**Universidade de Aveiro**
2023

**Samuel Felisberto Matias Pinto dos Santos**

# Tecnologias e metodologias de desenvolvimento Web

## O caso da plataforma Campus by Fundação Altice

**Universidade de Aveiro**
2023

**Samuel Felisberto Matias Pinto dos Santos**

# Tecnologias e metodologias de desenvolvimento Web

## O caso da plataforma Campus by Fundação Altice

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Comunicação e Tecnologias Web, realizada sob a orientação científica do Doutor Luís Francisco Mendes Gabriel Pedro e do Doutor Carlos Manuel das Neves Santos do Departamento de Comunicação e Arte da Universidade de Aveiro

Dedico este trabalho ao meu pai e à minha mãe

o júri

presidente

Professora Ana Rita Costa Bonifácio Selores dos Santos
professora associada da Universidade de Aveiro

vogais

Professor Doutor Ademar Manuel Teixeira de Aguiar
professor associado da Universidade do Porto

Professor Doutor Luís Francisco Mendes Gabriel Pedro
professor associado da Universidade de Aveiro

**agradecimentos**

Em primeiro lugar, quero expressar o meu sincero agradecimento aos meus orientadores, Luís Pedro e Carlos Santos, pela assistência, disponibilidade e paciência que demonstraram durante todo o desenvolvimento deste projeto.

Gostaria de expressar um agradecimento especial à equipa da AlticeLabs@UA, cujo apoio foi essencial durante toda a investigação. Miguel Filipe, Ana Carvalho, Laura Ferreira, Lara Riboira, Nuno Ribeiro, João Almeida, Gabriel Ribeira, Mariana Marques, Marta Fradique, Daniel Figueiredo, Catarina Dionísio, João Brandão, João Freitas e Henrique Silva, o meu muito obrigado. Adicionalmente, quero estender o agradecimento à Adriana Machado, cuja ajuda, paciência e palavras encorajadoras tiveram um impacto extremamente positivo e influente ao longo de todo o processo.

Ao Miguel, amigo de longa data, agradeço pela sua constante presença, apoio, incentivo, acompanhamento e disponibilidade ao longo de todo este percurso académico, que agora termina.

Por fim, gostaria de manifestar o meu amor e gratidão à minha família: à minha mãe, ao meu pai e ao meu irmão. Com um agradecimento especial à minha mãe, pela educação e pelos ensinamentos valiosos que levarei comigo para toda a vida.

**palavras-chave**

Tecnologias Web, Metodologias Ágeis, Campus by Fundação Altice, Scrum, CI/CD

**resumo**

O desenvolvimento Web é uma área tecnológica em constante desenvolvimento, obrigando as equipas de investigação e desenvolvimento a uma atenção permanente relativamente a tecnologias, processos e metodologias de trabalho. No entanto, principalmente em contextos de equipas de pequena dimensão, muitas vezes não existe disponibilidade de recursos humanos para que tais preocupações sejam tornadas uma realidade com a recomendada frequência. Nessas situações, as equipas correm o risco de se tornarem menos eficientes relativamente às práticas mais atuais da indústria.

Neste contexto, refere-se o caso da equipa de investigação e desenvolvimento (I&D) do Campus na Universidade de Aveiro, atualmente composta por dez elementos. A investigação acompanha a preparação do processo de desenvolvimento de uma nova ferramenta tecnológica, intitulada "*Fora d'Aulas*", para a plataforma *Campus by Fundação Altice* já existente. Nesse processo, foca-se nas novas tecnologias disponíveis, na metodologia de trabalho da equipa e na implementação técnica dos processos de desenvolvimento.

Tendo por base a metodologia de investigação de desenvolvimento, foram traçados os objetivos de pesquisar, identificar e averiguar a aplicabilidade de novas tecnologias e processos que permitam melhorar a infraestrutura que suporta o Campus, bem como desenvolver essa solução, em conjunto com a equipa de I&D. Pretendeu-se ainda elaborar uma proposta de um conjunto de práticas, metodologias, ferramentas e técnicas de desenvolvimento para melhorar as vigentes na equipa.

As soluções encontradas resultaram de uma recolha de dados na forma de um *Focus Group* aos membros da equipa e também na forma de entrevistas a especialistas. Essas soluções foram sofrendo mudanças ao longo da investigação, de acordo com um processo iterativo de avaliação cíclico que correspondeu aos vários *Sprints da metodologia aplicada*. No final, foram avaliadas com recurso a um questionário aos membros à data integrantes da equipa, incluindo novos elementos que se juntaram durante o processo, dando estes também a sua avaliação enquanto recém-chegados. Esta recolha de dados revelou que a produtividade da equipa aumentou, tendo esta ficado satisfeita com a metodologia aplicada. Contudo, existe ainda bastante margem para melhorias no futuro, nomeadamente ao nível dos processos e tecnologias de desenvolvimento.

**keywords**

Web Technologies, Agile Methodologies, Campus by Fundação Altice, Scrum, CI/CD

**abstract**

Web development is a technological area in constant development, forcing research and development teams to pay constant attention to technologies, processes and working methodologies. However, especially in small team contexts, human resources are often not available to make these concerns a reality as often as they should be. In these situations, teams run the risk of becoming less efficient in relation to the most up-to-date industry practices.

In this context, we will be focused on the case of the research and development (R&D) team at the University of Aveiro Campus, currently made up of ten members. The research follows the preparation of the development process for a new technological tool, entitled "Fora d'Aulas", for the existing Campus platform. In this process, it focuses on the new technologies available, the team's working methodology and the technical implementation of the development processes.

Based on the development research methodology, the objectives were to research, identify and verify the applicability of new technologies and processes that would improve the infrastructure that supports the Campus platform, and to develop this solution together with the R&D team. The aim was also to draw up a proposal for a set of development practices, methodologies, tools, and techniques to improve those already in the team.

The solutions found were the result of data collection in the form of a Focus Group of team members and in the form of interviews with specialists. These solutions underwent changes over the course of the research, according to an iterative cyclical evaluation process that corresponded to the various *Sprints* of the methodology applied. In the end, they were evaluated using a questionnaire to the team members at the time, including new members who joined during the process, who also gave their assessment as newcomers. This data collection revealed that the team's productivity had increased, and the team was satisfied with the methodology applied. However, there is still plenty of room for improvement in the future, particularly in terms of development processes and technologies.

# Índice

**1. INTRODUÇÃO ..... 1**

1.1. PROBLEMA DE INVESTIGAÇÃO ..... 2

1.2. FINALIDADES E OBJETIVOS ..... 3

**2. ENQUADRAMENTO TEÓRICO ..... 5**

2.1. TECNOLOGIA *CAMPUS MULTI-TENANT* ..... 5

*2.1.1. A plataforma Campus by Fundação Altice ..... 6*

*2.1.2. Fora d'Aulas, o futuro do Campus by Fundação Altice ..... 7*

2.2. METODOLOGIAS E DEVOPS ..... 8

*2.2.1. Modelo cascata ..... 8*

*2.2.2. Scrum ..... 9*

*2.2.3. DevOps ..... 14*

*2.2.4. CI/CD (Continuous Integration / Continuous Delivery) ..... 15*

**3. METODOLOGIA ..... 19**

3.1. DESENHO DA INVESTIGAÇÃO ..... 19

3.2. FASES DA INVESTIGAÇÃO ..... 20

3.3. PARTICIPANTES ..... 21

3.4. INSTRUMENTOS DE RECOLHA DE DADOS ..... 22

*3.4.1. Focus Group para discussão dos métodos de trabalho e organização atuais e futuros 22*

*3.4.2. Entrevistas com especialistas ..... 23*

*3.4.3. Questionário - Avaliação do impacto das mudanças metodológicas na equipa ..... 25*

