. Foi comentada a redução do tempo de desenvolvimento e a rentabilidade, salientando que esta redução e o aumento da rentabilidade é mais evidente a longo prazo. Quando se obtém mais conhecimento com a *framework*, o mesmo pode não se observar em programadores menos experientes. A performance das aplicações foi também um ponto comentado, em que foi referido que concordam que a performance geral das aplicações é melhorada devido às funcionalidades de balanceamento de carga dos servidores que a *framework* proporciona, mas observam-se algumas falhas de performance de *front-end*, sendo este aspeto já é abordado como uma desvantagem da *framework*.

Nas desvantagens houve também um nível de aceitação dos resultados quase total, Foi apenas referido que o *deployment* das aplicações é uma vantagem no ambiente *on-premises* mas no que diz respeito ao *deployment* no ambiente *cloud* ainda existem muitas limitações e não está devidamente automatizado. Em relação às outras desvantagens apresentadas não foi efetuado nenhum comentário ou sugestão por parte dos elementos do grupo.

Concluindo a validação dos resultados, houve concordância geral nos resultados obtidos, não sendo sugerido ou apontada nenhuma vantagem ou desvantagem em falta. Com este grupo de discussão focalizada foi possível ouvir um grupo de programadores que trabalham diretamente com a *framework* Elevation e perceber a apreciação que fazem dos resultados obtidos com este trabalho de investigação.

5.1.2 *Vantagens Identificadas*

É importante para a PRIMAVERA BSS que a *framework* Elevation seja considerada uma mais valia por parte dos programadores. Sendo assim, de acordo com os resultados obtidos podemos identificar principalmente as seguintes vantagens: criação de regras e boas práticas nas equipas, facilita o design das aplicações, aumento de performance das aplicações, aumento da segurança das aplicações, facilidade dos ciclos de teste, aumento da produtividade, criação de padrões nas linhas das aplicações e aumento da rentabilidade.

Conhecer estas vantagens permite à PRIMAVERA BSS saber se está no bom caminho e o que deve ser mantido em futuras atualizações à *framework* Elevation, pois é importante que as vantagens encontradas não sejam esquecidas em futuras atualizações.

5.1.3 *Desvantagens Identificadas*

Conhecer quais as desvantagens na utilização da *framework* Elevation permite-nos saber o que precisa de ser trabalhado por parte da PRIMAVERA BSS, de forma a acrescentar valor à sua *framework* de desenvolvimento, para que se possa tornar numa ferramenta mais eficaz e assim explorar todos os benefícios que uma *framework* de desenvolvimento pode trazer às equipas de desenvolvimento.

Uma desvantagem apontada tem que ver com a dificuldade das equipas no core da *framework*. Como se trata de uma *framework*, apenas a equipa responsável pelo seu desenvolvimento pode fazer estas alterações, estando o código base da *framework* abstraído dos programadores que desenvolvem sobre ela. Outra desvantagem apontada tem que ver com a dificuldade das equipas em adicionar algumas funcionalidades novas às suas aplicações, estando dependentes de *releases* da *framework*. A falta de formações foi um fator apontado várias vezes sendo também referida a documentação limitada existente na *framework*. A curva de aprendizagem e a complexidade foram também outras das desvantagens referidas pelos programadores.

A PRIMAVERA BSS deve atentar nestas desvantagens nas futuras atualizações da *framework*, de forma a tentar reduzir o impacto das mesmas.

5.1.4 *Recomendações do Investigador*

Ao longo desta investigação foi possível interagir com um grupo alargado de colaboradores da PRIMAVERA BSS, desde programadores, *testers* e até chefes de equipa que têm bastante experiência com a *framework* Elevation.

Com base nesta investigação, foi possível analisar um conjunto de informações relevantes não só interagindo com os colaboradores mas também consultando artigos sobre a temática. Os dados recolhidos permitem assim definir um conjunto de recomendações à PRIMAVERA BSS:

- Definir princípios para a evolução constante na *framework* Elevation.
- Melhorar a documentação de suporte à utillização da *framework*.
- Intensificar a formação dos colaboradores na *framework*.
- Promover a aproximação das equipas de desenvolvimento.

