Metodologias de Desenvolvimento para Aplicações Web

(Versão final após defesa)

Vasco Jorge Santos Vieira

Projeto de estágio para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador da UBI: Professor Doutor Paul Andrew Crocker
Orientador da Capgemini: Fábio Nunes Pinheiro

Covilhã, Julho de 2022

Declaração de Integridade

Eu, Vasco Jorge Santos Vieira, que abaixo assino, estudante com o número de inscrição M10676 de/o Mestrado em Engenharia Informática da Faculdade de Engenharias, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 01/06/2022

Assinado por: **Vasco Jorge Santos Vieira**
Num. de Identificação: 14710409
Data: 2022.07.15 14:15:13+01'00'

Agradecimentos

Este relatório marca o fim de uma etapa e com isto gostaria de agradecer mais uma vez ao Professor Doutor Paul Crocker pela oportunidade de realizar este estágio e pelo o acompanhamento durante o mesmo.
Na mesma nota, gostaria de agradecer à *Capgemini* por possibilitar a realização deste estágio e desenvolvimento conjunto. Gostaria de estender os agradecimentos ao meu co-orientador Fábio Pinheiro, bem como ao *Project Manager* (PM) do projeto, Pedro Cabeleira pela orientação, paciência e acompanhamento prestado.
Finalmente, gostaria de agradecer aos meus familiares e amigos por todo o apoio durante o estágio.

Resumo

Este relatório reflete sobre o estágio que decorreu na *Capgemini*, com o propósito de manter e implementar novas funcionalidades na aplicação *web* de um cliente.
Para um melhor entendimento, são descritos os objetivos a atingir. Tendo estes em conta, são apresentadas aplicações semelhantes e alguma literatura relevante com comparações entre diferentes *frameworks* entre outros.
Com o conhecimento de que a aplicação é desenvolvida em *Angular*, são apresentadas as tecnologias que serão usadas ao longo do estágio, sendo também apresentado um mapa cronológico sobre o as tarefas a ser desempenhadas no decorrer do mesmo.
Tendo em conta que a aplicação e desenvolvimento efetuado fazem parte da propriedade intelectual da empresa, é apresentada a arquitetura da aplicação e alguns dos requisitos (funcionais e não funcionais) desta.
Por fim, é descrito o trabalho efetuado por *sprints* e apresentadas as conclusões do estagiário após este estágio.

Palavras-chave

Angular, Capgemini, Aplicação *Web*

Abstract

This document focus on the intership that takes place at Capgemini, with the purpose of maintaining and implementing new features in a client's web application.
The objectives to be achieved are described for a better understanding. With these in mind, similar applications and some relevant literature with comparisons between different frameworks among other topics.
With the knowledge that the application will be developed in Angular, the technologies that will be used throughout the internship are presented, being also established a chronological map about the tasks to be performed throughout the internship.
Taking into account that the application and the development performed are part of the company's copyright, the application architecture and some of its requirements (both functional and non-functional) are introduced.
Finally, the development done by sprints is described and the conclusions of the student after this internship are stated.

Keywords

Angular, Capgemini, Web Application

Conteúdo

Lista de Figuras

Acrónimos e Siglas

ACID	*Atomicity, Consistency, Isolation, Durability*
API	*Application Programming Interface*
CO	*Chapter Owner*
CSS	*Cascading Style Sheets*
CRUD	*Create Read Update and Delete*
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
GUI	*Graphical User Interface*
JS	*Javascript*
JSON	*JavaScript Object Notation*
KB	*Kilobyte*
MB	*Megabytes*
MVCC	*Multiversion concurrency control*
NoSQL	*Not Only SQL*
npm	*Node Package Manager*
PDF	*Portable Document Format*
PM	*Project Manager*
PO	*Product Owner*
PROD	Produção
REST	*REpresentational State Transfer*
SQL	*Structured Query Language*
TL	*Team Leader*
TS	*TypeScript*
VS Code	*Visual Studio Code*
XDCR	*Cross Data Center Replication*

Capítulo 1

Introdução

1.1 Caraterização da Empresa

A *Capgemini* é uma multinacional francesa de consultoria na área da engenharia, tecnologia[Cap21b], tendo sido fundada em 1967 por *Serge Kampf* [Cap21a]. Em 2019 a *Capgemini* anunciou a fusão com o grupo *Altran*, com o propósito de tornar o grupo no líder mundial. Com a fusão, o grupo *ALtran* passou a designar-se *Capgemini Engineering*, contando atualmente com mais de cinquenta mil colaboradores espalhados por mais de trinta países. Desenvolve produtos nas seguintes áreas, sendo líder global em algumas delas:

- Aeronáutica;
- Automóvel;
- Defesa, Espacial e Naval;
- Energia;
- Indústria e Electrónica;
- Finanças e Setor Público;
- Infraestruturas, Transportes e Caminhos de Ferro;
- Semicondutores;
- *Software* e *Internet*;
- Telecomunicações e Media.

1.2 Objetivos do Estágio

O estágio teve como objetivo o desenvolvimento de novas funcionalidades e manutenção de uma aplicação web que já se encontrava em utilização pelos colaboradores da empresa cliente. A aplicação serve para atendimento a clientes, venda de produtos, seguimento de serviços requisitados e interligação a outras plataformas já estabelecidas dentro do grupo de telecomunicações (empresa cliente).
O desenvolvimento foi feito por uma equipa de 6 pessoas. Para desenvolver estas funcionalidades foi utilizado *Angular* para a aplicação *frontend* e *NodeJS* para o *backend*. Foi ainda ser necessário o desenvolvimento /modificações de bilbliotecas proprietárias já existentes e integração de novos microsserviços que foram desenvolvidos por outras equipas.
No final do estágio, pretendia-se que todas a funcionalidades requisitadas, bem como os defeitos eventualmente encontrados estivessem resolvidos, mantendo a aplicação atualizada.

1.3 A aplicação do estágio

A aplicação cujo é foco deste projeto de estágio já se encontra a ser desenvolvida há cerca de 5 anos, sendo o seu período de utilização mais curto 1 ano. Como mencionado no capítulo 1.2, esta aplicação é usada pelos colaboradores de uma empresa provedora de serviços para o atendimento de clientes nas suas lojas.
Com esta os colaboradores podem executar todas as tarefas necessárias dentro das suas funções. Tarefas estas como:

- confirmar a alteração de contratos ou produtos com a assinatura digital do cliente;
- consultar a situação na morada do cliente e ver que produtos se encontram disponíveis na zona;
- consultar historial de compras ou alterações contratuais;
- criação de uma nova ficha de cliente ou alterar os dados de um cliente já existente;
- facilitar a compra de produtos na loja (com criação de relatórios de orçamento, se necessário).

Claro que estas são apenas algumas das funcionalidades da aplicação que os colaboradores utilizam, isto por razões inerentes à propriedade intelectual (capítulo 1.6). Mesmo assim, com esta breve descrição é possível perceber o esperado e o objetivo desta aplicação.

1.4 Enquadramento do Estagiário na *Capgemini*

No decorrer do estágio, o estagiário foi alocado a um grupo de trabalho dentro da área do *frontend*. Dentro do qual estavam inseridos vários colaboradores, este grupo serve para informar sobre a situação atual dos projetos de cada colaborador.

1.5 Enquadramento do Estagiário na Empresa Cliente

Ao longo do estágio, o estagiário integrou uma equipa de desenvolvimento (na empresa cliente) e participou em todas as cerimónias descritas nos capítulos 3.2 e 3.3. Nas reuniões cujo o objetivo fosse a criação de novas tarefas, a opinião do estagiário foi sempre tomada em conta (sendo a tarefa modificada de acordo, caso necessário).
Para atingir os objetivos do estágio (capítulo 1.2) e realizar as tarefas definidas no planeamento (capítulo 4), este teve de se ambientar e utilizar as ferramentas já em uso no projeto (capítulo 3).

1.6 Propriedade Intelectual

Razões relativas ao direito de propriedade intelectual justificam limitações da descrição do trabalho. Com isto boa parte do processo e informação não pode ser partilhada.

1.7 Organização do Documento

De modo a refletir o trabalho feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta a empresa, os objetivos do estágio, a aplicação, enquadramento do estagiário, propriedade intelectual e a organização do documento.

2. O segundo capítulo – **Estado da Arte** – apresenta algumas aplicações com funcionalidades semelhantes, bem como alguma literatura relacionada com o projeto.

3. O terceiro capítulo – **Metodologias e Ferramentas Utilizadas** – apresenta metodologias utilizadas e tecnologias /ferramentsa necessárias para o estágio.

4. O quarto capítulo – **Planeamento** – descreve o plano de trabalho do estágio.

5. O quinto capítulo – **Engenharia de Software** – apresenta a arquitetura utilizada e requisitos funcionais /não funcionais.

6. O sexto capítulo – **Implementação** – descreve as diversas tarefas desempenhadas ao longo do estágio.

7. O sétimo capítulo – **Conclusões e Trabalho Futuro** – apresenta as principais conclusões da fase de planeamento e ambientação ao projeto.

Capítulo 2

Estado da Arte

2.1 Introdução

Neste capítulo são apresentadas algumas aplicações que tiram partido de tecnologias idênticas (capítulo 3) ou usam módulos semelhantes. São ainda discutidos alguns artigos relevantes para este estágio.

2.2 Aplicações Semelhantes

Como foi referido na secção 1.2, a aplicação é usada para atendimento de clientes, venda de produtos, entre outros (capítulo 1.3). Contudo, devido a regras comerciais e regulamentos aplicados pela empresa cliente, não é possível partilhar alguma informação (capítulo 1.6).
Nesta secção são apresentadas 3 aplicações que disponibilizam funcionalidades semelhantes à aplicação em desenvolvimento. Dito isto, estas aplicações têm algo que a aplicação deste estágio não contém. Estas aplicações foram desenvolvidas para serem utilizadas pelo público em geral, enquanto que a aplicação do estágio é utilizada apenas pelos colaboradores (ou seja, uso interno).

2.2.1 *Adobe Acrobat DC*

O *Adobe Acrobat DC* é uma das mais conhecidas aplicações para a visualização e edição de ficheiros *Portable Document Format* (PDF)[Ado21]. Para além das funcionalidades referidas, disponibiliza outras como a assinatura de um ficheiro. A assinatura do ficheiro pode ser efetuada de duas maneiras diferentes:

- Podem ser efetuadas manualmente através da colocação de uma assinatura visual (quer via teclado ou dispositivo tátil);

- Podem ser efetuadas como uma assinatura digital, com recurso a um *token* criptográfico como o certificado digital (este pode ser guardado ou armazenado num *Smart Card* como o cartão de cidadão).

A imagem 2.1 demonstra a assinatura com recurso a um teclado onde, posteriormente, esta é convertida para ser semelhante à caligrafia de uma pessoa.

Figura 2.1: Demonstração da assinatura no *Adobe Acrobat DC* (imagem retirada de um vídeo disponibilizado [Ado21]).

A aplicação que se encontra a ser desenvolvida engloba os dois primeiros tipos de assinatura referidos nesta secção.

2.2.2 *Uber*

A *Uber* é uma empresa que tem como objetivo o transporte de mercadorias e passageiros [Ube21], onde o cliente seleciona o percurso desejado e os condutores aceitam. O mesmo acontece no *Uber Eats*, mas este envolve mais uma entidade, o restaurante que prepara o pedido.
Para a aplicação funcionar de forma impercetível é usada sincronização em tempo real [UKM21]. Isto faz com que, por exemplo, quando um cliente requisita um percurso, este seja automaticamente enviado para os condutores disponíveis que se encontrem na área.

A sincronização em tempo real é usada na aplicação deste estágio, na disponibilização dos vários produtos oferecidos aos clientes, na atualização da informação dos mesmos e outras vertentes desta.