**4. RECOLHA DE DADOS ..... 27**

4.1. *FOCUS GROUP* PARA A AVALIAÇÃO DOS MÉTODOS DE TRABALHO E ORGANIZAÇÃO ATUAIS E FUTUROS ..... 27

*4.2.* ENTREVISTAS COM ESPECIALISTAS, EX-MEMBROS DA EQUIPA *CAMPUS BY FUNDAÇÃO ALTICE* ..... 32

*4.2.1. Metodologias ágeis e gestão de equipa ..... 33*

*4.2.2. Ferramentas de Desenvolvimento e Infraestrutura ..... 33*

**5. IMPLEMENTAÇÃO E DESENVOLVIMENTO ..... 36**

5.1. METODOLOGIAS DE DESENVOLVIMENTO ..... 36

*5.1.1. A otimização iterativa dos Sprints ..... 36*

*5.1.2. Teoria dos vidros partidos ..... 42*

*5.1.3. Resultados intercalares ..... 45*

5.2. DESENVOLVIMENTO PRÁTICO ..... 45

*5.2.1. Estudo de viabilidade técnica com parceiros ..... 45*

*5.2.2. Plano de implementação* ..... *47*

**6. AVALIAÇÃO DAS SOLUÇÕES IMPLEMENTADAS** ..... **65**

6.1. RESULTADOS DO QUESTIONÁRIO ..... 65

*6.1.1. Perfil dos inquiridos* ..... *65*

*6.1.2. Análise do questionário aos membros seniores* ..... *67*

*6.1.3. Análise do questionário aos novos membros* ..... *74*

**7. CONCLUSÕES E TRABALHO FUTURO** ..... **79**

**BIBLIOGRAFIA** ..... **81**

**APÊNDICES** ..... **83**

APÊNDICE 1 - CONSENTIMENTO INFORMADO FOCUS GROUP ..... 83

APÊNDICE 2 - ESTRUTURA DO FOCUS GROUP ..... 84

APÊNDICE 3 - GUIÃO DAS ENTREVISTAS ..... 85

APÊNDICE 4 - CONSENTIMENTO INFORMADO DAS ENTREVISTAS ..... 87

APÊNDICE 5 - QUESTIONÁRIOS PARA A EQUIPA DO CAMPUS BY FUNDAÇÃO ALTICE ..... 89

# Índice de Tabelas

TABELA 1 - PROCEDIMENTOS, INSTRUMENTOS E TIPOS DE RECOLHA DE DADOS..... 22

# Índice de Figuras

Figura 1 - Página do Campus by Fundação Altice ..... 6
Figura 2 - Modelo Tradicional de Cascata; Fonte: Scrum Reference Card (James, 2010) ..... 9
Figura 3 - Modelo Scrum; Fonte: Scrum Reference Card (James, 2010) ..... 10
Figura 4 - As fases da metodologia de desenvolvimento ..... 20
Figura 5 - Visão da plataforma Trello ..... 38
Figura 6 - Organização da documentação no Notion ..... 40
Figura 7 – Exemplo de um GitLab Runner instalado no servidor cmp2-stg-fe01 ..... 55
Figura 8 - Imagem, variáveis, serviços e fases do ficheiro gitlab-ci.yml ..... 57
Figura 9 – Cache e execução da pipeline do ficheiro gitlab-ci.yml ..... 57
Figura 10 - Triggers manuais para cada tenant no ficheiro gitlab-ci.yml ..... 58
Figura 11 - Resultado do script na interface do GitLab ..... 59
Figura 12 - Configurações iniciais do Dockerfile ..... 60
Figura 13 - Base da pasta de trabalho, cache e configuração do Yarn no Dockerfile ..... 61
Figura 14 - Instalação de pacotes no Dockerfile ..... 61
Figura 15 - Processo de build da plataforma no Dockerfile ..... 63
Figura 16 - Previsão de uma configuração para o servidor de NGINX com suporte a ambientes dinâmicos ..... 64

# Índice de Gráficos

Gráfico 1 - Perfil dos inquiridos no questionário ..... 66
Gráfico 2 - Perceção dos membros quanto à variação da produtividade ..... 67
Gráfico 3 - Avaliação dos inquiridos à execução dos Sprints ..... 69
Gráfico 4 - Avaliação da rigidez, rigor e controlo das soluções implementadas ..... 71
Gráfico 5 - Satisfação com a nova metodologia ..... 73
Gráfico 6 - Experiência prévia em metodologias ágeis dos novos membros da equipa ..... 75
Gráfico 7 - Perceção dos novos membros sobre a metodologia atual da equipa ..... 76
Gráfico 8 - Experiência dos novos membros na transição para a metodologia da equipa ..... 77

# 1. Introdução

Com o rápido avanço da tecnologia, é cada vez mais importante garantir que a infraestrutura tecnológica de um *website* esteja atualizada, não só para garantir uma boa experiência para o utilizador final, mas também para melhorar a eficiência, a segurança e a produtividade da equipa de desenvolvimento.

O desenvolvimento de uma nova ferramenta tecnológica para uma plataforma *Web* existente é um desafio complexo, visto que é necessário ter em conta a infraestrutura e arquitetura atuais, enquanto se aproveitam as vantagens de possíveis novas tecnologias e abordagens de desenvolvimento. Isso pode incluir a utilização de novas *frameworks*, bibliotecas e ferramentas de desenvolvimento, bem como a implementação de novas técnicas de programação e design.

Muitas equipas da área do desenvolvimento *Web* enfrentam desafios adicionais devido à rotatividade constante de membros, o que pode dificultar a continuidade do projeto, a curva de aprendizagem das tecnologias e metodologias de desenvolvimento e a incorporação de novos indivíduos. Por isso, é fundamental estabelecer uma metodologia de trabalho estruturada e clara para garantir que esses objetivos não sejam postos em causa.

Tendo isto em conta, a presente dissertação irá abordar o caso da equipa do *Campus by Fundação Altice* no seu processo de desenvolvimento e a preparação de uma nova ferramenta tecnológica, intitulada “Fora d’Aulas”, para a plataforma online já existente, focando-se nas novas tecnologias disponíveis, na metodologia de trabalho e na implementação técnica. Esta equipa é caracterizada pela sua pequena dimensão, cujo número de membros variou entre 6 e 10 elementos no decorrer desta investigação.

## 1.1. Problema de investigação

O *Campus by Fundação Altice*, anteriormente conhecido como *SAPO Campus*, é uma plataforma para a comunicação digital, no contexto educativo, resultado de uma parceria entre a Altice e a Universidade de Aveiro (Araújo et al., 2017).

Atualmente, devido a vários fatores, um deles e talvez o mais relevante sendo a recente pandemia e a crescente concorrência de plataformas a atuar onde o *Campus* outrora estrategicamente se posicionava, foi decidido avançar com um novo projeto, complementar ao *Campus*, chamado *Fora d'Aulas*. Este projeto é uma ferramenta tecnológica que tem como objetivo tornar-se num centro de convergência para o trabalho desenvolvido pelas escolas do ensino básico e secundário, especificamente no que diz respeito a projetos extracurriculares. Assim, a ferramenta pretende dar visibilidade a esses projetos e a eventuais parcerias com entidades externas, para além de promover a dinamização dessas iniciativas e incentivar a participação da comunidade escolar.

Como foi dito anteriormente, é um grande desafio desenvolver uma nova ferramenta tecnológica para uma plataforma *Web* existente há vários anos, visto que não é viável simplesmente começar do zero e construir um novo projeto uma vez que tal abordagem exigiria demasiado tempo e recursos. Em vez disso, o foco deve ser o de encontrar maneiras de otimizar e aprimorar a infraestrutura existente para aproveitar novas tecnologias e melhorar o seu desempenho, segurança e experiência, não só do utilizador final, mas também da equipa de investigação e desenvolvimento da infraestrutura.

Uma vez que a equipa encarregue do desenvolvimento da plataforma é constituída por estudantes que usufruem de uma bolsa de investigação em contexto académico, há alterações anuais ou semestrais nos membros que a compõem. Como tal, torna-se por vezes difícil garantir uma metodologia de trabalho consistente e otimizada às necessidades de desenvolvimento constantes para projetos como o *Fora d'Aulas*.