Uma das primeiras recomendações que se tem que ter em conta é que a equipa de desenvolvimento deve apostar na inovação constante, acompanhando as novas tecnologias, de forma a criar o máximo de valor possível para os programadores que utilizam a *framework* Elevation, não esquecendo a criação de ferramentas de apoio que permitem aumentar a produtividade e a facilidade de utilização.

A inovação deve ter em conta os fatores que os programadores mais valorizam numa *framework*. Desta forma em futuras atualizações à *framework* Elevation devem ser considerados os seguintes fatores: a performance, escalabilidade, reutilização de código, eficiência, extensibilidade e rapidez de implementação. Estes foram os 5 fatores mais destacados pelos programadores da PRIMAVERA BSS. Seria importante também tentar aproximar as equipas que trabalham diretamente com a *framework* Elevation, de forma a ser possível perceber as dificuldades estas estão a sentir, o que poderia vir a ser melhorado, o que está a funcionar bem e não deve ser alterado, entre outros. Esta aproximação poderia ser feita através de um fórum onde sejam criadas recomendações por parte destes programadores e onde também poderiam ser partilhadas dificuldades. Seria também interessante dedicar algum tempo para a automatização de testes nas aplicações geradas pela *framework*, facilitando ainda assim mais os testes finais antes dos produtos serem lançados para o mercado, retirando carga de trabalho às equipas de desenvolvimento.

Para além disto, deveria ser criada documentação mais eficaz para os programadores poderem consultar, visto que se trata de um produto interno da empresa e não é possível na maioria dos casos encontrar informação na online ou em livros. A documentação deve ser a base da *framework* e devem ser documentadas todas as funcionalidades e características técnicas da *framework* e devem existir exemplos para diversas funcionalidades.

Seria interessante que a documentação fornecida aos programadores fosse do género de um *cook-book* uma vez que existem já vários *cook-books* sobre diversas ferramentas que a PRIMAVERA BSS

poderia usar como inspiração. Esta documentação deveria ser devidamente partilhada e dada a conhecer a todos os utilizadores da *framework*. Também seria interessante a criação de uma biblioteca de conhecimento ou até mesmo um fórum em que os vários colaboradores pudessem contribuir com informação que achassem relevante. Esta biblioteca de conhecimento deveria ser encorajada junto dos utilizadores, para que exista a maior de informação útil possível.

Uma outra recomendação seria a existência de formações sobre a *framework* Elevation, para permitir uma aprendizagem mais rápida por parte dos programadores menos experientes mas também ensinar novas práticas a programadores mais experientes. Estas formações seriam ainda uma ótima oportunidade para o esclarecimento de dúvidas existentes e sugestões de funcionalidades. As formações deveriam ser dadas pelos programadores mais experientes preferencialmente ligados ao desenvolvimento da *framework* Elevation, tentando manter estas formações o mais práticas possíveis, para que o formando pudesse assimilar o máximo de conhecimento possível.

Para além das tradicionais sessões de formação sugeridas, poderiam também ser criadas formações e-learning, de forma a ajudar os colaboradores que começam a trabalhar com a *framework* Elevation, que explicariam todo o processo de desenvolvimento, fazendo com que os colaboradores resolvam alguns exercícios práticos. Este conjunto de formações teriam um grande impacto no desenvolvimento, pois iria ajudar a reduzir a curva de aprendizagem e a diminuir a complexidade da *framework*.

Por último recomenda-se que a PRIMAVERA BSS esteja sempre pronta a ouvir as opiniões de quem trabalha com a *framework* diariamente de forma a garantir que a utilização da *framework* seja um benefício para os programadores. Adicionalmente, seria vantajoso que a PRIMAVERA BSS tentasse de alguma forma aproximar a equipa que desenvolve a *framework* Elevation com as restantes equipas que a utilizam, visto que a necessidade de criar padrões e utilizar a *framework* nasce dos produtos, por isso é muito importante ouvir quem os produz.

5.2 LIMITAÇÕES E TRABALHO FUTURO

Durante o decorrer desta investigação foram encontradas algumas limitações, mas que podem ser aproveitadas como base de trabalho para desenvolvimentos futuros.

As principais limitações que surgiram foram:

- Não existe nenhuma informação que destaque a evolução das equipas que desenvolvem *software* utilizando a *framework* de forma a perceber os aumentos de produtividade.