2.2.3 Meo *website*

A Meo é uma empresa prestadora de serviços na área das telecomunicações, venda de produtos e produtos empresariais [Meo21].
O *website* da Meo contém algumas funcionalidades semelhantes à aplicação que se encontra a ser desenvolvida no âmbito deste estágio. Estas funcionalidades incluem a disponibilização de serviços por localização, a disponibilização de pacotes e a apresentação de um catálogo.

Disponibilidade por Localização

A morada do cliente é essencial para a aplicação, o que igualmente acontece no caso da Meo. Com a morada torna-se possível apresentar ao cliente que serviços ou pacotes se encontram disponíveis na sua zona. É possível visualizar o sistema na imagem 2.2.

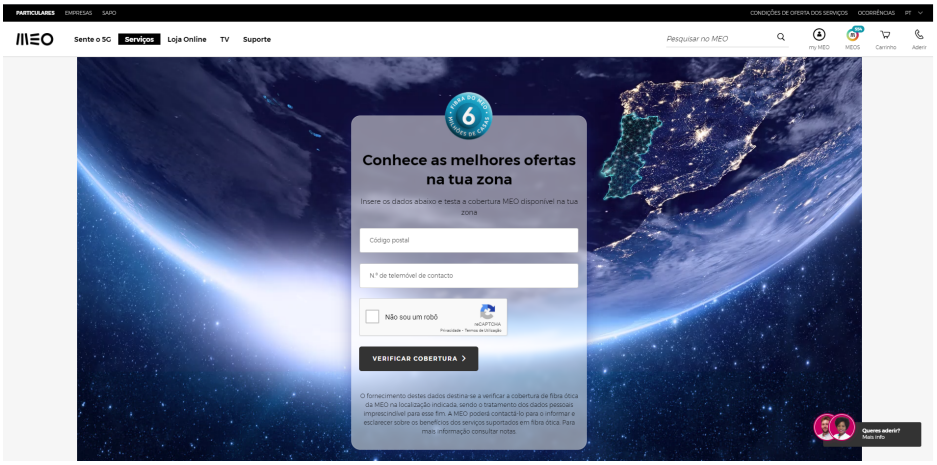

Figura 2.2: Demonstração da pesquisa para produtos disponibilizados por zona (imagem do site da Meo [Meo21]).

Sistema de Pacotes

Esta parte disponibiliza apenas os pacotes que a Meo tem disponível de momento aos utilizadores. Estes podem ser influenciados pela morada (falado na

secção anterior). Sem a informação da morada são apresentados os pacotes de forma geral, sem condicionantes. A apresentação dos pacotes é demonstrada na imagem 2.3.

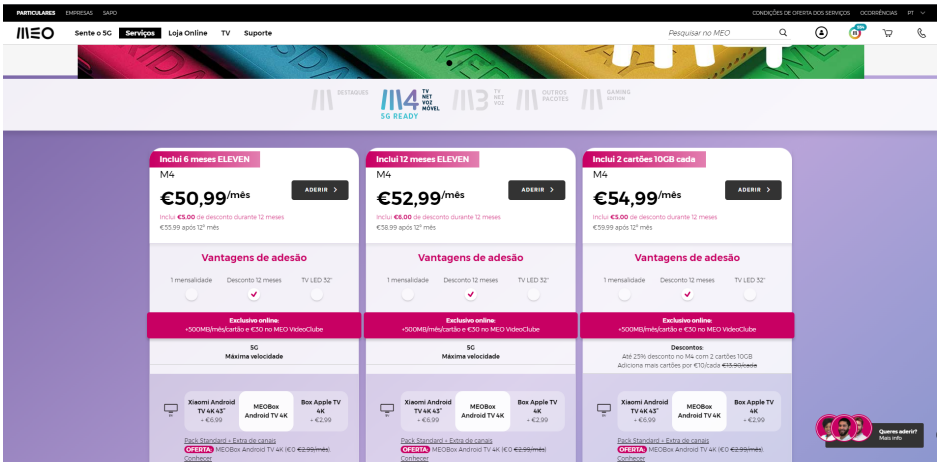

Figura 2.3: Demonstração da apresentação dos pacotes da Meo (imagem do site da Meo [Meo21]).

Catálogo e Promoções

No caso dos catálogos, são apresentados os produtos (como telemóveis, televisões, etc). Neste caso podem ainda ser aplicadas promoções a produtos que resultam de campanhas efetuadas pela Meo, de pontos (que são acrescentados com outros pagamentos) ou derivados da fidelização do cliente (consoante o pacote atribuído a este ou consoante o período no qual usufrui dos serviços da Meo). Isto é demonstrado na imagem 2.4

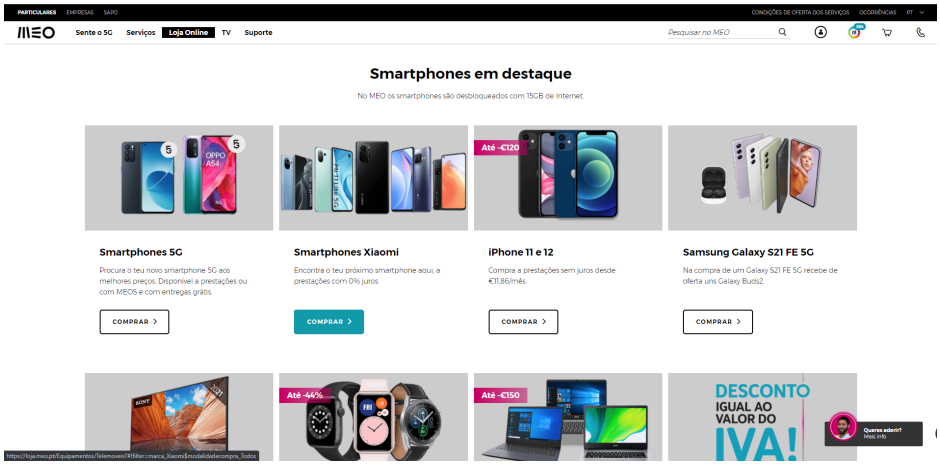

Figura 2.4: Demonstração da apresentação da loja da Meo (imagem do site da Meo [Meo21]).

2.3 Artigos Relacionados

Nesta secção são apresentados e discutidos alguns artigos que foram considerados relevantes (pela empresa) para o desenrolar deste estágio. Os artigos tratam de tecnologias que são usadas, como o *Couchbase*, *Jira* e *Angular* (que é comparado nas secções 2.3.4 e 2.3.5 com outros *frameworks* que começam a ser mais usados no mundo empresarial, embora em projetos mais pequenos).

2.3.1 *A Case Study on Tool Support for Collaboration in Agile Development*

Os autores deste artigo estudaram a vantagem da implementação de ferramentas (como *Jira*, *Confluence*, entre outras, sendo algumas destas descritas no capítulo 3) de auxílio à metodologia *Agile Scrum* [FC20]. Este estudo foi efetuado em colaboração com uma empresa alemã (*Klopotek*), onde foram incorporadas estas ferramentas nas rotinas diárias de trabalho.
De forma a obter resultados para se poder efetuar uma comparação válida, foram recolhidos dados sobre o desempenho da equipa sem recorrer ao uso destas plataformas. Posteriormente, foram introduzidas as plataformas em duas fases para permitir um maior período de adaptação. Durante este período, as plataformas tiveram uma grande adesão por parte dos programadores, das equipas de testes e controlo de qualidade.
Por fim, os autores concluíram que as plataformas mencionadas têm um papel

positivo no aumento do desempenho das equipas, tendo em conta a constante necessidade de passagem de informação. Saliente-se ainda que estas são essenciais para uma rápida colaboração dentro e entre equipas.

2.3.2 *Designing a Bot for Efficient Distribution of Service Requests*

Este artigo incide sobre a criação de um *bot* (diminutivo para *robot*, uma aplicação concebida para desempenhar ações) na ferramenta *Jira* (descrita na secção 3.4.9) [eKB21].
Este tem um comportamento simples: praticamente visa notificar (em diferentes plataformas como *Microsoft Teams* ou email) as partes interessadas na tarefa a cada mudança de etapa no fluxo das tarefas no *Jira*. As etapas em questão são apresentadas na imagem 2.5.
Aquando da criação da tarefa, o *bot* notifica os programadores da criação da nova tarefa. Na passagem para o estado em progresso, notifica o responsável pela equipa e pela tarefa em questão que esta se encontra a ser desenvolvida. No caso de a tarefa se encontrar bloqueada, quer por ser necessário o prévio desenvolvimento de algo, quer falta de informação, informa as mesmas entidades que no caso anterior. Na revisão do código são notificados os programadores da equipa, de forma avisar que a tarefa que encontra terminada e precisa de aprovação. Por último, quando esta é dada como terminada informa o responsável pela equipa e pela tarefa.

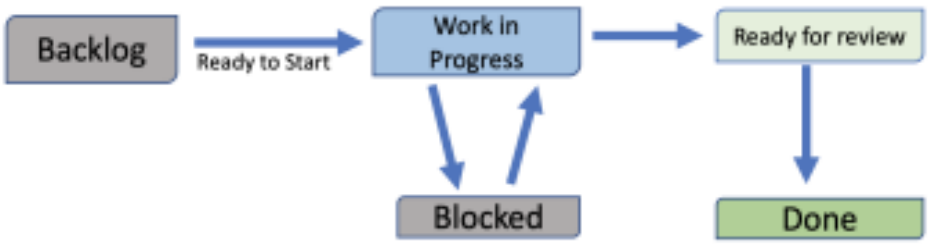

Figura 2.5: Fluxo das tarefas na ferramenta *Jira* (imagem retirada do artigo [eKB21]).

Por fim, os resultados obtidos demonstram uma redução no período necessário para cada tarefa e uma diminuição do intervalo de tempo entre cada etapa, bem como o tempo comunicação. Dito isto, estas conclusões deverão relativizadas, dependendo das equipas (ambiente, dimensão), da hierarquia de comunicação, ente outros fatores.

2.3.3 *Performance evaluation for CRUD operations in asynchronously replicated document oriented database*

Com este artigo os seus autores pretendem efetuar uma comparação do desempenho de vários sistemas de gestão de base de dados, como o *MongoDB*, *CouchDB* e *Couchbase* (que é usado neste aplicação) [COT18].
Estes sistemas de gestão de bases de dados foram escolhidos devido à sua escalabilidade e flexibilidade em relação a *Big Data* e computação na nuvem. A avaliação destes sistemas é baseada nas operações *Create Read Update and Delete* (CRUD), onde é executada uma operação numa instância de base de dados que se encontra distribuída por dois nodos. Posteriormente, os resultados destes 3 sistemas são comparados com sistemas de base de dados relacionais como *Microsoft SQL Server*, *MySQL* e *PostgreSQL*.
Por fim, o *Couchbase* não foi o sistema com maior desempenho (tendo ficado ligeiramente abaixo do *MongoDB*), mas sim o *CouchDB* seguido do *MongoDB* e nos últimos os sistemas relacionais. É ainda importante referir que os sistemas foram desenvolvidos com um propósito específico em mente e o facto de o *Couchbase* se basear num arquitetura partilhada faz com que este seja uma boa escolha quando é necessário um elevado processamento de informação.

2.3.4 *Comparative study of some applications made in the Angular and Vue.js frameworks*

Este artigo visa comparar dois *framewroks*: *Angular* e *Vue.js* [OCN21]. Ambos bastante utilizados hoje em dia para a criação de aplicações. Como os autores referem, as semelhanças são várias, desde a linguagem utilizada ao tipo de aplicações que estes possibilitam criar.
Para o propósito da comparação foi desenvolvida uma simples aplicação *ToDo*, sendo esta essencialmente uma lista de tarefas a realizar. Após o desenvolvimento da aplicação em ambos os *frameworks*, os autores afirmaram que o *Vue.js* é mais acessível para quem está a começar.
Posteriormente, na fase de testes, o desempenho destes foi relativamente semelhante. O *Vue.js* consegue ter tempos de execução menores, o que torna o *Vue.js* uma alternativa preferível. No caso do *Angular*, os tempos de execução mais longos podem ser explicados pelo facto deste acrescentar vários módulos inicialmente, em vez de serem adicionados progressivamente quando necessários. Dito isto, o *Vue.js* perde noutros campos como o suporte, o que é bastante importante para qualquer projeto.
Por fim, os autores afirmam que o desempenho da aplicação desenvolvida usando o *Vue.js* é mais eficiente. No entanto, é importante notar que esta comparação

não tem em conta a possibilidade de tráfego elevado e o uso de *cache*, sendo assim uma comparação simples destes *frameworks*.