Sendo assim, a presente investigação pretende contribuir para a implementação futura da nova ferramenta tecnológica na plataforma *Campus by Fundação Altice*, na forma do projeto *Fora d'Aulas*, estudando e implementando soluções tecnológicas que contribuam para o aprimoramento das metodologias e ferramentas de trabalho atuais, no contexto da equipa de I&D, para que as futuras implementações de novos projetos já beneficiem dessas melhorias.

## 1.2. Finalidades e objetivos

Como foi mencionado anteriormente, o presente estudo, pela sua natureza projetual, tem como principal finalidade auxiliar na definição e melhoria das metodologias, ferramentas e tecnologias adequadas para contribuir futuramente para o desenho e desenvolvimento de novas ferramentas tecnológicas, capazes de serem integradas na plataforma *Campus by Fundação Altice*.

Para este tipo de estudo não existe, necessariamente, a obrigatoriedade de formular uma pergunta de investigação para a qual se tentem encontrar respostas. Em vez disso, foram estabelecidos alguns objetivos que servirão como linhas de orientação durante o desenvolvimento do estudo:

- Pesquisar, identificar e averiguar a aplicabilidade de novas tecnologias que permitam melhorar a infraestrutura que suporta o *Campus*;
- Desenvolver essa solução, em conjunto com a equipa de *I&D*;
- Propor um conjunto de práticas, metodologias, ferramentas e técnicas de desenvolvimento para melhorar as vigentes na equipa;

# 2. Enquadramento teórico

## 2.1. Tecnologia *Campus multi-tenant*

A plataforma *Web Campus by Fundação Altice* é parte integrante de uma estrutura tecnológica *multi-tenant*. Segundo Pathirage et al. (2012), uma estrutura *multi-tenant* é uma arquitetura de software que permite que múltiplos clientes (conhecidos como "*tenants*") compartilhem uma instância única de um sistema de software, enquanto mantêm as suas próprias configurações, dados e personalizações. Isto é diferente de uma arquitetura de software tradicional, onde cada cliente possui a sua própria instância do sistema.

"A arquitetura *multi-tenan*t é ainda caracterizada por uma estratégia de partilha de recursos – "*resource polling*" e serviços numa lógica de *Software as a Service – SaaS* entre múltiplas máquinas clientes. No caso específico da infraestrutura que suporta a plataforma *Campus*[1], e que é a base para três outras plataformas - *miOne*[2]*, Global Portuguese Scientists (GPS)*[3] e *ColLab*[4] - possui apenas uma base de dados e respetivo esquema (*schema)*, partilhada por todas as plataformas" (Silva, 2021, p. 27). Atualmente, em parceria com a Fundação Altice, encontra-se em desenvolvimento um novo *tenant* com a tecnologia *Campus* com o nome de "Espaços Inclui".

A principal vantagem deste tipo de tecnologia é a capacidade de aplicar funcionalidades comuns com diferentes níveis de personalização, adaptadas às necessidades específicas de cada plataforma.

Por exemplo, a plataforma *Campus* possui uma estrutura semelhante às outras plataformas, mas pode diferenciar-se em aspetos como as suas políticas de privacidade, design, estrutura e hierarquia de contextos. Mesmo com estas diferenças, existe um número considerável de funcionalidades partilhadas entre as

[1] Campus, https://campus.altice.pt (Data de acesso: 12 de janeiro de 2023)
[2] miOne, https://mione.altice.pt (Data de acesso: 12 de janeiro de 2023)
[3] GPS, https://gps.pt (Data de acesso: 12 de janeiro de 2023)
[4] ColLab, https://collab-edu.com (Data de acesso: 12 de janeiro de 2023)

diferentes plataformas que incluem, por exemplo, a capacidade de criar comunidades, grupos, páginas, partilhar conteúdo, comentar e salas de chat (Silva, 2021).

Em resumo, a arquitetura *multi-tenant* é uma solução escalável e eficiente que permite a personalização dos recursos e funcionalidades de acordo com as necessidades específicas de cada *tenant*, sem comprometer a eficiência operacional ou de gestão.

### 2.1.1. A plataforma *Campus by Fundação Altice*

O *Campus by Fundação Altice* é uma plataforma digital para comunicação no contexto educativo, resultante de uma parceria entre a Fundação Altice e a Universidade de Aveiro (Araújo et al., 2017).

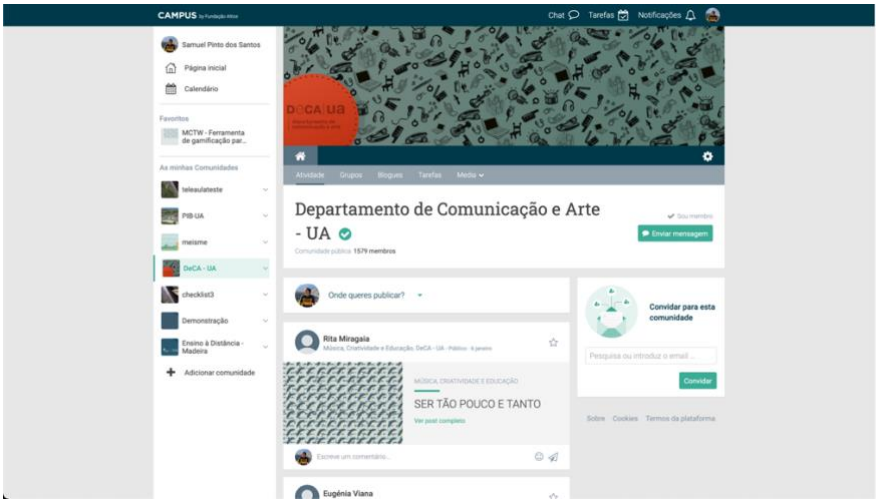



*Figura 1 - Página do Campus by Fundação Altice*

A plataforma, anteriormente conhecida como *SAPO Campus*, foi lançada pela primeira vez em setembro de 2009 como uma plataforma exclusiva para a comunidade universitária da Universidade de Aveiro (Santos, 2013). Desde então, a plataforma tem evoluído, adaptando-se às necessidades e às novas tecnologias da *Web*, tendo sido lançadas novas versões com funcionalidades adicionais e permitindo o seu uso em diversos contextos além do ensino superior, tais como escolas básicas ou secundárias.

Em setembro de 2019, a plataforma foi relançada como *Campus by Fundação Altice* (Figura 1), continuando a evoluir e a adaptar-se às necessidades dos seus utilizadores e às novas tecnologias disponíveis. Essa atualização, que está agora em vigor como a versão atual da plataforma, foi responsável por recriar a antiga infraestrutura e organizá-la na metodologia de reutilização entre *tenants* (Silva, 2021).

### 2.1.2. Fora d'Aulas, o futuro do Campus by Fundação Altice

Atualmente, a equipa de I&D responsável pelo projeto planeia e executa já a próxima atualização da plataforma. Neste plano, insere-se o trabalho desta e o de outras seis dissertações, correspondentes a outros tantos membros da equipa. O resultado esperado destes trabalhos de investigação consubstancia, entre outras coisas, uma modernização gráfica e técnica da plataforma *Web* e a adição de uma nova ferramenta – o *Fora d'Aulas*.

O ***Fora d'Aulas*** é um projeto de desenvolvimento de uma ferramenta digital que será uma funcionalidade integrante da plataforma *Campus by Fundação Altice*. O objetivo é criar um polo agregador do trabalho desenvolvido por escolas do ensino básico e secundário, ao nível da dinamização de projetos extracurriculares, promovendo assim o sentimento de pertença à comunidade escolar.

Esta ferramenta terá o objetivo de facilitar a dinamização de iniciativas e reconhecer a participação da comunidade escolar nas mesmas. Para isso, terá recursos integrados que se destacam pela sua capacidade de melhorar a experiência de utilização e enriquecer a experiência escolar do utilizador. Em particular, será dada atenção especial às estratégias de gamificação auxiliadas por uma componente de *storytelling*, atuando como um guia para ajudar os utilizadores a interagir com a plataforma e se envolverem nas situações de aprendizagem informal alavancadas com estas iniciativas, e na atribuição de *badges* digitais, numa lógica de reconhecimento da participação e da qualidade dela.