- O facto de não haver nenhum estudo, que mostre quais foram os custos do desenvolvimento desta *framework*. Assim, não é possível comparar com os atuais ganhos, de forma a perceber qual é o retorno do investimento na *framework* Elevation para a PRIMAVERA BSS.

As seguintes recomendações de trabalho futuro são baseadas nas limitações identificadas e também nos resultados obtidos. Como desenvolvimentos futuros, sugerem-se:

- Efetuar um estudo a uma equipa especifica de forma a perceber a sua evolução ao longo do tempo, se a evolução da *framework* está no bom caminho e perceber quais as diferenças do inicio da utilização da *framework* para e presente.

- Fazer um estudo a duas equipas, em que seja proposto o planeamento do desenvolvimento de uma aplicação. Uma das equipas utilizaria a *framework* Elevation e a outra efetuaria o desenvolvimento sem a *framework*. Posteriormente seriam comparados os planeamentos e orçamentação de forma a perceber as diferenças. As equipas seriam ainda questionadas sobre soluções a possíveis riscos que poderiam ocorrer durante o projeto. Os envolvidos teriam de ter experiência profissional equivalente para que o estudo fosse o mais preciso possível.

- Criar um conjunto de boas práticas de criação de documentação. Este manual seria elaborado de acordo com as necessidades dos colaboradores.

5.3 REFLEXÕES FINAIS

Dou assim por terminado esta investigação, com enormes lições tiradas e com a sensação de missão cumprida. Com este estudo foi possível incorporar o dia a dia de uma equipa de desenvolvimento, que trabalha utilizando uma *framework* de desenvolvimento. Foi possível viver os problemas que os programadores sentem no dia a dia, quando estão dependentes deste tipo de *frameworks*. Muitas vezes as empresas focam-se demasiado em tentar produzir uma *framework* que faça tudo e que consiga abrangir o máximo de funcionalidades, deixando um pouco de parte o que realmente as equipas de desenvolvimento precisam.

As *frameworks* são feitas para ajudarem as equipas de desenvolvimento, e como consequência melhorar a empresa. Deve-se ter muito cuidado sempre que estamos a desenvolver este tipo de *frameworks*, visto que por vezes são desenvolvidas funcionalidades que não fazem falta. É importante escutar as equipas, recolher o máximo de *feedback* possível dos colaboradores, fornecer meios para que estas (as equipas) possam usufruir de uma ferramenta que possa ser tão poderosa. Outro aspeto importante é que a framework deve ser tão simples quanto possível, ou seja, abstrair a complexidade associada à sua implementação. Só porque o desenvolvimento da framework em si é um processo complexo e elaborado, não significa que essa complexidade deva ser passada para o utilizador (neste caso, developer) da framework! Deve ser dada particular atenção a uma documentação bem estruturada e formações, pois são investimentos que não vão trazer diretamente retornos monetários, mas vão aumentar a produção e, como efeito colateral, também a rentabilidade das equipas.

A identificação das vantagens e desvantagens, bem como a apresentação de conclusões relativamente à utilização da framework Elevation, são contributos desta investigação para a literatura referente ao desenvolvimento de *frameworks*, relativamente escassa em trabalhos empíricos. Constitui, também, um contributo não só para a Primavera, mas para todas as *software houses* que estejam ou

venham a estar envolvidas no desenvolvimento de *frameworks* para aumentarem a produtividade das suas equipas de desenvolvimento.

REFERÊNCIAS BIBLIOGRÁFICAS

ACM. Acm digital library, 1996. URL http://dl.acm.org/dl.cfm.

David Beazley and Brian K. Jones. *Python Cookbook, Third edition*. O'Reilly Media, Beijing, 3 edition edition, June 2013. ISBN 9781449340377.

Margaret Burnett, Adele Goldberg, and Ted Lewis. *Visual object-oriented programming: concepts and environments*. Manning, February 1995. ISBN 9780131723979.

G. Butler, P. Grogono, and F. Khendek. A reuse case perspective on documenting frameworks. In *Software Engineering Conference, 1998. Proceedings. 1998 Asia Pacific*, pages 94–101, December 1998. doi: 10.1109/APSEC.1998.733596.