2.3.5 *Angular and Svelte Frameworks: a Comparative Analysis*

Semelhante ao artigo anteriormente discutido, este compara dois *frameworks*: o *Angular* e o Svelte[TDT21]. Novamente, esta compara dois *frameworks* que podem ser utilizados para o desenvolvimento de aplicações de tipo semelhantes.
Neste artigo também foi desenvolvida uma aplicação *ToDo* devido ao fácil desenvolvimento. Após a criação da aplicação, é afirmado que no caso do *Angular*, o processo para iniciar o desenvolvimento é mais demorado. No caso do *Svelte*, o iniciar do desenvolvimento é algo direto e simples, mas quando chega a fase de testes é necessário escolher a ferramenta para os fazer e prepará-la, ao contrário do *Angular* que disponibiliza suporte por defeito.
Em relação aos tempos de execução, estes são bastante semelhantes, bem como os resultados obtidos no *lighthouse* (ferramenta da *Google* para medir o desempenho de aplicações *web*). Para além do desempenho, é interessante notar a diferença de tamanho em ambas as aplicações antes e após a compilação. No *Angular*, antes da compilação, a diretoria ocupa 274 MB versus os 11.6 MB do *Svelte*. Isto deve-se ao facto do *Angular* importar um conjunto base de bibliotecas. Já após compilação, o tamanho é 704 KB no *Angular* contra os 106 KB do *Svelte*. Isto deve-se ao facto de no *Svelte* durante a compilação é criado um ficheiro único em *javscript* de todo o código no momento, onde o compilador mantém apenas o código necessário à execução da aplicação.
Por fim, os autores referem que a aplicação implementada é pequena e não permite testar todas funcionalidades base fornecidas pelo *Angular*. Referem ainda que o *Angular* é a melhor opção para aplicações de maior dimensão e maior tráfego. Isto não quer dizer que não seja possível com o *Svelte*, mas seria necessário muito mais tempo para implementar outras funcionalidades.

2.4 Conclusão

Neste capítulo são descritas 3 aplicações que não possuem a mesma finalidade (como dito no capítulo 2.2), mas fazem uso de funcionalidades semelhantes às presentes na aplicação deste estágio (capítulos 2.2.1, 2.2.2 e 2.2.3).
Para além das aplicações, são referidos artigos relacionados com o *Jira* e ainda sobre algumas tecnologias utilizadas neste estágio.

Capítulo 3

Metodologias e Ferramentas Utilizadas

3.1 Introdução

Neste capítulo, são discutidas as metodologias implementadas (secções 3.2 e 3.3), bem como as ferramentas já utilizadas no projeto e utilizadas para a continuação de desenvolvimento e manutenção desta aplicação (secção 3.4). Estas ferramentas foram escolhidas pela empresa cliente e membros presentes na criação da aplicação.

3.2 *Agile /Scrum*

Na primeira metade do estágio foi usada a metodologia *Agile Scrum*. Este processo é englobado *sprints, backlog, daily standup, retrospective, planning* e *review* como é mostrado na imagem 3.1. Para uma melhor e mais otimizada utilização deste método são usadas duas ferramentas, *Jira* e *Confluence*, que são descritas nas secções 3.4.9 e 3.4.8, respetivamente.
Começando pelo *product backlog*, nesta parte são guardadas as *stories* (ou tarefas) a desenvolver. Estas novas funcionalidades só são incorporadas no *sprint backlog* se forem importantes para a situação atual ou a próxima atualização, sob decisão do *Product Owner* (PO), arquitetos e analistas.
No início de cada, *sprint* é feita uma *sprint planning* (reunião de planeamento do *sprint*) para definir os objetivos e quais as *stories* que devem ser desenvolvidas tendo em conta a disponibilidade da equipa. Isto porque a capacidade da equipa é aferida tendo em conta os dias de trabalho de cada membro e quantidade de *story points* feitos em *sprints* anteriores. Calculada a capacidade da equipa disponível para o *sprint*, são escolhidas as *stories* a passar para o *sprint*,

tendo em conta a prioridade e o foco de desenvolvimento atual, sendo ordenadas por prioridade.

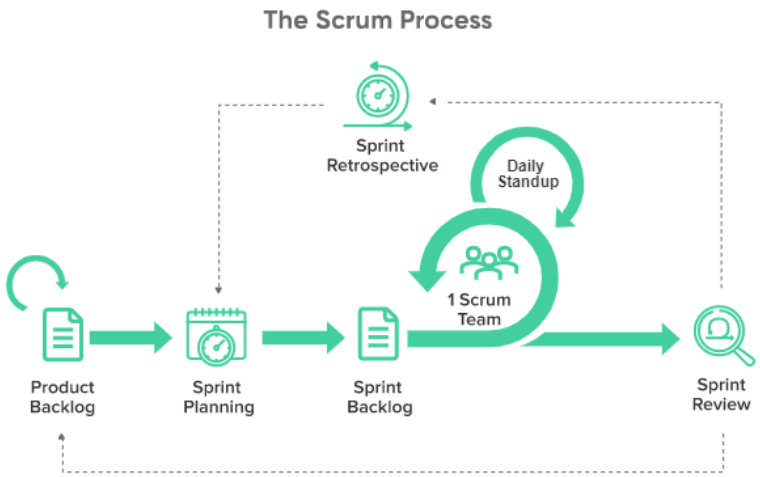

Figura 3.1: Etapas de um processo de Scrum (imagem retirada de [soo21]).

Neste caso, a equipa está a utilizar *sprints* com duração de duas semanas, ou seja, a equipa tem duas semanas para realizar as *stories* definidas no *sprint planning*. Durante o *sprint* são realizados todos os dias de manhã uma reunião (*Daily Standup*) para a equipa ficar ao corrente sobre o trabalho de cada membro. Nesta reunião, cada membro informa o que fez no dia anterior, o que vai fazer durante o dia e se tem problemas ou está bloqueado nalgum assunto (isto pode ajudar a que estes problemas sejam resolvidos mais depressa, com ganho de eficiência). Também durante o *sprint* é feita uma *refinement*, reunião esta com o propósito a criação de novas *stories* (tarefas /funcionalidades) que são guardadas no *product backlog*. Como dito anteriormente, estas são criadas pelo PO, arquitetos e analistas. Onde cada *story* é explicada a lógica do negócio e comportamento esperado quando finalizada. No final da explicação é atribuída a cada *story* um valor que indica a dificuldade/complexidade desta, valor este que é a média da votação de todos os membros da equipa (*developers*).
No final de cada *sprint*, é necessário efetuar uma *review* e *retrospectiva* (retrospectiva). A *review* tem como finalidade rever e analisar o resultado do desenvolvimento feito no *sprint em questão*. A *retrospective* (retrospectiva) serve para analisar a equipa (developers) de forma a identificar o que correu bem, o que

correu mal e sugerir uma ação em conformidade.
Por fim, é feita uma *demo* (demonstração) do desenvolvimento efetuado durante o *sprint*. Para o caso de o desenvolvimento atual não ser suficiente, a demonstração pode ser adiada com o consentimento do PO até conter funcionalidades suficientes.

3.3 Metodologia *Spotify*

Na segunda metade do estágio foi usada a metodologia *Spotify* [Cru22]. Este modelo é em base bastante semelhante ao *Agile /scrum* (capítulo 3.2), sendo que continua a fazer uso do mesmo processo demonstrado na figura 3.1.
A mudança deve-se ao facto deste modelo visa uma maior especialização das equipas, bem como que estas equipas sejam mais autónomas no âmbito do seu trabalho diário. Com isto veio a criação de tribos (*Tribes*), capítulos (*Chapters*) e equipas (*squads*).
As tribos são constituídas por múltiplos capítulos e equipas que trabalham numa das especializações, como por exemplo apoio ao cliente. Dentro destas é discutido todo o assunto relativo à tribo.
Os capítulos são constituídos por todos os membros da mesma especialização dentro da tribo (por exemplo: programadores, analistas, etc.), onde são discutidos assuntos relativos à sua especialização.
Por fim, temos as equipas (*squads*), onde o comportamento /tarefas a desempenhar é idêntico ao explicado em cima (capítulo 3.2).

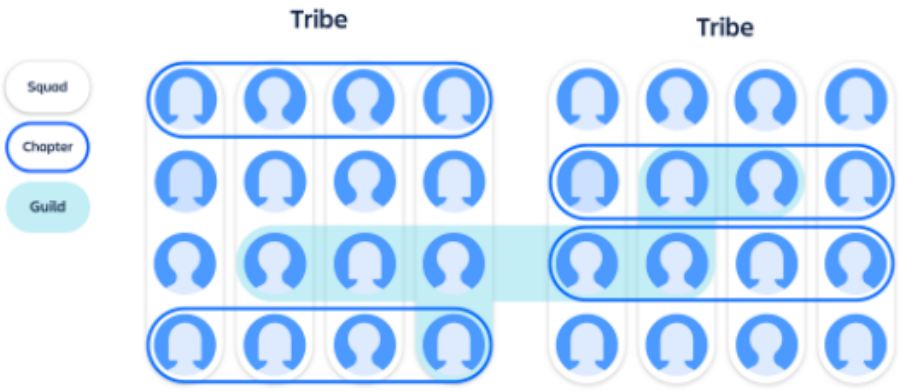

Figura 3.2: Estrutura consoante a nova metodologia (imagem retirada de [Cru22]).

3.3.1 Apreciação do Modelo

Esta mudança na metodologia trouxe alguns obstáculos (uns ainda por superar) para o quotidiano. Como descrito na secção 3.3 esta metodologia visa uma maior especialização das equipas, por outras palavras, equipas com menor dimensão. Com isto, o que originalmente era uma equipa de 6 membros, passou a ser 2 equipas (uma com 4 membros e outra com 2).
Por um lado, esta mudança veio agilizar alguns processos e comunicações visto que são equipas mais pequenas.
Por outro lado, a maior desvantagem desta metodologia é a divisão em equipas, sendo um dos maiores desafios a superar por ambas visto que trabalham na mesma aplicação monolítica.

3.4 Ferramentas Utilizadas

Nesta secção são apresentadas as ferramentas (estas não foram escolhidas pelo estagiário) necessárias no ambiente de desenvolvimento do estágio e algumas extensões que considero essenciais de modo a facilitar o desenvolvimento e os testes que lhe estão associados.

3.4.1 *Angular Framework*

Angular é um *framework* criado pela *Google* para desenvolvimento de aplicações (quer nativas, web, reativas ou progressivas) com uso de *TypeScript* (TS) [Goo21]. O *Angular* permite o desenvolvimento de aplicações que se adaptem a múltiplas plataformas, com o objetivo de manter produtividade e alto desempenho .
O *TypeScript*, é um *superset* de *javascript*, pois fornece todas as funcionalidades do *javascript* e adiciona uma camada que é o sistema de tipos [Mic21a]. Isto faz com que a variável mantenha o mesmo tipo no qual foi definida. O *TypeScript* tem também uma verificação sobre a atribuição feita a uma variável. Estas funcionalidades proporcionam a redução de defeitos, maior robustez e maior segurança.
Nos capítulos 2.3.4 e 2.3.5 temos comparações com outros *frameworks* usados no mundo profissional.