## 2.2. Metodologias e DevOps

As metodologias ágeis surgiram como uma resposta às limitações e ineficiências das metodologias tradicionais de desenvolvimento de *software*. O manifesto ágil foi publicado em 2001 por um grupo de programadores de *software* que se reuniram para discutir as melhores práticas e formas de melhorar a eficiência e qualidade do desenvolvimento de *software*. Esse manifesto apresenta quatro valores fundamentais: indivíduos e interações, software funcional, colaboração com o cliente e resposta a mudanças. Esses valores servem como guia para a implementação de metodologias ágeis no desenvolvimento de *software*, visando a melhoria contínua e a adaptabilidade às necessidades e mudanças do projeto (Martin, 2009).

Uma das principais características das metodologias ágeis é a valorização do trabalho colaborativo, com a participação ativa do cliente no processo de desenvolvimento. Outra característica, também importante, é a capacidade de adaptação a mudanças, permitindo que os projetos sejam ajustados de acordo com as necessidades do cliente e do mercado.

### 2.2.1. Modelo cascata

O modelo cascata é antecedente às metodologias ágeis. Segundo este modelo, todos os detalhes e requisitos de um projeto devem ser especificados e documentados no início, sem possibilidade de mudanças durante o desenvolvimento. Desta forma, o cliente tem de saber desde o princípio o que quer, antes de ver o produto final. Este modelo é caracterizado por ser bastante rígido, com longos e exaustivos processos de planeamento e especificação de requisitos no início do projeto (Figura 2). Essa especificação não contempla possíveis mudanças no decurso do projeto, pelo que o cliente tem de identificar as suas necessidades desde logo. Para que isto seja possível, todo o processo de desenvolvimento de software tem de ser previsível e repetível (Martin, 2009).

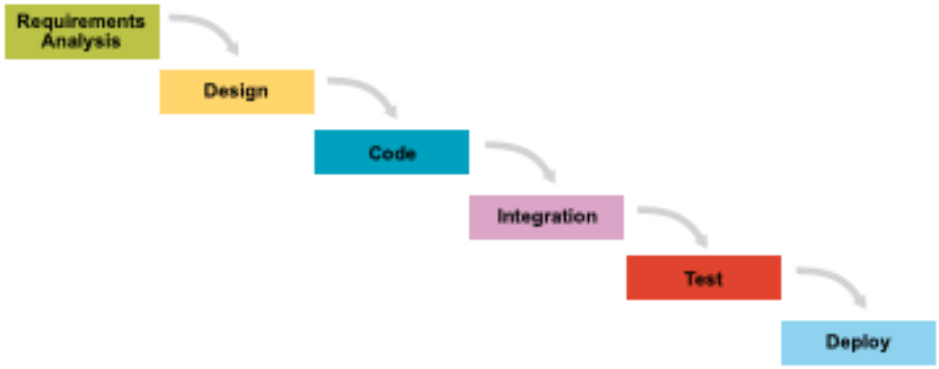



*Figura 2 - Modelo Tradicional de Cascata; Fonte: Scrum Reference Card* (James, 2010)

O excessivo rigor e a inflexibilidade com que esses projetos são conduzidos representam uma restrição significativa e frequentemente resulta em atrasos na entrega e desvios nos custos previstos. As metodologias ágeis, como o *Scrum* e o *Kanban*, surgiram como uma potencial resposta às limitações do modelo cascata. Estas metodologias enfatizam a flexibilidade, a colaboração entre equipas, a entrega iterativa e a capacidade das equipas se adaptarem a mudanças de forma mais rápida (Martin, 2009).

### *2.2.2. Scrum*

A metodologia *Scrum* adota uma abordagem iterativa e visa aperfeiçoar a gestão de riscos e a previsibilidade. Em contraste com o modelo tradicional de cascata, é possível prever o esboço inicial do produto final, tendo sempre em conta que ele pode sofrer alterações devido a fatores externos. A mudança não é rejeitada, mas vista como uma inevitabilidade e como um fator que contribui para o sucesso dos projetos. O *Scrum* baseia-se nos três princípios da transparência, inspeção e adaptação.

**Transparência:** Todos os aspetos do projeto devem estar disponíveis e compreensíveis para todos os membros da equipa. Desta forma, este princípio permite que a equipa identifique rapidamente os problemas e tome decisões informadas.

**Inspeção:** Isto significa que a equipa deve regularmente inspecionar o projeto para garantir que está no caminho certo. Isto pode ser feito através de reuniões diárias, reuniões de *Sprint* e revisões de *Sprint*. A inspeção permite assim que a equipa identifique problemas e tome medidas para os corrigir antes que estes se tornem críticos.

**Adaptação:** A equipa deve estar preparada para mudar o plano do projeto se as necessidades do cliente mudarem. Isso pode ser feito através de reuniões de *Sprint*, revisões de *Sprint* e reuniões de retrospetiva. A adaptação permite que a equipa se adapte rapidamente às mudanças e mantenha o projeto no caminho certo.

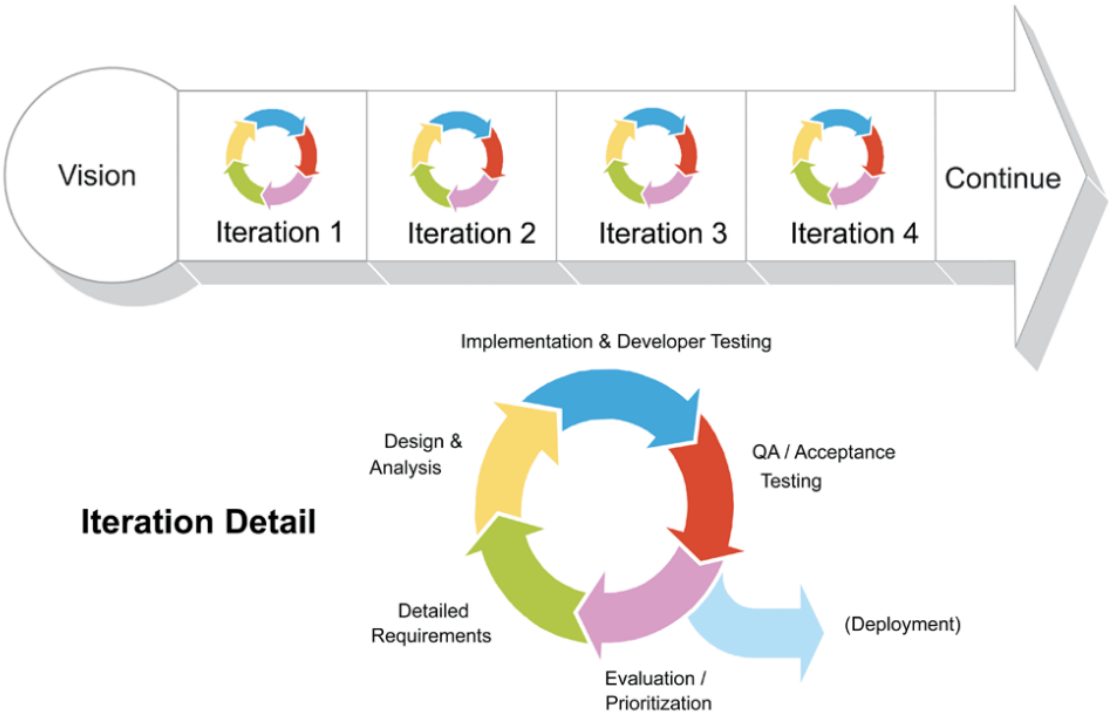



*Figura 3 - Modelo Scrum; Fonte: Scrum Reference Card* (James, 2010)

Em Scrum, existem rituais, artefactos e figuras-chave (Fokina, 2013; James, 2010; Martin, 2009; Schwaber & Sutherland, 2011), que serão abordados a seguir.

#### 2.2.2.1. Rituais do Scrum

Os rituais do Scrum são práticas fundamentais que estruturam o fluxo de trabalho em ambientes ágeis. Estes rituais visam promover a colaboração constante, a adaptabilidade e a transparência dentro das equipas, ajudando a manter o foco nos objetivos do projeto e a facilitar a comunicação eficiente e o feedback contínuo.