L. P. Deutsch. Software reusability. chapter Design Reuse and Frameworks in the Smalltalk-80 System, pages 57–71. ACM, New York, NY, USA, 1989. ISBN 0-201-50018-3. doi: 10.1145/75722.75725. URL http://doi.acm.org/10.1145/75722.75725.

A Eliot. Guidelines for conducting a focus group. 2005.

Mohamed Fayad and Douglas C. Schmidt. Object-oriented application frameworks. volume 40. Special Issue on Object-Oriented Application Frameworks, 1997.

J. E. Gaffney, Jr. and R. D. Cruickshank. A general economics model of software reuse. In *Proceedings of the 14th International Conference on Software Engineering*, ICSE '92, pages 327–337. ACM, 1992. ISBN 0-89791-504-6. doi: 10.1145/143062.143150. URL http://doi.acm.org/10.1145/143062.143150.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, Reading, Mass. u.a., 1 edition edition, May 1998. ISBN 9780201634983.

Adele Goldberg and David Robson. *Smalltalk-80: The Language and Its Implementation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983. ISBN 0-201-11371-6.

Google. Google scholar, 2005. URL https://scholar.google.pt/.

Martin L. Griss and Kevin D. Wentzel. Hybrid domain-specific kits for a flexible software factory. In *Proceedings of the 1994 ACM Symposium on Applied Computing*, SAC '94, pages 47–52, New York, NY, USA, 1994. ACM. ISBN 0-89791-647-6. doi: 10.1145/326619.326658. URL http://doi.acm.org/10.1145/326619.326658.

IEEE. Xplore digital library, 2005. URL www.ieeexplore.ieee.org.

Ralph E. Johnson. Documenting frameworks using patterns. *SIGPLAN Not.*, 27(10):63–76, October 1992. ISSN 0362-1340. doi: 10.1145/141937.141943. URL http://doi.acm.org/10.1145/141937.141943.

Ralph E. Johnson. Components, frameworks, patterns. *COMMUNICATIONS OF THE ACM*, 40: 10–17, 1997a.

Ralph E. Johnson. Frameworks = (components + patterns). *Commun. ACM*, 40(10):39–42, October 1997b. ISSN 0001-0782. doi: 10.1145/262793.262799. URL http://doi.acm.org/10.1145/262793.262799.

Ralph E Johnson and Brian Foote. Designing reusable classes. *Journal of object-oriented programming*, 1(2):22–35, 1988.

Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, August 1988. ISSN 0896-8438. URL http://dl.acm.org/citation.cfm?id=50757.50759.

Sérgio F Lopes. *Architecture and documentation integrated method for the design and reuse of frameworks*. PhD thesis, Universidade do Minho, 2008.

Michael Mattsson. *Evolution and Composition of Object-Oriented Frameworks*. PhD thesis, University of Karlskrona/Ronneby, 2000.

M. D. McIlroy. Mass-produced software components. *Proc. NATO Conf. on Software Engineering, Garmisch, Germany*, 1968.

Microsoft. Microsoft .net framework, 2002. URL http://microsoft.com/net.

Vandad Nahavandipoor. *8 Swift Programming Cookbook: Solutions & Examples for iOS Apps*. O'Reilly Media, 1 edition edition, December 2014. ISBN 9781491908693.

NATO. North atlantic treaty organization, 1949. URL http://www.nato.int/.

Oscar Nierstrasz and Laurent Dami. Object-oriented software composition. chapter Component-oriented Software Technology, pages 3–28. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1995. ISBN 0-13-220674-9. URL http://dl.acm.org/citation.cfm?id=232469.232470.

Inc ParcPlace-Digitalk. *VisualWorks: Cookbook*. ParcPlace-Digitalk, Incorporated, 1995. URL http://books.google.pt/books?id=tP0RHQAACAAJ.

Wolfgang Pree and Hermann Sikora. Design patterns for object-oriented software development (tutorial). In *Proceedings of the 19th International Conference on Software Engineering*, ICSE '97, pages 663–664, New York, NY, USA, 1997. ACM. ISBN 0-89791-914-9. doi: 10.1145/253228.253810. URL http://doi.acm.org/10.1145/253228.253810.