3.4.1.1 *Ngrx*

Ngrx significa *Angular Reactive Extensions* (Extensões Reativas em Angular), esta biblioteca foi desenvolvida pela *Open Collective* para o desenvolvimento de aplicações reativas em *Angular* [col21]. Baseia-se no *Redux* (um *container*

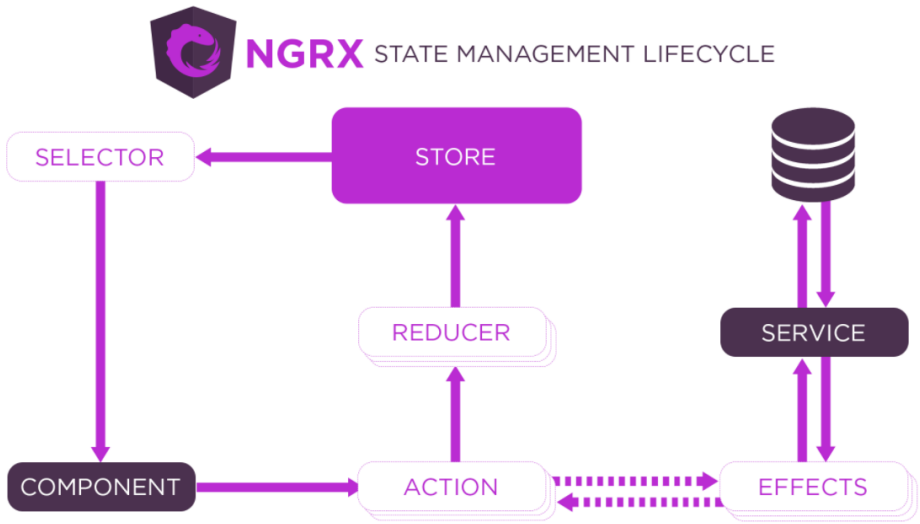

Figura 3.3: Diagrama do funcionamento do *Ngrx* (imagem retirada de [col21]).

de estados para aplicações em *Javascript* [Abr21]) que é uma biblioteca utilizado para passar informação entre os vários componentes da aplicação com recurso a *actions* (ações), *effects* (efeitos), *selector* e reducers (imagem 3.3). Uma *action* é uma instrução emitida com informação (*metadata* /*payload*). Baseada no tipo de ação, a *store* decide a operação a executar. *Reducers* são responsáveis por modificar o estado e retornar um novo estado do objeto com as modificações. *Effects* permitem que algo seja emitido para a *store* através da execução de uma *action*. Por fim, os *Selectors* são as chamadas funções "puras", usadas para obter partes do estado guardado na *store*.

3.4.2 *Visual Studio Code* (VS Code)

O VS Code é um editor de código desenvolvido pela *Microsoft*. Diferencia-se do *Visual Studio Integrated Development Environment* (IDE) pelo facto de não trazer funcionalidades como um ciclo de *debug* e por fazer a *build* de uma aplicação entre outras funcionalidades mais complexas. O VS Code conta com o *git* embutido, *IntelliSense* e um *Debugger* menos complexo[Mic21b]. A grande vantagem deste editor de código é o facto de o podermos completar com recurso às extensões disponibilizadas. Isto pode reduzir o desempenho deste editor de código mas traz funcionalidades que poderão ser indispensáveis ou que facilitem o desenvolvimento. Encontra-se demonstrado na imagem 3.4.

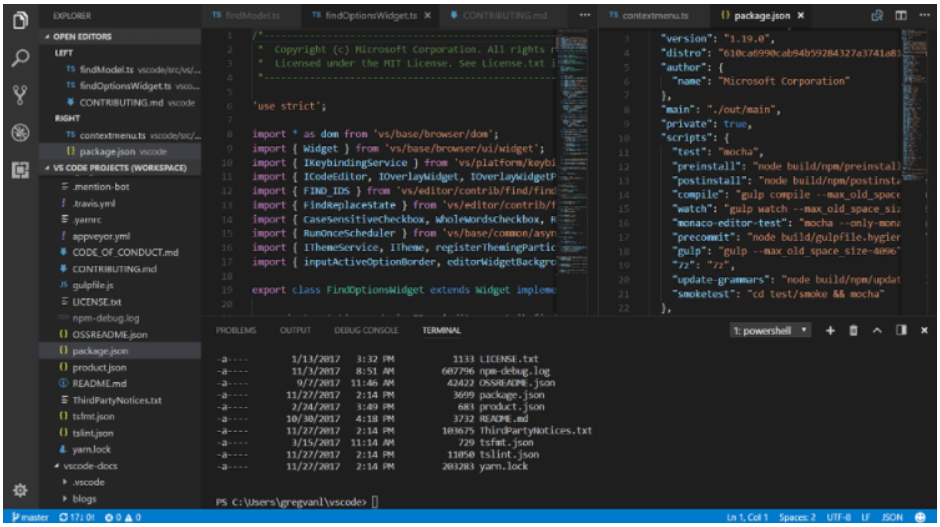

Figura 3.4: Janela de desenvolvimento do VS Code (imagem retirada de [Mic21b]).

3.4.2.1 Extensões

Como referido na secção (3.4.2), o VS Code permite o uso de extensões. Estas são as extensões que considero essenciais para desenvolvimento dentro de um equipa, são as seguintes (estão todas disponíveis no *marketplace* do VS Code):

- ***Angular Essentials*** (feita por John Papa) - esta extensão é um conjunto de extensões essenciais ao desenvolvimento em *angular*;
- ***Angular Snippets*** (feita por John Papa) - esta extensão permite à função *IntelliSense* do VS Code completar com código *TypeScript*;
- ***GitLens*** (feita por GitKraken) - esta extensão adiciona a capacidade de visualizar a autor de certa linha ou bloco de código;
- ***Git Grpah*** (feita por mhutchie) - esta extensão permite ver o *GitFlow* (é explicado na secção 3.4.7) com recurso a gráficos;
- ***Jest Runner*** (feita por firsttris) - esta extensão permite executar um ou múltiplos testes unitários (*Unit Tests*) a partir do próprio ficheiro onde estes são desenvolvidos através do menu de contexto.

3.4.3 *Spectator*

Spectator é uma biblioteca disponibilizada no *github*, que tem como propósito simplificar a construção de testes unitários [Bas21]. Esta é compatível com *Jest*, *Jasmine* e permite testar todos os componentes de uma aplicação. O *Spectator* possibilita a criação de *mocks*, ou seja, uma versão alterada para os testes (isto para não ser necessário atualizar os testes de um componente quando são feitas alterações num serviço).

3.4.4 *Node.js* e *Node Package Manager*

Node.js é um interpretador assíncrono de *javascript*, muitas vezes usado como *backend*, pois consegue manipular grandes quantias de informação em tempo real (é rápido pois guarda informação em *JavaScript Object Notation* (JSON) e por conseguir executar tarefas em simultâneo)[Nod20].
O *Node Package Manager* (npm) é um projeto *open source*, desenvolvido para facilitar a partilha /incorporação de pacotes (nomeadamente módulo ou bibliotecas) entre programadores que usem *javascript* [Man20].

3.4.4.1 *npm* vs *yarn*

O *Yarn* é um gestor de pacotes como o npm, que fornece todas as funcionalidades que o npm e outras mais avançadas [Yar22]. O *Yarn* disponibiliza a funcionalidade "zero instalações", onde todos os módulos ficam descritos para uma melhor otimização. Por outro lado o *Yarn* em problemas com versões antigas do *Node.js* e a instalar módulos nativos.
O npm permite maior facilidade de habituação para programadores que já tenham usado versões anteriores e está otimizado para salvaguardar espaço no disco. Porém, o npm é conhecido por ter um desempenho instável e precisa de ter acesso à internet para instalar os módulos.
As desvantagens do npm parecem ser bastantes em relação às do *Yarn* quando falamos de um projeto atual. Mas o facto de o desenvolvimento ser feito através de uma máquina virtual e o espaço de armazenamento ser fulcral, torna o npm a escolha imediata.

3.4.5 *Postman*

O *Postman* é uma plataforma usada para desenvolver APIs [Pos22]. Esta permite também monitorizar pedidos, pode ser integrada em repositórios para manter o código fonte, entre outras funcionalidades que não são utilizadas pela equipa de desenvolvimento.

No entanto, neste projeto o *Postman* foi utilizado meramente para obter a resposta autal da API, sendo feito tudo através do GUI (figura 3.5). Isto para se poder estruturar os modelos corretamente e também para resolução de defeitos.
O *Postman* permite ainda guardar os pedidos já efetuados, bem como os seus requisitos (*header, body, aut*, etc) por ambiente. Algo importante, tendo em conta diferença dos requisitos.

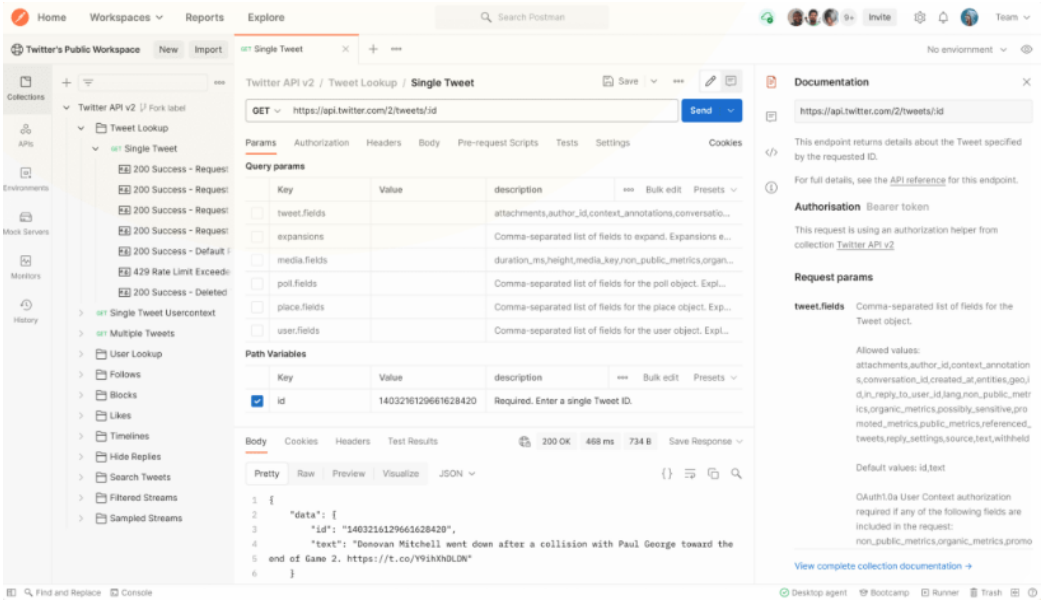

Figura 3.5: Ambiente gráfico da ferramenta *Postman* (imagem retirada de [Pos22].

3.4.5.1 *Postman* vs *Insomnia*

O *Insomnia* como o *Postman* permite mais funcionalidades que as necessárias à equipa de desenvolvimento [Ins22]. Este tem um funcionamento e GUI semelhante ao *Postman* como demonstrado na figura 3.6.
Dito isto, o *Insomnia* não permite o agrupamento de requisitos em grupos como o *Postman*, apenas permite por ambiente (algo aceitável). Mas, por outro lado embora tenha melhor desempenho o *Insomnia* nunca mostra respostas com tamanho superior a 5 MB por defeito (essencial neste projeto).
Esta comparação limita-se a estas duas ferramentas pelo facto de estarem disponíveis para o mesmo sistema operativo (*Windows*). No entanto, existe outra alternativa, o *Paw* [Paw22], apenas disponível para sistema operativo *Mac* e não é gratuita.