- ***Sprint***

O *Sprint* é o ciclo de desenvolvimento no processo de implementação do projeto. Deve ter uma duração mínima de uma semana e máxima de um mês. Esta duração deve ser consistente durante todo o processo de desenvolvimento. Assim que se termina um *Sprint*, começa outro. São, assim, caixas temporais com tarefas a serem executadas desde o início até ao final do projeto.

- ***Daily Scrum* (ou *Daily Stand-up*)**

Chama-se *Daily* à reunião que é realizada diariamente e cujo objetivo principal consiste não só em efetuar uma análise crítica da evolução desde a última reunião, mas também em definir quais são os desenvolvimentos que devem estar feitos na próxima. As reuniões deverão ter um caráter objetivo, realizar-se sempre à mesma hora e no mesmo local, não ultrapassando a duração máxima de 15 minutos.

- ***Sprint Review***

Trata-se de uma reunião no final do *Sprint* onde a equipa apresenta o trabalho concluído. É uma oportunidade para o Product Owner, a equipa de *Scrum* e quaisquer *stakeholders* reverem o progresso e darem *feedback*.

- ***Sprint Retrospective***

Após a *Sprint Review*, a equipa realiza uma retrospetiva para discutir o que funcionou bem, o que pode ser melhorado, e que ações podem ser tomadas para melhorar no próximo *Sprint*. É um momento chave para a aprendizagem e melhoria contínua da equipa.

#### 2.2.2.2. Artefactos do *Scrum*

Os artefactos do Scrum são ferramentas que fornecem informações essenciais sobre o progresso do projeto e o trabalho em curso. Estes artefactos ajudam a equipa a visualizar o trabalho, priorizar tarefas e gerir expectativas, garantindo que todos os membros estejam alinhados com os objetivos do projeto e com as entregas necessárias.

- ***Product Backlog***

O *Product Backlog* é uma lista que contém todos os componentes, requisitos funcionais do produto, erros encontrados, alteração das funcionalidades implementadas, entre outros. É um documento dinâmico e flexível, em que estão especificadas as tarefas a serem realizadas para o desenvolvimento do projeto. É monitorizado e organizado pelo *Product Owner*.

- ***Sprint Backlog***

É um conjunto de tarefas retiradas do *Product Backlog* que a equipa de desenvolvimento se comprometeu a realizar durante o decorrer do *Sprint*. Divide-se normalmente em três sessões: tarefas por fazer, tarefas em execução e tarefas concluídas.

- **Incremento (ou potencialmente Produto "*Shippable*")**

O resultado do *Sprint*, que deve ser uma versão utilizável e potencialmente entregável do produto. Isto não significa que o produto é necessariamente lançado no final de cada *Sprint*, mas deve estar numa forma que poderia ser, se assim fosse decidido.

#### 2.2.2.3. Figuras-Chave do Scrum

As figuras chave do Scrum desempenham um papel vital no sucesso de projetos que seguem esta metodologia ágil. Estas figuras, compostas por distintos papéis com responsabilidades específicas, garantem que os princípios e práticas do Scrum sejam implementados eficazmente. A sua existência e importância residem na necessidade de uma liderança clara, uma visão de produto coerente e uma equipa eficiente e auto-organizada. Juntas, estas figuras trabalham em uníssono para promover um ambiente onde a comunicação é fluida, os objetivos são claros e o progresso é contínuo e alinhado com as necessidades do cliente.

- ***Scrum Master***

É o responsável pela implementação e execução da metodologia *Scrum* e garante que a metodologia é compreendida pela equipa. As funções principais do *Scrum Master* compreendem não só a realização e a gestão de todos os eventos do *Scrum* (*Daily Stand-up*, *Sprint Review*, *Sprint Retrospective*), mas também a manutenção do *Product Backlog* e *Sprint Backlog* em constante e estreita comunicação com o *Product Owner*. Além disso, monitoriza o trabalho da equipa, elabora diagnósticos concisos sobre o desempenho das equipas e resolve impedimentos de caráter não técnico;

- ***Product Owner***

O *Product Owner* é responsável por maximizar o valor do produto e o trabalho da equipa de desenvolvimento. Ele é o responsável por tomar as decisões mais importantes em relação ao rumo do projeto, a fim de potenciar o aumento do retorno do investimento. Ele assume a gestão, configuração da equipa, garante o aumento do valor do trabalho realizado, a visibilidade e a transparência do *Product Backlog*;

- **Equipa de Desenvolvimento**

Uma equipa de desenvolvimento deve ser auto-organizada, versátil, e não controlada por pessoas externas. Cabe aos membros da equipa tomar decisões sobre a melhor forma de cumprir o seu trabalho, pois cada membro possui as

competências necessárias para a sua realização, tais como a flexibilidade, criatividade e produtividade.

#### 2.2.2.4. *Epic* e *Milestone*

No Scrum e na gestão ágil de projetos, os conceitos de *epic* e *milestone* são fundamentais, mas cada um desempenha um papel distinto. Um *epic*, no contexto do Scrum, é uma grande unidade de trabalho que engloba várias tarefas. Representa uma categoria ampla para um conjunto de funcionalidades relacionadas, que são demasiado extensas para serem tratadas como uma única tarefa. O uso dos *epics* ajuda a organizar o *backlog*, permitindo dividir o trabalho em secções de mais fácil gestão . Estas secções podem, posteriormente, ser decompostas em tarefas mais detalhadas, facilitando o planeamento e acompanhamento a longo prazo do projeto.

Por outro lado, um *milestone*, é um ponto específico no tempo dentro do projeto, marcando um evento ou um momento significativo. Este pode ser o fim de uma fase importante do projeto, a conclusão de um conjunto chave de funcionalidades, ou qualquer outro ponto que denote um progresso importante. Os *milestones* são utilizados para monitorizar e celebrar estes avanços, fornecendo à equipa pontos de referência claros e ajudando a manter todos alinhados e focados nos objetivos do projeto.

Portanto, enquanto um *epic* se concentra mais no conteúdo e no objetivo do trabalho a ser realizado, um milestone está mais relacionado com a marcação de momentos significativos no tempo durante a execução do projeto. Ambos são essenciais na estruturação e no acompanhamento do progresso em metodologias ágeis de gestão de projetos. (Fokina, 2013; James, 2010; Martin, 2009; Schwaber & Sutherland, 2011)

### 2.2.3. DevOps

*DevOps* é uma abordagem que visa aumentar a eficiência e a eficácia da colaboração entre os departamentos de desenvolvimento de *software* (*Dev*) e operações (*Ops*) numa organização. Esta abordagem permite que as organizações

desenvolvam, testem e implementem software de forma mais rápida e segura, melhorando a capacidade de responder às necessidades do negócio (Kim et al., 2018).

Esta abordagem é baseada em três pilares principais: cultura, ferramentas e metodologias. A cultura *DevOps* é centrada na colaboração, na comunicação e na automação, com o objetivo de melhorar a eficiência e a qualidade do software. As ferramentas *DevOps* permitem automatizar o processo de desenvolvimento, desde o código-fonte até a implementação em produção, a fim de agilizar o processo e diminuir o risco de erros (Forsgren et al., 2018).

Ao aplicar este tipo de abordagem, as equipas podem identificar e resolver problemas de forma mais célere, o que permite lançar novas funcionalidades e correções com mais frequência. Isso é especialmente importante num mundo cada vez mais digital, onde as expectativas dos utilizadores são cada vez mais altas. Consequentemente, a qualidade do *software* aumenta, dado que graças à automatização do processo de desenvolvimento e implementação, é possível diminuir o risco de erros e melhorar a robustez do código. Isso é especialmente importante em aplicações que comunicam com uma quantidade de público elevada ou sistemas que, pela sua natureza, necessitam de uma camada extra de segurança, como sistemas de saúde ou sistemas financeiros, onde um erro pode ter consequências graves.