PRIMAVERA. Anual report. Technical report, PRIMAVERA BSS, 2011.

Alexander Reelsen. *Play Framework Cookbook*. Packt Publishing, Birmingham, August 2011. ISBN 9781849515528.

Don Roberts and Ralph Johnson. Evolving frameworks: A pattern language for developing object-oriented frameworks. In *Proceedings of the Third Conference on Pattern Languages and Programming*. Addison-Wesley, 1996.

RUM. Repositório da universidade do minho, 2003. URL https://repositorium.sdum.uminho.pt/.

K. Schmucker. *Object-Oriented Programming for the Macintosh*. Hayden Book Company, 1986.

Scopus. Scopus, 2004.

Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002. ISBN 0201745720.

Inc. Taligent. Building object-oriented frameworks. *A Taligent White Paper*, page 23, November 1993. URL http://lhcb-comp.web.cern.ch/lhcb-comp/Components/postscript/buildingoo.pdf.

Kevin D. Wentzel. Software reuse—facts and myths. In *Proceedings of the 16th international conference on Software engineering*, pages 267–268. IEEE Computer Society Press, 1994. URL http://dl.acm.org/citation.cfm?id=257779.

Rebecca J. Wirfs-Brock and Ralph E. Johnson. Surveying current research in object-oriented design. *Commun. ACM*, 33(9):104–124, September 1990. ISSN 0001-0782. doi: 10.1145/83880.84526. URL http://doi.acm.org/10.1145/83880.84526.

WoK. Web of knowledge. URL https://webofknowledge.com/.

R.K. Yin. *Case Study Research: Design and Methods*. Applied Social Research Methods. SAGE Publications, 2009. ISBN 9781412960991. URL http://books.google.pt/books?id=FzawIAdilHkC.

A

ANEXOS

A.1 DESCRIÇÃO DAS QUESTÕES DO QUESTIONÁRIO FINAL

A tabela 9 apresenta as questões (e respetivo tipo) efetuadas aos elementos das equipas de desenvolvimento da PRIMAVERA BSS.

Tabela 9.: Questionário efetuado aos elementos da PRIMAVERA BSS

Questão	**Tipo**	**Objetivo**
Qual a sua equipa?	Livre	O objetivo desta pergunta é identificar qual a equipa em que este colaborador está inserido na PRIMAVERA BSS.
Qual a sua função na empresa/equipa?	Livre	O objetivo desta pergunta é saber qual a função do inquirido dentro da PRIMAVERA BSS.
Quantos anos tem de experiência profissional?	Escolha Única	O objetivo desta pergunta é perceber qual a experiência profissional do inquirido.
Já trabalhou com a *framework* Elevation?	Escolha Única	Esta pergunta tem como objetivo saber se o inquirido já trabalhou com a *framework* Elevation.
Já trabalhou com outro tipo de *framework* de desenvolvimento?	Escolha Única	Esta questão tem como objetivo saber se o inquirido já trabalhou com outra *framework* de desenvolvimento de forma a perceber o seu nível de experiência com *frameworks*.
continua na próxima página		