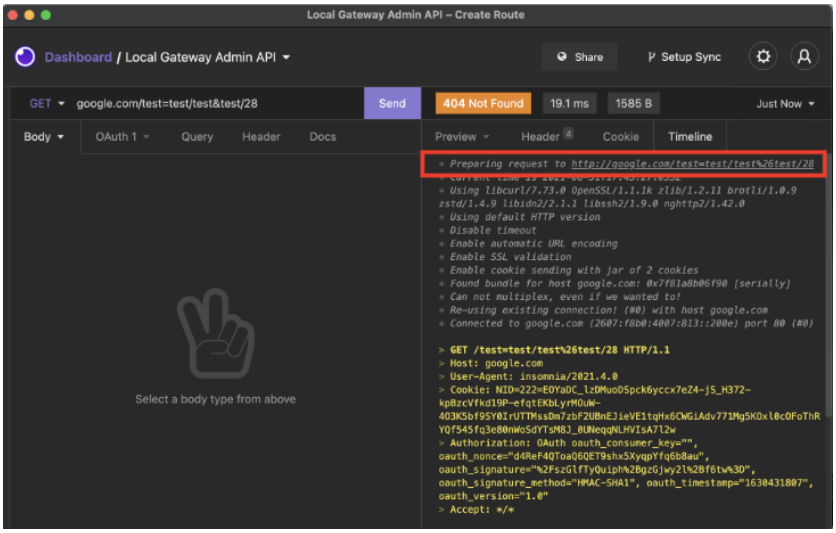

Figura 3.6: Ambiente gráfico da ferramenta *Insomnia* (imagem retirada de [Ins22]).

3.4.6 *Couchbase*

Couchbase é uma base de dados de documentos que não tem uma estrutura para armazenamento dos dados, ou seja, usa *Not Only SQL* (NoSQL). Esta foi concebida com o desempenho, escalabilidade e rápido desenvolvimento em mente [Cou21a].
Neste caso, temos uma configuração de *cluster* de servidores *Couchbase* para uma melhor eficiência e desempenho tendo em conta o elevado número de pedidos esperados.

3.4.6.1 *Sync Gateway*

O *Sync Gateway* é um módulo que tem como propósito sincronizar a informação entre o *cluster* de *Couchbase* e as aplicações (*web* e *mobile*)[Cou21b]. Este módulo permite à aplicação funcionar sem conexão (*offline*). Quando esta é restabelecida o *Sync Gateway* encarrega-se de comparar os objetos e atualiza-los (dando assim controlo e segurança para trabalhar no conceito *offline first*). Sincroniza as aplicações cliente com recurso a REST API (*REpresentational State Transfer Application Programming Interface*). Assegura ainda a autenticação do utilizador e *data-routing* (que o utilizador só tem acesso aos canais que lhe estão atribuídos).

3.4.6.2 *Couchbase* versus *Apache CouchDB*

Para além da comparação efetuada na tabela 3.1, temos no capítulo 2.3.3 entre *Couchbase* e *CouchDB*.

	Couchbase Server	Apache CouchDB
Data Models	JSON document, Key-value	JSON document
Consistency	Forte, inclui transações ACID distribuidas	Eventual
Replication	Master-Master	Master-Master
Locking	Optimistic and pessimistic	Optimistic com MVCC modificado
Query language	Sim, N1QL (SQL for JSON)	Yes, usa uma API derivada do MongoDB
Secondary Indexes	Sim	Sim
Notifications	Sim, Database Change Protocol	Sim, Changes feeds
Services	Data, Query, Index, Full Text Search, Analytics e Eventing	Data, Query e Index

Tabela 3.1: Comparação entre *Couchbase* e *Apache CouchDB* [Cou22].

3.4.7 *Git /Gitlab*

O *git* é simplesmente um gestor de controlo de versões e pode ser usado através da linha de comandos ou de uma interface gráfica (*Graphical User Interface* (GUI)).

Para a equipa poder trabalhar e desenvolver o produto como esperado é necessário um repositório. Sendo neste projeto usado o *GitLab* [Git21]. Neste repositório é onde se encontra armazenado o código fonte da aplicação e os *branches* (ramos) a ser trabalhos no momento. No *GitLab* é onde ainda é feita revisão do código (alusivo à tarefa) implementado pelos programadores.

O fluxo dos *branches* (ramos) do repositório é relativamente semelhante ao mostrado na imagem 3.7. Nesta imagem, conseguimos identificar 4 ramos principais à exceção de *master* (ou *main*), sendo estes *develop*, *feature*, *release* e *hotfix*, onde fica a faltar a nomenclatura *fix* que tem um comportamento semelhante às *features* (funcionalidades).

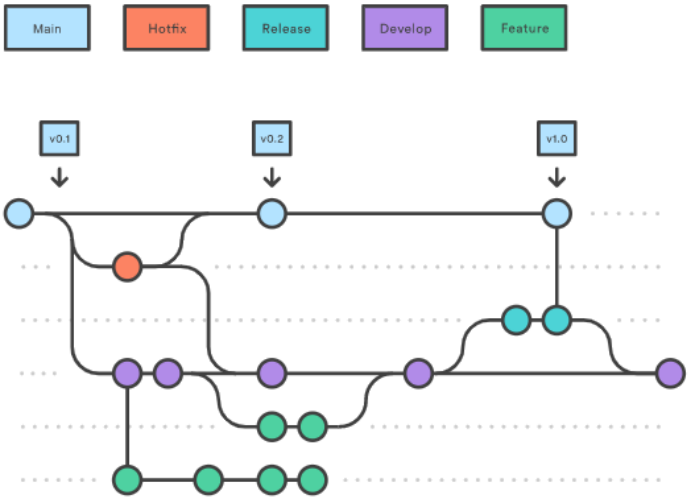

Figura 3.7: Demonstra o fluxo dos ramos no *git* (imagem retirada de [Atl21b]).

3.4.8 *Confluence*

O *Confluence* é uma ferramenta de partilha de informação, onde por norma as organizações criam várias páginas dedicadas a múltiplos assuntos[Atl21a].

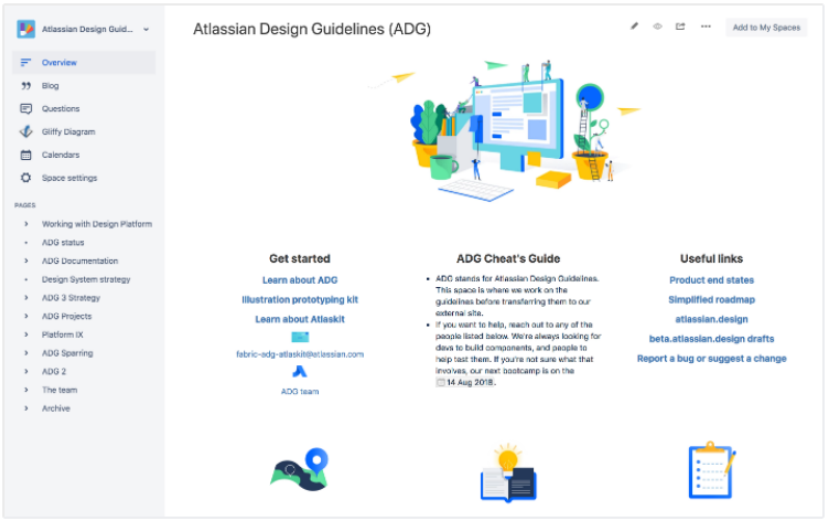

Figura 3.8: Demonstra do ambiente *conflunece* (imagem retirada de [Atl21a]).

Neste caso, o *Confluence* contém informação desde sobre como preparar a máquina para poder correr o projeto (configurar o ambiente), requisitos dos vários

módulos até informação pessoal sobre todos os membros que participam no projeto. Basicamente, é uma enciclopédia sobre o projeto e empresa. O *Confluence* permite a edição em simultâneo de até 12 pessoas na mesma página, algo bastante útil para reuniões.

3.4.9 *Jira*

O *Jira* é uma ferramenta essencial que ajuda a manter a metodologia apresentada na secção 3.2. Através dela consegue-se ter a noção do progresso de cada *story*. Na imagem 3.9 vê-se o *active board* ou *current sprint* onde temos 4 colunas, *TO DO*, *In Progress*, *Code Review* e *Done*. Na coluna *TO DO* são colocadas as *stories* para o *sprint* ou o *sprint backlog*. Na coluna *In Progress* estão *stories* que estão a ser desenvolvidas por um membro da equipa. Na coluna *Code Review* são colocadas as *stories* terminadas, mas que aguardam aprovação de 2 membros para a funcionalidade poder passar para o *branch* de *develop*. Na última coluna (*Done*) são colocadas as *stories* que foram declaradas como feitas ou canceladas.
Na coluna da esquerda podemos ver o *backlog* (imagem 3.9), aqui são armazenadas as *stories* que se encontram no *product backlog*.
Esta aplicação permite aos membros da equipa, bem como ao *Project Manager* (PM) e *Team Leader* (TL), uma melhor gestão e eficiência.

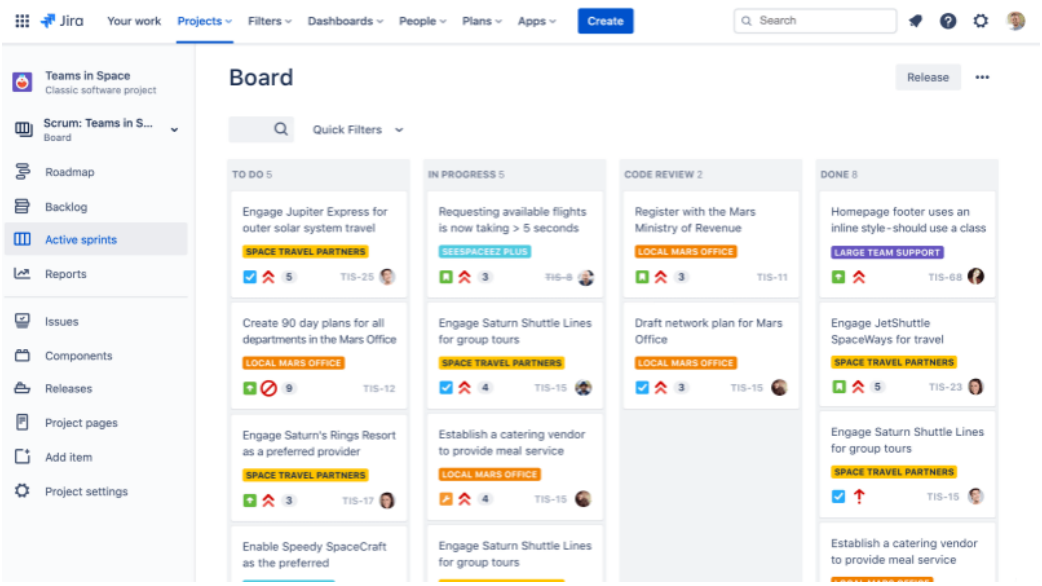

Figura 3.9: Demonstração do ambiente *Jira* (imagem retirada de [Atl21c]).

3.4.10 *Target Process*

A mudança para a metodologia apresentada no capítulo 3.3, trouxe também a substituição da ferramenta *Jira* (capítulo 3.4.9) pelo *Target Process* [App22].