### 2.2.4. CI/CD (Continuous Integration / Continuous Delivery)

*CI/CD*, ou *Continuous Integration* / *Continuous Delivery*, é uma metodologia que permite aos programadores implementar mudanças no código-fonte de um projeto de forma rápida e segura. A ideia é que, em vez de esperar por um ciclo de lançamento completo para fazer as modificações, elas possam ser feitas e testadas de forma contínua, garantindo assim maior qualidade e estabilidade no software (Humble & Farley, 2010).

Este é um conceito cada vez mais importante na indústria de desenvolvimento de *software*. Uma das ferramentas chave para implementar *CI/CD*

é o uso de *pipelines*. Estas *pipelines* são uma série de etapas automatizadas que são executadas quando o código é enviado para o repositório, podendo mesmo percorrer todo um processo (caso assim esteja definido pelos programadores) até ao momento em que é implementado num ambiente de produção (Humble & Farley, 2010). Cada etapa da *pipeline* é responsável por verificar se o código atende aos critérios de qualidade previamente definidos, e só permite que se avance para a próxima etapa se esses critérios forem satisfeitos. Um exemplo de *pipeline* de *CI/CD* poderia incluir etapas como:

- Verificação de sintaxe e erros de compilação;
- Execução de testes automáticos;
- Implementação em ambiente de testes;
- Implementação em ambiente de produção.

Os testes automáticos são uma técnica utilizada para automatizar a verificação de que um software está a funcionar corretamente. Estes são escritos utilizando uma linguagem de programação específica, com o objetivo de simular a interação do utilizador com o software e verificar se o resultado obtido é o esperado (Ebert et al., 2016).

Existem vários tipos de testes automáticos, como testes unitários, testes de integração e testes de aceitação. Pode-se encarar a principal diferença entre eles como um sistema hierárquico. Ou seja, os testes unitários são os mais básicos e verificam se cada pequena parte do código (ou unidade) está a operar consoante o que foi previamente estabelecido. Os testes de integração, por sua vez, verificam se as unidades do código, enquanto grupo, estão a funcionar corretamente. E os testes de aceitação, que são os mais complexos, verificam se o software atende às necessidades dos utilizadores como um todo, muitas vezes testando processos ou ações comuns (ex. Processo de registo de um utilizador numa plataforma).

Os testes automáticos têm várias vantagens, como a possibilidade de detetar erros de forma rápida e precisa, aumentar a confiança no software e facilitar a manutenção do código. Além disso, permitem que os programadores se possam

concentrar no desenvolvimento de novas funcionalidades, sem se preocupar tanto com possíveis erros (Ebert et al., 2016; Humble & Farley, 2010).

No entanto, existem também algumas desvantagens relacionadas com os testes automáticos. Um dos principais problemas é que estes podem ser difíceis de configurar e manter, especialmente se o software estiver em constante evolução. Além disso, os testes automáticos podem ser dispendiosos em termos de tempo e recursos, e podem não ser capazes de detetar todos os erros, especialmente aqueles relacionados com a interface ou a usabilidade da plataforma em questão (Ebert et al., 2016; Humble & Farley, 2010).

Um aspeto interessante relacionado com os processos de CI/CD é o conceito introduzido por Forsgren et al. (2018) relativo ao *burnout* dos programadores. A aplicação de uma estratégia de *CI/CD* pode ajudar a reduzir ou mitigar este fenómeno de várias maneiras. Em primeiro lugar, a automatização de tarefas repetitivas e demoradas, como testes e implementações em ambientes de produção, liberta tempo para que os programadores se possam concentrar em tarefas mais criativas e desafiantes. Isso pode ajudar a evitar a sensação de tédio e exaustão que muitas vezes está associado à realização deste tipo de tarefas.

Em segundo lugar, a implementação de um fluxo de trabalho baseado em *CI/CD* pode ajudar a garantir que os programadores estejam a trabalhar num ambiente que é o mais estável e previsível possível. Isso pode ajudar a reduzir a incerteza e o stress associados às mudanças de última hora e às implementações falhadas.

Além disso, como vimos, a implementação de testes automáticos e *pipelines* pode auxiliar os programadores a submeterem código de maior qualidade. Isso pode obviamente ajudar a evitar erros e, consequentemente, stress adicional.

Finalmente, a cultura de melhoria contínua e *feedback* que é cultivada por uma estratégia de *CI/CD* pode apoiar os programadores e fazer com que estes sintam que as suas contribuições estão a ser valorizadas e, ao mesmo tempo, a progredir na completude das tarefas e objetivos do projeto em que estão inseridos.

# 3. Metodologia

No presente capítulo será exposto o modelo metodológico adotado para a realização desta dissertação de mestrado. Abordar-se-á em detalhe o desenho da investigação, os instrumentos de recolha de dados e os participantes envolvidos na mesma.

## 3.1. Desenho da investigação

Tal como foi mencionado nos capítulos anteriores, a presente investigação consiste no aprimoramento das tecnologias, metodologias e ferramentas de trabalho no contexto da equipa da bolsa de investigação, para que a implementação atual (na forma da atualização gráfica da plataforma) e futura (na forma do *Fora d'Aulas*) seja otimizada e melhore a experiência de desenvolvimento dos membros da equipa.

Inicialmente, numa fase embrionária do trabalho, foi equacionado utilizar a investigação-ação como metodologia para este trabalho, uma vez que é uma forma de pesquisa que visa compreender e melhorar uma prática ou contexto específico através da participação ativa e colaboração das pessoas ou grupos envolvidos. É um processo cíclico que envolve planeamento, ação, observação e reflexão sobre os resultados da ação e ajustes de acordo com os resultados. O objetivo da investigação-ação é promover mudanças positivas e melhorar a eficácia da prática ou do contexto estudado e é caracterizado pela sua natureza participativa e colaborativa, uma vez que pode ser realizada por investigadores, profissionais ou pelos próprios indivíduos ou grupos do estudo, com o objetivo de compreender a sua própria prática e melhorá-la (Reason & Bradbury, 2008).

Apesar de uma parte deste trabalho se suportar bastante no contributo dos membros da equipa, nomeadamente no que diz respeito às metodologias de trabalho da equipa de I&D, tal contributo não pode ser garantido para o desenvolvimento técnico e para as escolhas tecnológicas que virão a ser feitas. Por essa razão, foi decidido não enveredar por esta metodologia.

Assim, verifica-se que a abordagem metodológica que mais se adequa a este trabalho é a **metodologia de investigação de desenvolvimento**. De acordo com Van den Akker (1999), a metodologia de investigação de desenvolvimento tem como objetivo proporcionar contribuições tanto práticas quanto teóricas, num processo cíclico, que passa por várias fases de análise, desenvolvimento e avaliação, voltando sempre ao momento de início do processo, até que o produto satisfaça as necessidades previamente estabelecidas (Figura 4). Esta metodologia difere da investigação-ação pois dá ao investigador mais liberdade de escolha nos momentos de tomada de decisão, sem ter de depender do contributo de outros, mantendo a lógica de elaboração de sucessivas iterações até que se atinja um equilíbrio satisfatório entre as ideias e o resultado.

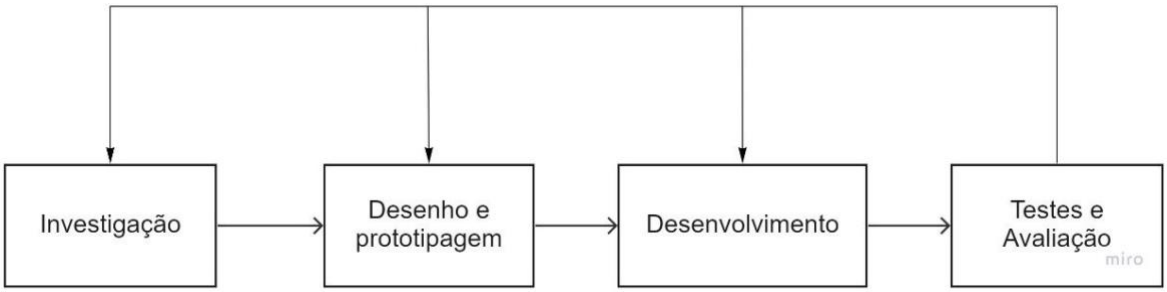



*Figura 4 - As fases da metodologia de desenvolvimento*

## 3.2. Fases da investigação

- **Investigação** – Foi feita a pesquisa bibliográfica, bem como consulta e aprendizagem de recursos sobre tecnologias em que não havia conhecimento prévio, para que houvesse um entendimento suficiente da dimensão teórica e prático desta investigação;

- **Desenho e prototipagem** – Nesta fase foi feita uma consulta e respetiva análise dos requisitos técnicos e metodológicos da equipa, bem como o desenho do plano de implementação;

- **Desenvolvimento** – Implementação das soluções tecnológicas na infraestrutura e soluções metodológicas na equipa;

- **Testes e avaliação** – Avaliação das soluções implementadas através do levantamento da experiência dos membros da equipa de investigação e desenvolvimento.