Tabela 9 – continuação da pagina anterior

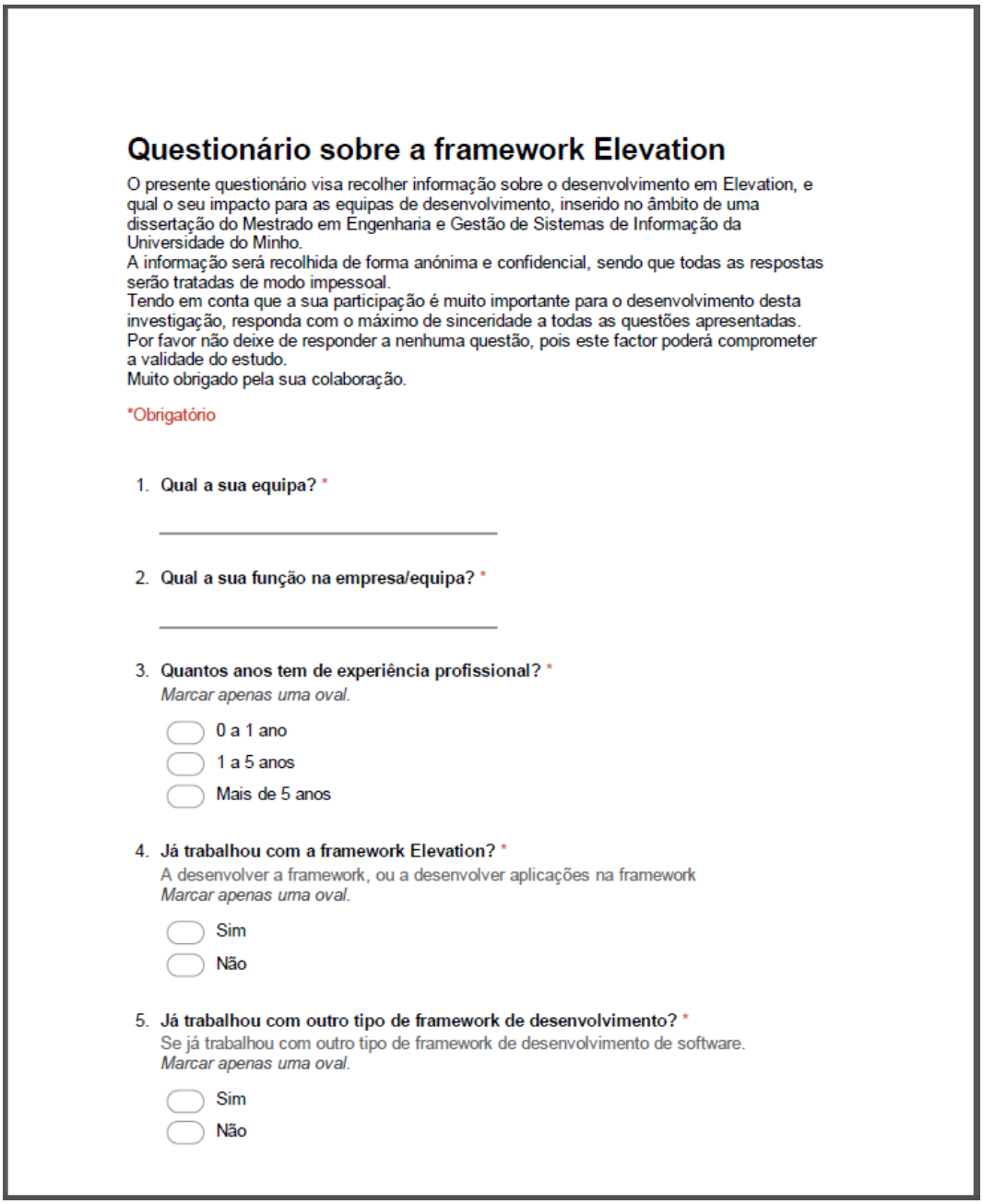
Questão	Tipo	Objetivo
Classifique a importância dos seguintes características de uma *framework* para quem as utiliza.	Livre	Esta pergunta tem como objetivo identificar quais os fatores mais importantes para quem desenvolve aplicações sobre uma *framework* e perceber o que realmente os inquiridos valorizam numa *framework*. Desta forma consegue-se extrair aspetos que devem ser tidos em conta na evolução da *framework* Elevation.
Concorda com os seguintes vantagens da utilização da *framework* Elevation?	Livre	Pretende-se com esta pergunta validar quais são os maiores benefícios que a *framework* Elevation trás para os inquiridos. Desta forma consegue-se evidenciar se a *framework* está mesmo a trazer vantagens para os colaboradores da PRIMAVERA BSS e identificar quais as vantagens mais sentidas por eles.
Concorda com os seguintes desvantagens da utilização da *framework* Elevation?	Livre	Esta pergunta tem como objetivo identificar quais são os pontos mais fracos na utilização da *framework* Elevation. Desta forma pode-se evidenciar quais as principais desvantagens da *framework* e assim tecer um conjunto de recomendações para que seja possível tornar a *framework* o mais vantajosa possível.
Como classifica a documentação existente sobre a framework Elevation?	Livre	Pretende-se identificar o grau de documentação existente. Com esta questão podemos perceber melhor se este é um problema na PRIMAVERA BSS, identificando assim qual a opinião dos colaboradores em relação à documentação da *framework* Elevation.
Como classifica as ferramentas de apoio à framework Elevation?	Livre	Esta pergunta tem como objetivo classificar as ferramentas de apoio existentes, percebendo assim qual a opinião dos colaboradores em relação às ferramentas de apoio da *framework* Elevation.