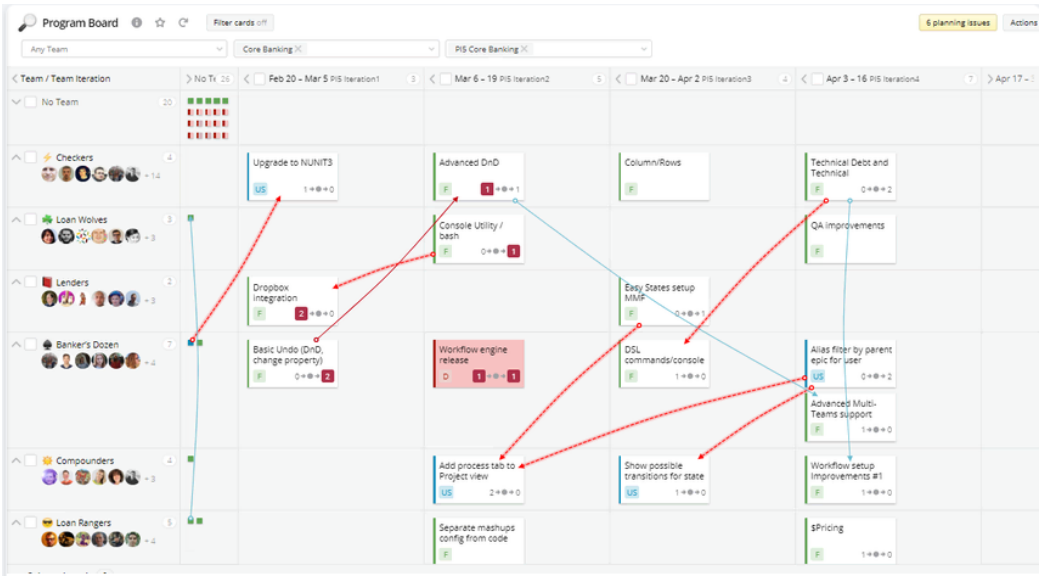

Figura 3.10: Demonstração do ambiente *Jira* (imagem retirada de [App22]).

O *Target Process* permite fazer todas funcionalidades que o *Jira* (descritas no capítulo 3.4.9) em relação às tarefas a desempenhar. No entanto a interface é bastante diferente como mostra a figura 3.10.

Tendo isto em conta, o *Target Process* permite mostrar de forma direta e clara as dependências das tarefas atuais ou futuras, algo bastante importante para um melhor planeamento do lançamento de novas versões do *software* que se encontra a ser desenvolvido.

Para além das funcionalidades referidas o *Target Process* fornece a possibilidade de gerir orçamentos alocados aos projetos (figura 3.11), fazendo com que os PO e *Chapter Owner* (CO) usem a mesma ferramenta que as equipas de desenvolvimento.

Estas funcionalidades adicionais são fulcrais para agilizar a passagem de informação entre as equipas de gestão e desenvolvimento. Esta passagem de informação permite se sejam tomadas decisões de forma mais eficiente e célere.

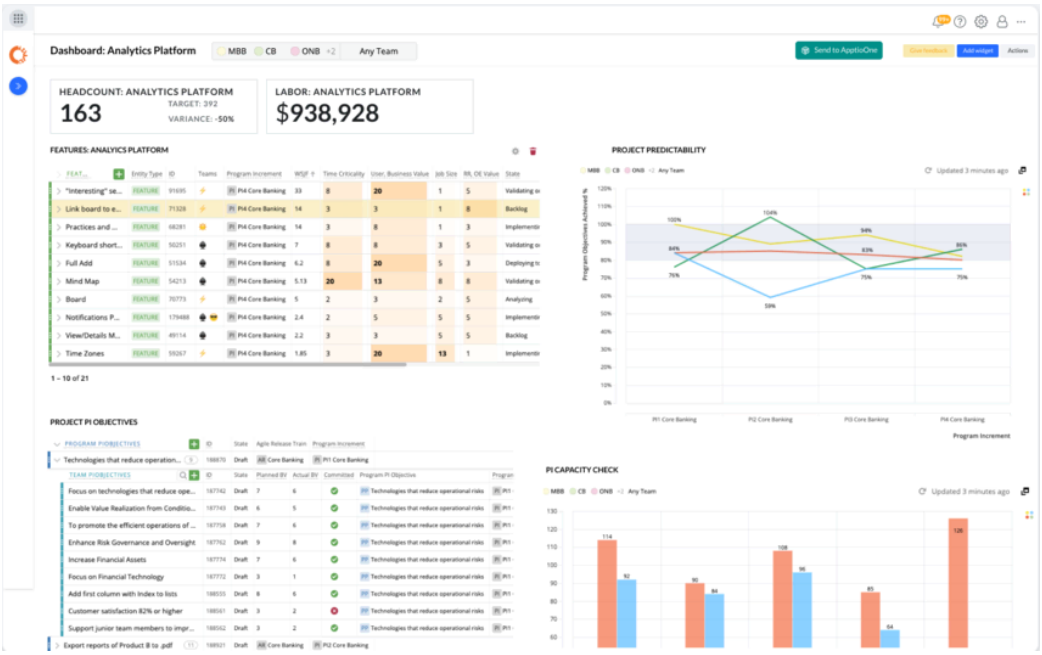

Figura 3.11: Demonstração de um dos ambientes disponiveis aos PO e CO (imagem retirada de [App22]).

3.5 Conclusão

A integração num projeto implica uma certa adaptação aos membros que nele constam, bem como às ferramentas utilizadas. Como referido na introdução estas ferramentas foram escolhidas pela empresa e membros originais (capítulo 3.1).
O *Confluence* (3.4.8) e o *Jira* (3.4.9) requereram maior habituação pelo facto que ser necessária a atualização do estado das *stories* a cada alteração.
Por fim, a adaptação que requereu maior esforço foi a introdução do *Ngrx* (3.4.1.1) e do *Spectator* (3.4.3) por ser algo completamente novo num ambiente a que já estava acostumado.

Capítulo 4

Planeamento e Trabalho Realizado

4.1 Introdução

Neste capítulo foram definidas as tarefas que foram executadas durante o estágio. Isto segundo a metodologia apresentada no capítulo 3.2, as tarefas em concreto são definidas perto do *sprint*. Para além da apresentação das mesmas, foi ainda disponibilizado um mapa cronológico deste estágio.

4.2 Definição de Tarefas

O estagiário teve de desempenhar múltiplas tarefas dentro da empresa onde decorreu o estágio, bem como para a empresa cliente (que recorreu aos serviços da *Capgemini*).
De modo geral, as tarefas executadas para a empresa cliente durante o estágio foram as seguintes:

- **Tarefa 1** - Integração na equipa de trabalho - apresentação a todos os membros que integram /interagem a equipa de desenvolvimento;
- **Tarefa 2** - Ambientação ao projeto e ferramentas utilizadas - estudo das ferramentas apresentadas no capítulo 3.4 e efetuar desenvolvimento acompanhado;
- **Tarefa 3** - Implementação de novas funcionalidades e testes unitários - desenvolver novas funcionalidades de acordo com tarefas (lógica de negócio e requisitos técnicos) e criação dos testes unitários adequados ao código;

- **Tarefa 4** - Resolução de defeitos - resolução de defeitos encontrados pela equipa de testes e adaptação dos testes unitários (no caso de não existirem devem ser criados);

- **Tarefa 5** - Elaboração do relatório de estágio.

No capítulo 6 é descrito o trabalho efetuado ao longo dos *sprints* (segundo a metodologia utilizada) as tarefas as executadas.
Para além das tarefas referidas, o estagiário teve ainda de participar em cerimónias (nome dado às reuniões no âmbito do cliente) cujo o objetivo é manter a *Capgemini* ao corrente da situação atual e dar sugestões /opinião sobre procedimentos. Este teve ainda de efetuar avaliações a novos colaboradores que se encontravam em processo de recrutamento na empresa /projeto.

4.3 Plano de trabalho

De acordo com as tarefas descritas nos capítulos 4.2 e 6 é possível efetuar um mapa cronológico do estágio efetuado. O mapa pode ser visto na figura 4.1.

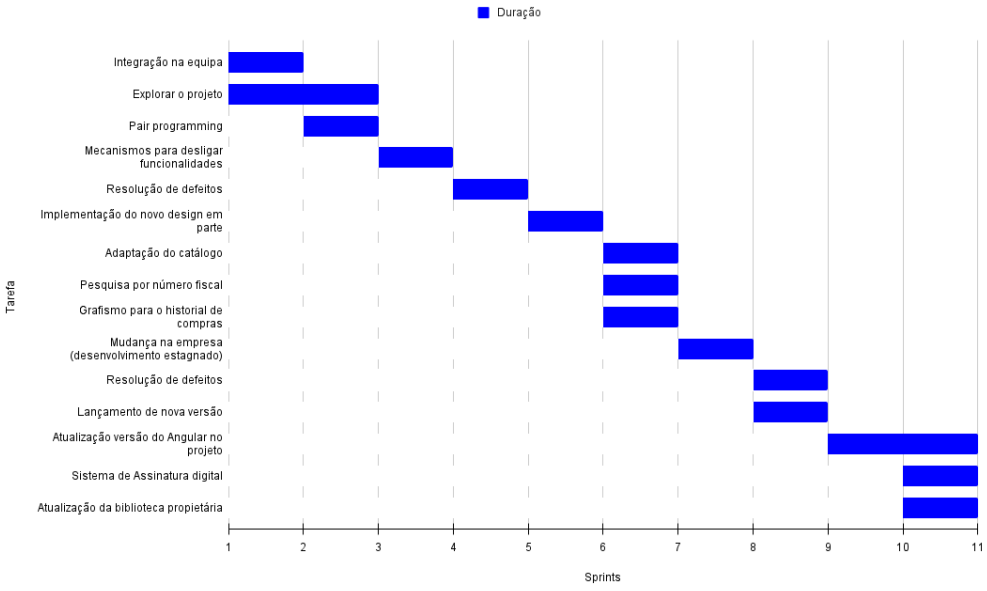

Figura 4.1: Mapa cronológico tendo em conta as tarefas desempenhadas.

4.4 Responsabilidade do Estagiário

Tendo em conta o enquadramento do estagiário (descrito nos capítulos 1.5 e 1.4) e as tarefas executadas (capítulo 4.2).
Da empresa cliente o estagiário tem a seu cargo a implementação de código funcional de qualidade, suportar os membros da equipa. Ao longo do estágio este teve um acréscimo de responsabilidades, tendo se participar no lançamento de novas versões da aplicação e reuniões onde é determinada a lógica de novas funcionalidades.
Para *Capgemini*, o estagiário teve ainda de desempenhar entrevistas de *soft skills* aos novos colaboradores que fossem integrar este projeto. Sendo o objetivo apurar os colaboradores com o perfil mais adequado ao projeto.

4.5 Conclusão

A concluir, é de notar que o plano de trabalho já se encontrava estabelecido, elaborado tendo em conta a metodologia em uso na empresa antes das alterações e mudança na organização. Mesmo assim, serviu como referência para o desenrolar do estágio.
É ainda importante notar que, como referido ao longo deste capítulo, a definição das tarefas foi feita conforme a metodologia apresentada na secção 3.2, pelo que não se mostra possível pormenorizar as tarefas a realizar.

Capítulo 5

Engenharia de Software

5.1 Introdução

Neste capítulo, é apresentada a arquitetura da aplicação no capítlo 5.2), bem como alguns dos requisitos (funcionais e não funcionais) da mesma no capítulo 5.3).

5.2 Arquitetura da Aplicação

Como demonstrado na figura 5.1, o esquema é divido em duas partes *DR* e Produção (PROD), sendo *DR* um ambiente *backup /pré-produção* e PROD o principal ambiente de produção.
O ambiente *DR* é maioritariamente utilizado para executar os últimos testes antes de ser lançada uma nova versão da aplicação. Porém, em alturas de elevado tráfego o controlador *3DNS* distribui os acessos entre os dois ambientes.
Estes dois ambientes fazem uso de *clusters* com servidores de *Couchbase* e utilizam *Cross Data Center Replication* (XDCR). Com isto é possível replicar duplicar informação entre ambos os *clusters*, bem como entre a fonte do *bucket* e o destino. Desta forma obtemos, um melhor desempenho no acesso da informação e proteção contra falhas em *data-centers*.
Dito isto, a arquitetura da aplicação é a mesma independente do ambiente. É possível observar na figura 5.1 que temos o *frontend*, que contém o código fonte (CSS, HTML e JS ou TS) e recebe o resultado dos *endpoints* via uma *reverse proxy*. O *frontend* comunica depois com o *backend* e *sync gateway* (capítulo 3.4.6.2).
No *backend* temos a camada de *middleware* que trata da comunicação com o *frontend* e gere outras APIs, sendo 3 delas: pesquisa, sistema de fila e autenticação.