## 3.3. Participantes

Tendo em conta a tipologia e os objetivos deste trabalho podemos definir três grupos de participantes distintos:

- Membros da equipa de I&D do *Campus by Fundação Altice*;
- Ex-membros da equipa, especialistas a nível da infraestrutura tecnológica;
- Novos membros da equipa, integrados sensivelmente a meio da investigação.

Visto que os ex-membros da equipa já se encontram, na maior parte dos casos, com anos de experiência na indústria e foram responsáveis pela criação da infraestrutura tecnológica atual da plataforma *Campus* e respetivos *tenants*, foram considerados especialistas no contexto do presente trabalho.

Sendo assim, considera-se que a seleção da amostra para o momento de recolha de dados será feita através de um processo de amostragem por conveniência, que é caracterizado pela sua praticidade. No entanto, é importante salientar que esta amostragem pode levar a resultados parciais e não generalizáveis, pelo que é importante considerar essa limitação no momento de interpretação dos resultados.

## 3.4. Instrumentos de recolha de dados

Notando desde já que foi aplicada uma abordagem exploratória mista, realizou-se a recolha de dados recorrendo a vários procedimentos e devidos instrumentos, identificados na Tabela 1.

*Tabela 1 - Procedimentos, Instrumentos e Tipos de recolha de dados*

| Procedimento | Instrumento | Tipo |
|---|---|---|
| **(**3.4.1**) Avaliação dos métodos de trabalho e organização atuais e futuros** | *Focus Group* com a equipa atual do *Campus by Fundação Altice* | Qualitativo |
| **(**3.4.2**) Recolha de testemunhos e recomendações de especialistas** | Entrevistas com especialistas, ex-membros da equipa *Campus by Fundação Altice* | Qualitativo |
| **(**3.4.3**) Avaliação do impacto das mudanças metodológicas na equipa** | Questionário aos membros da equipa atual do *Campus by Fundação Altice* | Quantitativo e Qualitativo |

### 3.4.1. *Focus Group* para discussão dos métodos de trabalho e organização atuais e futuros

Num momento inicial da investigação foi realizado um *Focus Group* com a equipa do *Campus by Fundação Altice*. Este procedimento qualitativo permitiu obter uma visão geral dos problemas e desafios enfrentados pela equipa e, ainda, identificar as áreas de possível melhoria, bem como fomentar a discussão sobre metodologias de trabalho e organização passíveis de serem implementadas na equipa.

Para a estrutura deste *Focus Group* foram tidos em conta os conhecimentos adquiridos até à data na fase de análise da literatura, nomeadamente as regras e boas práticas que se devem cumprir num contexto de trabalho com metodologias

ágeis. Considerando a forma como a equipa trabalhava até então, e comparando certos pontos com essas boas práticas, foi desenvolvida uma estrutura base para o *Focus Group* (Apêndice 2 - Estrutura do Focus Group), com alguns temas ou questões obrigatórias a serem abordadas durante a discussão.

Numa fase posterior, para que fosse possível analisar os resultados obtidos neste método de recolha de dados, durante a discussão foram anotadas à mão os pontos de vista apresentados pelos participantes. Para além disso, foi pedido a esses participantes que preenchessem um consentimento informado de participação no Focus Group (Apêndice 1 - Consentimento Informado Focus Group), assegurando a conformidade com as normas do Regulamento Geral de Proteção de Dados (RGPD).

### 3.4.2. Entrevistas com especialistas

De seguida, foram realizadas entrevistas com especialistas - ex-membros da equipa *Campus by Fundação Altice*. Apesar de existir muita informação disponível sobre as metodologias de trabalho e boas práticas em equipas de tecnologias da informação, cada empresa e cada equipa acaba por ter a sua interpretação e implementação, obviamente adaptada à sua realidade. Uma vez que este trabalho se insere numa equipa com uma realidade muito específica (sob o contexto de bolsas de investigação, numa equipa de número de membros variável entre apenas seis e dez estudantes), foi decidido preparar um conjunto de entrevistas semiestruturadas com alguns especialistas, atualmente a trabalhar na indústria de desenvolvimento *Web*, de forma a enriquecer a informação recolhida na fase de leitura da literatura sobre o mesmo tema.

**Objetivo:** Como foi mencionado acima, o objetivo deste conjunto de entrevistas foi o de complementar a informação recolhida na análise da literatura e conhecer efetivamente quais são as metodologias de trabalho e de desenvolvimento na indústria, assim como perceber, de uma forma prática, como é que cada equipa se adapta a essas metodologias consoante as suas necessidades específicas. Assim, foi possível estabelecer não só um conjunto de práticas de

trabalho em equipa, mas também de ferramentas e soluções tecnológicas possíveis de serem implementadas e testadas no contexto de trabalho deste estudo.

**Justificação*:*** A participação nesta entrevista foi limitada a ex-membros da equipa *Campus by Fundação Altice,* que atualmente desempenham cargos relacionados com o desenvolvimento *Web* ou gestão de equipas em empresas nacionais e internacionais. Desta forma, foi possível recolher informações sobre a sua realidade de trabalho, mas também a sua opinião quanto à viabilidade dessa realidade no contexto da equipa em que este estudo se insere, baseando-se na sua própria experiência prévia.

Em alternativa a esta entrevista, na fase embrionária desta dissertação, tinha sido estipulado realizar um *Focus Group* com o mesmo objetivo. No entanto, existem dois motivos principais que inviabilizaram esta abordagem. O primeiro é a dificuldade de estabelecer uma data que permitisse a participação de todos os membros numa sessão síncrona, por incompatibilidade de horários de trabalho. O segundo, dada a discrepância de experiência, conhecimento e anos de trabalho entre os vários participantes, seria um risco iniciar uma discussão aberta entre todos, uma vez que a participação de uns poderia ofuscar ou obstruir as opiniões dos outros.

**Protocolo*:*** Sendo esta uma entrevista semiestruturada com o objetivo de recolher informação detalhada e as respetivas perspetivas dos participantes relativamente à temática, foi utilizado um conjunto de questões fixas, com espaço e flexibilidade para questões de seguimento caso essas fossem pertinentes. Mesmo sendo o universo de participantes escolhido tão específico, ainda existia alguma heterogeneidade devido às funções que esses participantes atualmente desempenham ou desempenharam na equipa do *Campus by Fundação Altice* (*Front-End Developers*; *Back-End Developers* e *DevOps*). Por isso, foi decidido dividir as questões, separando aquelas que dizem respeito a desenvolvimento *Web* das que se focam em gerir, manter e coordenar a equipa de desenvolvimento. As questões foram aplicadas consoante o perfil do entrevistado. O guião utilizado para a condução das entrevistas pode ser consultado no Apêndice 3 - Guião das Entrevistas.

**Métodos e tratamento de dados:** Com o objetivo de assegurar uma análise precisa dos resultados, foram utilizadas gravações de áudio das entrevistas. Este método foi escolhido devido à sua eficácia na captura da verbalização dos participantes, sem interferir nos comportamentos e expressões não verbais.

Para conduzir as entrevistas em conformidade com as regulamentações do RGPD, no início de cada sessão, foi apresentado ao entrevistado um formulário de consentimento informado (que pode ser consultado no Apêndice 4 - Consentimento Informado das Entrevistas), pedindo a sua autorização para a participação e respetiva gravação da entrevista.