continua na próxima página

Tabela 9 – continuação da pagina anterior

Questão	Tipo	Objetivo
Considera que existe formação adequada?	Escolha Única	Pretende-se determinar o nível de formação concedido sobre a *framework* Elevation. Conseguindo assim perceber se os inquiridos concordam se existe formação adequada sobre a *framework* Elevation ou se precisa de ser melhorada.
Qual o tempo de aprendizagem da *framework* Elevation?	Escolha Única	Com esta pergunta procura-se saber qual o tempo de aprendizagem da *framework*. Desta forma podemos observar quais os tempos de aprendizagem que são identificados pelos inquiridos, conseguindo perceber se a *framework* é complexa ou não.
Qual o seu nível de satisfação para com a *framework* Elevation?	Livre	Com esta pergunta procura-se identificar o nível de satisfação do utilizador com a *framework*, de forma a observar o grau de satisfação dos colaboradores, alertando para possíveis problemas da *framework* Elevation.
Quais as vantagens que sente com a utilização da *framework* Elevation? Defina 5 vantagens ordenada da mais importante para a menos importante que vê com o uso da *framework*.	Livre	O objetivo desta questão é tentar perceber quais as vantagens sentidas, recolhendo assim mais variáveis de avaliação que não estavam presentes no questionário.
Quais as desvantagens que sente com a utilização da *framework* Elevation? Defina 5 vantagens ordenada da mais importante para a menos importante que vê com o uso da *framework*.	Livre	O objetivo desta questão é tentar perceber quais as desvantagens sentidas, recolhendo assim mais variáveis de avaliação que não estavam presentes no questionário.
Se pudesse realizar algumas alterações na *framework* Elevation o que sugeria?	Livre	Esta questão tem como objetivo tentar perceber o que os entrevistados alterariam na *framework*.

A.2 QUESTIONÁRIO AOS COLABORADORES DA PRIMAVERA BSS



Questionário sobre a framework Elevation

O presente questionário visa recolher informação sobre o desenvolvimento em Elevation, e qual o seu impacto para as equipas de desenvolvimento, inserido no âmbito de uma dissertação do Mestrado em Engenharia e Gestão de Sistemas de Informação da Universidade do Minho.
A informação será recolhida de forma anónima e confidencial, sendo que todas as respostas serão tratadas de modo impessoal.
Tendo em conta que a sua participação é muito importante para o desenvolvimento desta investigação, responda com o máximo de sinceridade a todas as questões apresentadas.
Por favor não deixe de responder a nenhuma questão, pois este factor poderá comprometer a validade do estudo.
Muito obrigado pela sua colaboração.

*Obrigatório

1. **Qual a sua equipa?** *

2. **Qual a sua função na empresa/equipa?** *

3. **Quantos anos tem de experiência profissional?** *
   Marcar apenas uma oval.
   - 0 a 1 ano
   - 1 a 5 anos
   - Mais de 5 anos

4. **Já trabalhou com a framework Elevation?** *
   A desenvolver a framework, ou a desenvolver aplicações na framework
   Marcar apenas uma oval.
   - Sim
   - Não

5. **Já trabalhou com outro tipo de framework de desenvolvimento?** *
   Se já trabalhou com outro tipo de framework de desenvolvimento de software.
   Marcar apenas uma oval.
   - Sim
   - Não

Figura 28.: Questionário aos colaboradores da PRIMAVERA BSS parte 1

6. **Classifique a importância dos seguintes características de uma framework para quem as utiliza.** *

Sendo 1 "Não Importante" e 5 "Muito Importante"
Marcar apenas uma oval por linha.

	Irrelevante	Pouco Importante	Importante	Muito Importante
Documentação				
Requisitos de hardware				
Curva de aprendizagem				
Pré-Customização				
Estruturação				
Abstração				
Eficiência				
Extensibilidade				
Uniformização				
Manutenção				
Design das Aplicações				
Reutilização de código				
Segurança				
Rapidez de implementação				
Formação				
Escalabilidade				
Modularização				
Compatibilidade				
Performance				
Exemplos existentes				

Figura 29.: Questionário aos colaboradores da PRIMAVERA BSS parte 2

7. **Concorda com os seguintes vantagens da utilização da framework Elevation?** *

Marcar apenas uma oval por linha.