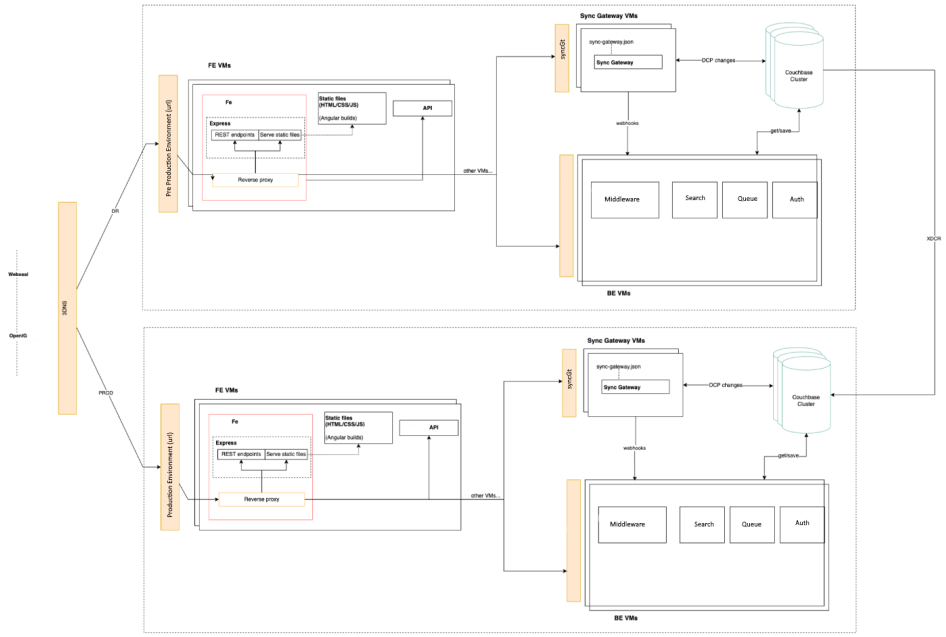

Figura 5.1: Ilustração da arquitetura da aplicação.

5.3 Requisitos

Nesta secção, são descritos alguns dos requisitos desta aplicação, isto devido ao seu elevado número e pela situação descrita no capítulo 1.6. Dito isto, os requisitos (funcionais e não funcionais) já se encontravam definidos aquando da integração do estagiário na equipa. Alguns dos mais significativos dos requisitos funcionais e não funcionais são descritos nas tabelas 5.1 e 5.2 respetivamente.

Tabela 5.1: Alguns dos requisitos funcionais da aplicação.

Requisitos Funcionais	
RF	Descrição
1	O utilizador deve possuir a habilidade de se autenticar na aplicação.
2	A aplicação deve ter um sistema de monitorização (de forma a saber quais as áreas mais utilizadas).

Tabela 5.1 – Continuação da página anterior

RF	Descrição
3	A aplicação deve permitir ao funcionário alterar os dados do cliente.
4	A aplicação deve pedir a assinatura do cliente aquando de uma compra (de acordo com a regulamentação do país).
5	A aplicação deve abrir uma nova janela no navegador com a página de pesquisa de números de contribuinte (página do sistema legal) quando este é clicado (aplicável a pessoas coletivas).
6	A aplicação deve redirecionar para a aplicação *backup* quando certas APIs falham.
7	A aplicação deve mostrar os historial de compras do cliente.
8	A aplicação deve mostrar o estados dos pedidos efetuados.
9	A aplicação deve mostrar um catálogo de acordo com a morada do cliente que está no sistema.
10	A aplicação deve notificar quando um novo produto fica disponível para o cliente.
11	A aplicação deve seguir o sistema de design da empresa.
12	A aplicação deve ter 3 linguagens disponíveis.
13	A aplicação deve mostrar ao funcionário quantos clientes se encontram em espera na loja.
14	A aplicação deve mostrar ao funcionário o estado das APIs utilizadas na aplicação.
15	A aplicação deve permitir criar um novo cliente.
16	A aplicação deve permitir criar um novo pedido.
17	A aplicação deve identificar se a morada do cliente tem a possibilidade de ter um contrato de internet por fibra.
18	A aplicação deve poder mostrar os contratos ativos do cliente.
19	A aplicação deve apresentar janelas flutuantes (*pop-ups*) em caso de erro.
20	A aplicação deve poder redicionar os funcionários para outras plataformas internas.
21	A aplicação deve detetar o ambiente de utilização (desenvolvimento, produção, etc).
22	A aplicação deve ter um sistema de notificações para o funcionário.
23	A aplicação deve ter um sistema de validação de dados.

Requisitos Não Funcionais	
NRF	Descrição
Capacidade	O sistema tem de ser capaz de fornecer as funcionalidades necessárias para os colaboradores trabalharem.
Compatibilidade	A aplicação tem de poder executar pelo menos no *Google Chrome*, *Microsoft Edge*, *Mozilla Firefox*.
Desempenho	A aplicação não deve demorar mais que x tempo a executar tarefas.
Escalabilidade	A aplicação tem de aguentar cargas (tráfego) elevadas.
Fiabilidade /Disponibilidade	A aplicação não deve ter falhas e deve poder ser acedida em durante x horário.
Manutenção	A manutenção /resolução de problemas não deve ultrapassar x tempo.
Segurança	A aplicação guarda informação sensível, como tal, o sistema e informação devem protegidos contra ataques.
Usabilidade	A plataforma deve ser de fácil utilização pelo colaborador.

Tabela 5.2: Alguns dos requisitos não funcionais da aplicação.

5.4 Conclusão

Neste capítulo foi apresentada e descrita a arquitetura da aplicação, bem como os requisitos funcionais e não funcionais da aplicação. A informação presente neste capítulo é relativamente limitada pela quantidade de requisitos a apresentar e pela propriedade intelectual (como explicado no capítulo 1.6). Mesmo assim os requisitos apresentados conseguem fornecer informação substancial sobre a aplicação.

Capítulo 6

Implementação

6.1 Introdução

Neste capítulo é descrito o trabalho desempenhado, é importante notar que o desenvolvimento não é descrito na totalidade (como explicado na secção 1.6) e que os *sprints* têm a duração de duas semanas (como descrito no capítulo 4).

6.2 *Sprint 1*

Como explicado no capítulo 4 o primeiro *sprint* foi dedicado à apresentação da equipa e habituação às ferramentas descritas na secção 3.4.

Este foi também utilizado para explorar a aplicação, isto pelo tamanho da aplicação e por esta se encontrar a ser desenvolvida há cerca de 5 anos.

6.3 *Sprint 2*

O segundo *sprint* foi tomado em grande parte por sessões passagem de conhecimento. Estas sessões foram compostas por conhecimento técnico e lógica do negócio (da empresa cliente), desempenhadas por um programador da equipa e um PO.

Durante este *sprint* comecei a desenvolver algumas tarefas com outro programador (pair-programming) com o propósito de agilizar a passagem de conhecimento.

6.4 *Sprint 3*

No terceiro *sprint* comecei a desenvolver sem acompanhamento, maioritariamente mecanismos para haver a possibilidade de desligar um módulo ou funcionalidade (mecanismo *switch off*). Estes mecanismos são úteis para o caso de funcionalidade estar acabada mas o micro-serviço que a utiliza não, a funcionalidade é desligada.

6.5 *Sprint 4*

Este *sprint* foi dedicado por completo à resolução de defeitos encontrados pela equipa de testes. Isto deve-se ao facto de ser lançada uma nova versão no final deste *sprint*.

6.6 *Sprint 5*

Durante este *sprint* a aplicação sofreu algumas mudanças a nível estético, tendo sido eliminado o gradiente visível na figura 6.2. O Resultado desta alteração é percetível nas figuras 6.1 e 6.3.

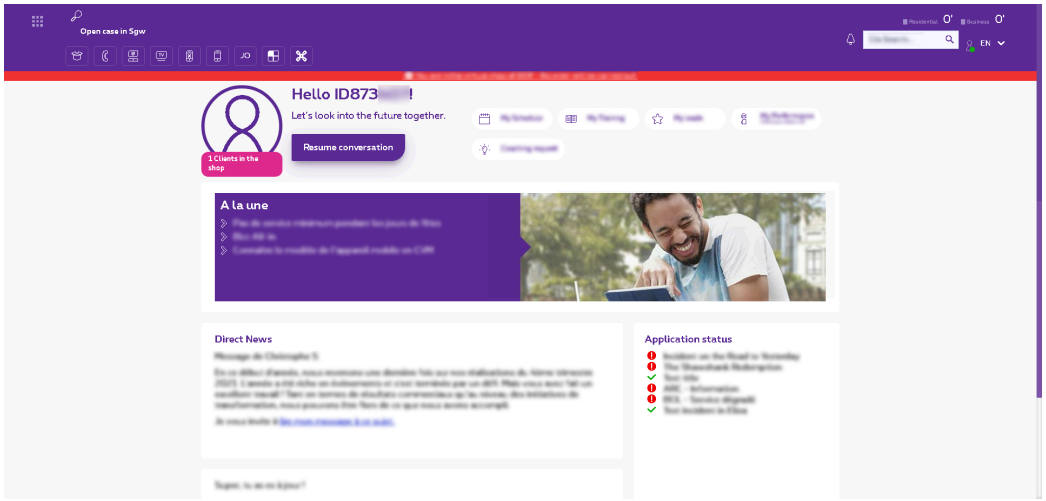

Figura 6.1: Demonstração da aplicação.

Nestas figuras é ainda possível identificar algumas funcionalidades mencionadas neste relatório, como o sistema de tradução, a ficha do cliente e a monito-

rização compras efetuadas.

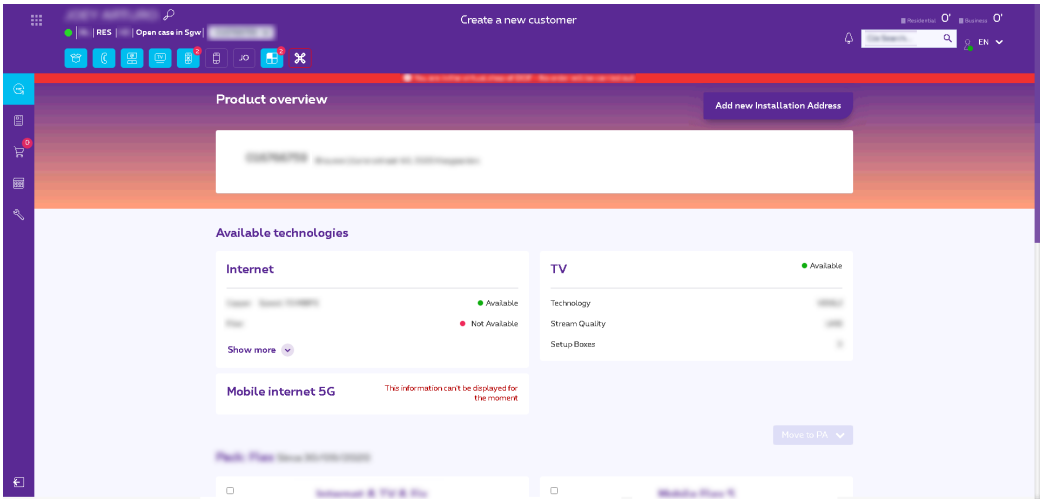

Figura 6.2: Demonstração da aplicação.

Na figura 6.2 temos a informação que é utilizada para as funcionalidades semelhantes às descritas no capítulo 2.2.3 poderem funcionar corretamente.