	Discordo totalmente	Discordo em parte	Concordo em parte	Concordo totalmente
Aumenta a performance das aplicações	◯	◯	◯	◯
Reduz o time-to-market	◯	◯	◯	◯
Facilita o design das aplicações	◯	◯	◯	◯
Facilita as decisões de arquitetura	◯	◯	◯	◯
Aumenta a segurança das aplicações	◯	◯	◯	◯
Facilita o desenvolvimento dos parceiros	◯	◯	◯	◯
Diminui o tempo de desenvolvimento	◯	◯	◯	◯
Aumenta a rentabilidade	◯	◯	◯	◯
Facilita o Deployment	◯	◯	◯	◯
Cria padrões nas linhas das aplicações	◯	◯	◯	◯
Redução do custo de licenciamento	◯	◯	◯	◯
Reduz os recursos humanos	◯	◯	◯	◯
Facilita os ciclos de testes	◯	◯	◯	◯
Aumenta a produtividade	◯	◯	◯	◯
Cria regras e boas práticas nas equipas	◯	◯	◯	◯
Reduz o investimento	◯	◯	◯	◯

Figura 30.: Questionário aos colaboradores da PRIMAVERA BSS parte 3

8. **Concorda com os seguintes desvantagens da utilização da framework Elevation?** *
Marcar apenas uma oval por linha.

	Discordo totalmente	Discordo em parte	Concordo em parte	Concordo totalmente
Curva de Aprendizagem	◯	◯	◯	◯
Dificuldade de alterações do core da framework	◯	◯	◯	◯
Documentação limitada	◯	◯	◯	◯
Falta de formações	◯	◯	◯	◯
Dificuldade Suporte	◯	◯	◯	◯
Licenciamento dos produtos	◯	◯	◯	◯
Custos de implementação	◯	◯	◯	◯
Rigidez da framework em casos mais especificos	◯	◯	◯	◯
Limitações da framework	◯	◯	◯	◯
Requisitos de sistema das aplicações	◯	◯	◯	◯
Custo de introdução de novas funcionalidades	◯	◯	◯	◯
Dificuldade de manutenção	◯	◯	◯	◯
Tecnologias utilizadas	◯	◯	◯	◯
Complexidade	◯	◯	◯	◯

9. **Como classifica a documentação existente sobre a framework Elevation?** *
Marcar apenas uma oval.

	1	2	3	4	5	
Muito Insuficiente	◯	◯	◯	◯	◯	Muito Suficiente

10. **Como classifica as ferramentas de apoio à framework Elevation?** *
Marcar apenas uma oval.

- ◯ Inexistente
- ◯ Alguma
- ◯ Bastante

11. **Considera que existe formação adequada Elevation?** *
Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

Figura 31.: Questionário aos colaboradores da PRIMAVERA BSS parte 4

12. **Qual o tempo de aprendizagem da framework Elevation?** *
Marcar apenas uma oval.

- ◯ Menos de 6 meses
- ◯ 6 meses a 1 ano
- ◯ Mais de um ano

13. **Qual o seu nível de satisfação para com a framework Elevation?** *
Marcar apenas uma oval.

	1	2	3	4	5	
Muito Insatisfeito	◯	◯	◯	◯	◯	Muito Satisfeito

14. **Defina 5 vantagens ordenada da mais importante para a menos importante que vê com o uso da framework Elevation.** *
Exemplo: 1 - Extensibilidade 2- etc...

_____

_____

_____

_____

_____

15. **Defina 5 desvantagens ordenada da mais importante para a menos importante que vê com o uso da framework Elevation.** *
Exemplo: 1 - Complexidade 2- etc...

_____

_____

_____

_____

_____

16. **Sugestões para o futuro da framework Elevation**
Se pudesse fazer algum tipo de alteração, o que faria?

_____

_____

_____

_____

_____

Figura 32.: Questionário aos colaboradores da PRIMAVERA BSS parte 5