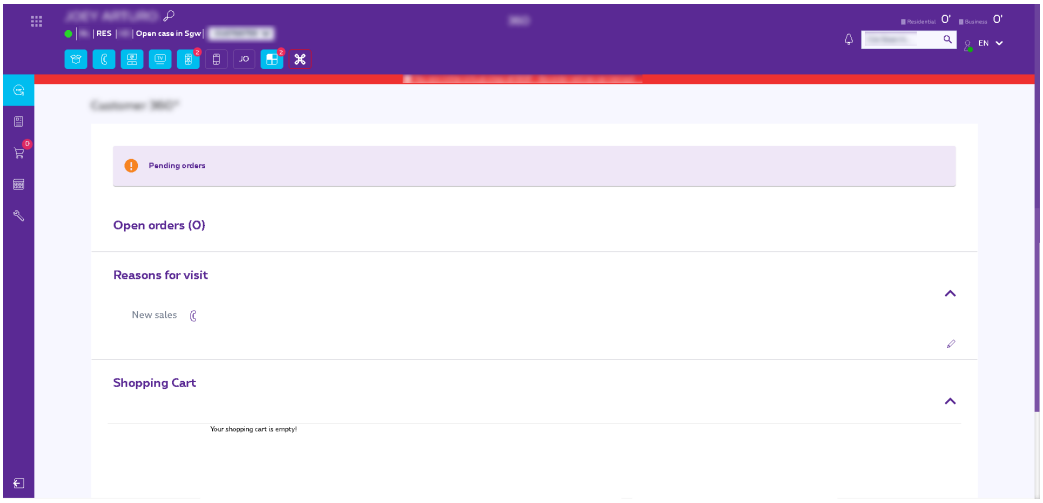

Figura 6.3: Demonstração da aplicação.

6.7 *Sprint 6*

O sexto *sprint* foi dedicado para o desenvolvimento de novas funcionalidades na aplicação. Algumas das funcionalidades incluem adaptações ao catálogo disponível na aplicação, pesquisa do número de identificação fiscal num plataforma jurídica e grafismo para um melhor seguimento do progresso dos pedidos (compras /contratos).

6.8 *Sprint 7*

Neste *sprint* o desenvolvimento da aplicação estagnou devido à mudança na organização da empresa, onde se passou da metodologia *Agile /Scrum* (secção 3.2) para a metodologia *Spotify* (secção 3.3). Devido às grande mudanças da organização e metodologia, este *sprint* foi utilizado para habituação.

No entanto, algum esforço da equipa foi também direcionado para a resolução de defeitos para preparar o lançamento da nova versão.

6.9 *Sprint 8*

Neste *sprint* houve o lançamento de uma nova versão para produção onde o estagiário foi o programador designado para supervisionar o procedimento. Este processo é totalmente automatizado, mas a presença de um programador é necessária para caso de serem encontrados defeitos que bloqueiem o funcionamento essencial da aplicação.

O Resto do *sprint* foi utilizado para adaptação e desenvolvimento de novas funcionalidades.

6.10 *Sprint 9 e 10*

Estes *sprints* finais foram utilizados para adaptar e desenvolver novas funcionalidades como: redesenhar o sistema da assinatura digital para acomodar vários tipos de contratos, atualizar a biblioteca partilha (biblioteca que contém funcionalidades comuns a vários projetos da empresa) e atualização da versão do *angular* usada no projeto.

6.11 Conclusão

De acordo com o capítulo 4 e as metodologias apresentadas no capítulo 3, a definição das tarefas foi feita nas *refinements*. Mesmo não estando totalmente familiarizado com o projeto (e aplicação), nem tendo conhecimento das tarefas, tal não impediu o seu desenvolvimento.
Para além dos pontos referidos, a mudança na organização foi o maior obstáculo a superar e continua a ser um dos maiores riscos. Dito isto, existem outros fatores mencionados no capítulo 3.3.1 que continuam a atrasar o desenvolvimento e põem em risco o lançamento de futuras versões.

Capítulo 7

Conclusão

7.1 Conclusões Principais

Este relatório foi elaborado de forma a apresentar o projeto de estágio com a *Capgemini Engineering* com o intuito do contínuo desenvolvimento da aplicação web, que já se encontra em utilização nas lojas da empresa.

Ao longo deste documento foi apresentada a empresa, os objetivos atingir, bem como, a descrição e comparação de ferramentas utilizadas ao longo deste estágio. Ferramentas estas já utilizadas pela equipa pelo que não houve opção de escolha por parte do estagiário. Dito isto, foram também apresentadas algumas aplicações com funcionalidades semelhantes já inseridas no mercado. Estas aplicações embora pouco se relacionem com aplicação do cliente, serviram de referência para demonstrar o que se pode realizar com as mesmas tecnologias ou funcionalidades (para fins diferentes).

O plano de trabalho elaborado no início deste projeto de estágio foi efetuado de acordo com as tarefas expectáveis e metodologia utilizada pela empresa no momento. Porém, no decorrer do estágio o leque de tarefas a desempenhar foi alargado, tendo a empresa cliente sofrido também alterações na metodologia utilizada.

Por fim, bastaria afirmar que os objetivos propostos foram atingidos e que a aplicação continuou a evoluir. Contudo, estaria a omitir os desafios e obstáculos que o mundo profissional por vezes impõe. Algo que foi possível ultrapassar em parte graças à equipa que atualmente integro, sendo também necessário abordar estes desafios com motivação e iniciativa.

7.2 Trabalho Futuro

A aplicação sobre a qual este projeto de estágio incide sofre constantes evoluções e modificações (devido a alterações dos requisitos).
Posto isto, se o estagiário se mantiver no projeto continuará a desempenhar as funções descritas neste relatório. Estas funções incluem a participação nas cerimónias de planeamento e criação de tarefas, bem como a participação em lançamento de novas versões e incorporação de novas funcionalidades.

Bibliografia

[Abr21] Dan Abramov. Redux, 2021. [Online] `https://redux.js.org/`. Último acesso a 19 de Novembro de 2021.

[Ado21] Adobe. Como preencher e assinar um formulário PDF com a Adobe., 2021. [Online] `https://www.adobe.com/pt/acrobat/how-to/fill-sign-pdf-forms-electronically.html?promoid=4JW79FH4&mv=other`. Último acesso a 15 de Dezembro de 2021.

[App22] Apptio. *Apptio Targetprocess*, 2022. [Online] `https://www.apptio.com/products/targetprocess/`. Último acesso a 29 de Março de 2022.

[Atl21a] Atlassian. Confluence, 2021. [Online] `https://www.atlassian.com/software/confluence`. Último acesso a 25 de Outubro de 2021.

[Atl21b] Atlassian. Gitflow Workflow, 2021. [Online] `https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow`. Último acesso a 26 de Novembro de 2021.

[Atl21c] Atlassian. Jira, 2021. [Online] `https://www.atlassian.com/software/jira`. Último acesso a 22 de Outubro de 2021.

[Bas21] Netanel Basal. <Spectator>, 2021. [Online] `https://github.com/ngneat/spectator`. Último acesso a 21 de Novembro de 2021.

[Cap21a] Capgemini. A History of our Founder, Serge Kampf, 2021. [Online] `https://www.capgemini.com/company-profile-key-figures/a-history-of-our-founder-serge-kampf/`. Último acesso a 13 de Dezembro de 2021.

[Cap21b] Capgemini. SOBRE A CAPGEMINI ENGINEERING, 2021. [Online] `https://capgemini-engineering.com/pt/pt-pt/`

quem-somos/sobre-a-capgemini-engineering/. Último acesso a 13 de Dezembro de 2021.

[col21] Open collective. Getting Started, 2021. [Online] https://ngrx.io/guide/store. Último acesso a 15 de Novembro de 2021.

[COT18] Alexandru Boicea e Ion Bucur Ciprian-Octavian Truică, Florin Rădulescu. *Performance evaluation for CRUD operations in asynchronously replicated document oriented database*, 2018. [Online] https://arxiv.org/abs/1806.04761. Último acesso a 27 de Dezembro de 2021.

[Cou21a] Couchbase. Couchbase, 2021. [Online] https://docs.couchbase.com/server/current/learn/architecture-overview.html. Último acesso a 10 de Novembro de 2021.

[Cou21b] Couchbase. Sync Gateway, 2021. [Online] https://www.couchbase.com/products/sync-gateway. Último acesso a 11 de Novembro de 2021.

[Cou22] Couchbase. *Couchbase and CouchDB Differences*, 2022. [Online] https://www.couchbase.com/comparing-couchbase-vs-couchdb. Último acesso a 17 de Março de 2022.

[Cru22] Mark Cruth. *Discover the Spotify model*, 2022. [Online] https://www.atlassian.com/agile/agile-at-scale/spotify. Último acesso a 11 de Abril de 2022.

[eKB21] Arkadip Basu e Kunal Banerjee. *Designing a Bot for Efficient Distribution of Service Requests*, 2021. [Online] https://arxiv.org/abs/2103.05970. Último acesso a 23 de Dezembro de 2021.

[FC20] Marco Losavio e Filippo Lanubile Fabio Calefato, Andrea Giove. *A Case Study on Tool Support for Collaboration in Agile Development*, 2020. [Online] https://arxiv.org/abs/2004.00289. Último acesso a 20 de Dezembro de 2021.

[Git21] GitLab. GitLab, 2021. [Online] https://about.gitlab.com/. Último acesso a 29 de Outubro de 2021.

[Goo21] Google. Angular, 2021. [Online] https://angular.io/. Último acesso a 17 de Novembro de 2021.

[Ins22] Insomnia. *Insomnia*, 2022. [Online] `https://insomnia.rest/`. Último acesso a 1 de Maio de 2022.

[Man20] Node Package Manager. About npm, 2020. [Online] `https://www.npmjs.com/about`. Último acesso a 1 de Junho de 2022.

[Meo21] Meo. Meo, 2021. [Online] `https://www.meo.pt/`. Último acesso a 18 de Dezembro de 2021.

[Mic21a] Microsoft. TypeScript Documentation, 2021. [Online] `https://www.typescriptlang.org/docs/`. Último acesso a 26 de Novembro de 2021.

[Mic21b] Microsoft. Visual Studio Code, 2021. [Online] `https://code.visualstudio.com/`. Último acesso a 22 de Novembro de 2021.

[Nod20] Node.js. About Node.js®, 2020. [Online] `https://nodejs.org/`. Último acesso a 6 de Março de 2020.

[OCN21] Cornelia Mihaela Novac Gyöngyi Bujdosó Mihai Oproescu e Teofil Gal Ovidiu Constantin Novac, Damaris Emilia Madar. *Comparative study of some applications made in the Angular and Vue.js frameworks*, 2021. [Online] `https://ieeexplore.ieee.org/abstract/document/9484150`. Último acesso a 28 de Dezembro de 2021.

[Paw22] Paw. *Paw*, 2022. [Online] `https://paw.cloud/`. Último acesso a 1 de Maio de 2022.

[Pos22] Postman. *Postman*, 2022. [Online] `https://www.postman.com/`. Último acesso a 1 de Maio de 2022.

[soo21] soohian. Doing Scrum using Backlog, 2021. [Online] `https://community.nulab.com/t/doing-scrum-using-backlog/371`. Último acesso a 12 de Novembro de 2021.

[TDT21] Gianina Adela Gabor e Elisa Valentina Moisi Teodor-Dorin Tripon. *Angular and Svelte Frameworks: a Comparative Analysis*, 2021. [Online] `https://ieeexplore.ieee.org/abstract/document/9484119`. Último acesso a 29 de Dezembro de 2021.

[Ube21] Uber. Uber, 2021. [Online] `https://www.uber.com/`. Último acesso a 11 de Dezembro de 2021.

[UKM21] Anirudh Raja e Madan Thangavelu Uday Kiran Medisetty, Nilesh Mahajan. Uber's Real-Time Push Platform, 2021. [Online] `https://eng.uber.com/real-time-push-platform/`. Último acesso a 12 de Dezembro de 2021.

[Yar22] Yarn. *Yarn*, 2022. [Online] `https://yarnpkg.com/`. Último acesso a 4 de Maio de 2022.