Posteriormente, foi realizada uma análise cuidadosa dos dados qualitativos. Essa análise permitiu identificar relações e padrões entre os comentários realizados durante as sessões, fornecendo uma compreensão mais profunda dos dados recolhidos. Dessa forma, foi possível obter resultados confiáveis e precisos.

### 3.4.3. Questionário - Avaliação do impacto das mudanças metodológicas na equipa

Para avaliar as mudanças implementadas na metodologia de trabalho de desenvolvimento na equipa do projeto *Campus* foi constituído um questionário a ser aplicado aos membros da equipa de investigação e desenvolvimento. No momento de aplicação dos questionários, a equipa já contava com mais quatro elementos novos, que não acompanharam a investigação e, por essa razão, foram formuladas um conjunto de questões distintas para os mesmos.

Foi considerado que, apesar do *feedback* limitado que estes elementos novos poderiam dar em relação às mudanças resultantes desta investigação, havia interesse em perceber se essas mesmas mudanças eram compreendidas, aceites, e compatíveis com a rotatividade frequente e característica de membros desta equipa de I&D.

Este questionário foi desenvolvido com o intuito de avaliar a eficácia da implementação da metodologia ágil na equipa. O propósito é reunir *insights* importantes que permitam o aperfeiçoamento dos processos e assegurem que a

metodologia ágil seja eficiente e proveitosa para todos os envolvidos. O foco está em compreender como é que essa mudança afetou diversos aspetos do trabalho, abrangendo categorias como:

**Produtividade**: Avaliação das mudanças na produtividade pessoal e da equipa.

**Processo de Execução**: Análise dos processos de trabalho, incluindo gestão de tempo, clareza de tarefas e comunicação.

**Implementação de Soluções**: Opiniões sobre a execução das soluções propostas, adequação ao perfil dos membros da equipa, e níveis de rigidez e controlo.

**Satisfação com a Metodologia Ágil**: Avaliação geral da nova metodologia implementada.

**Feedback e Sugestões**: Espaço para sugestões de melhoria e comentários abertos.

Os questionários para a equipa do Campus by Fundação Altice podem ser consultados no Apêndice 5 - Questionários para a equipa do Campus by Fundação Altice.

Durante esta etapa de conceção do questionário, após deliberação coletiva com os orientadores, optou-se pela utilização da plataforma *FormsUA* como o meio mais apropriado para a criação do questionário final, assegurando a conformidade com as normas do RGPD.

A investigação dos dados quantitativos provenientes das perguntas de resposta fechada foi realizada com a aplicação de técnicas de estatística descritiva. Isso deve-se ao caráter quantitativo destes dados e ao tipo de perguntas envolvidas, que incluem opções de escolha múltipla e escalas Likert. Para esta análise, utilizou-se o programa IBM Statistical Package for Social Sciences (SPSS statistics).

# 4. Recolha de dados

## 4.1. *Focus Group* para a avaliação dos métodos de trabalho e organização atuais e futuros

No dia 21 de março de 2023 realizou-se o *Focus Group* cujo objetivo se centrava numa abordagem introspetiva e crítica às potenciais metodologias de trabalho em equipa a serem implementadas na nossa equipa. Este encontro, intencionalmente excluindo os coordenadores para propiciar um ambiente de partilha aberta e honesta entre os membros da equipa, procurou avaliar e debater metodologias estabelecidas de desenvolvimento, nomeadamente *Scrum* e *Kanban*, e, paralelamente, refletir sobre a aplicabilidade, vantagens, e desvantagens destas no contexto específico da nossa equipa e projetos.

A estrutura do *Focus Group* foi desenhada para proporcionar uma exploração abrangente e ponderada das diversas metodologias, analisando não só as suas características, mas também o potencial impacto da sua implementação no seio do nosso ambiente laboral. Foram inicialmente expostas as metodologias já referidas para contextualizar os participantes, seguindo-se uma análise das vantagens e desvantagens de cada uma delas. A metodologia usada até essa altura também foi submetida a um escrutínio similar, colocando em relevo não só as suas características, mas avaliando se uma transição para *Scrum* ou *Kanban* justificar-se-ia, e, em caso afirmativo, porquê.

Este debate construtivo desenrolou-se em torno de diversas questões fundamentais sobre a adoção de uma nova metodologia de desenvolvimento. Entre elas, aferiu-se as necessidades a variados níveis (tecnológicas, organizativas, recursos humanos, processuais) que uma tal mudança requereria, e como as práticas atuais no ambiente de trabalho da equipa poderiam ser otimizados sob uma perspetiva *Scrum* ou *Kanban*. Outras perguntas incluíram reflexões sobre a frequência de passagens para produção, o papel e envolvimento do cliente em diferentes momentos, a utilidade e aplicabilidade de métricas numa ótica de

alarmística (o tempo de resposta a incidentes, a frequência de falhas de *deploy*, a taxa de erros no ambiente de produção, etc.) e controlo da produtividade da equipa.

Ao convertermos as notas manuscritas feitas no decurso do *Focus Group* em dados analisáveis, a análise pretendeu decifrar e destilar os *insights* e preferências da equipa, identificar consensos e divergências e fornecer recomendações fundamentadas para o caminho metodológico futuro da nossa equipa e projetos. O nosso objetivo é garantir que a metodologia selecionada e implementada não apenas se alinhe teoricamente com as exigências e natureza do nosso trabalho, mas que, crucialmente, seja também relevante e aplicável à escala e realidade da nossa equipa num contexto de bolsas de investigação.

Com base na estrutura para este *Focus Group*, definida previamente no ponto 3.4.1, obtiveram-se os seguintes resultados:

**Perceções gerais sobre as metodologias**

Verificou-se uma inclinação dos participantes para um modelo que, embora baseado em *Scrum*, fosse desprovido, na visão dos participantes, da sua rigidez típica, permitindo uma adaptação mais flexível às necessidades e realidades da equipa.

Diversos tópicos surgiram sublinhando a necessidade de uma estrutura que permita alguma margem de manobra no que diz respeito ao planeamento de *Sprints* e retrospetivas, bem como a padronização e divisão de tarefas, uma vez que, normalmente, as pessoas que compõem a equipa são estudantes de licenciatura ou mestrado que, durante o período de trabalho, têm de se ausentar ou realizar tarefas distintas daquelas que se enquadram tipicamente no âmbito dos projetos da nossa equipa.

**Necessidades e desafios em foco: Tempo e padronização**

A gestão e alocação de tempo foram identificadas como um dos principais desafios, destacando-se a luta contínua contra prazos e a desistência ocasional de tarefas muito complexas ou trabalhosas. Surge um apelo claro para uma

abordagem que permita padronizar tarefas, de modo a minimizar e gerir eficientemente qualquer disparidade nas suas dimensões de tempo e complexidade. Na opinião dos participantes, isto permitirá aos membros da equipa ter uma sensação de progresso recorrente, que por sua vez, poderá aumentar a produtividade da equipa.

**A Importância da Comunicação Diária e Transparente**

A valorização das *Daily Stand-Up Meetings* como mecanismos vitais para atualizar o estado das tarefas foi unânime. Para além disso, estas *Daily Stand-Up Meetings* foram consideradas bastante vantajosas para a transparência no progresso diário de cada membro, evitando conflitos desnecessários e sentimentos de injustiça relativos à carga de trabalho individual quando o tamanho e o grau de exigência das tarefas, alocadas a cada um, não são estandardizados. Para além disso, os participantes também expressaram a necessidade de aprimorar a comunicação e a gestão das expectativas com intervenientes externos, como o coordenador do projeto - professor Carlos Santos, mantendo-o informado e alinhado com os desenvolvimentos do projeto a cada semana.

**Alternância de Papéis e Gestão de Equipa: Um Equilíbrio Necessário**

A dinâmica e gestão de papéis na equipa também foram temas-chave. Enquanto a consistência de ter um gestor fixo foi valorizada, a ideia de um esquema rotativo do papel de *Scrum Master*, que poderia promover novas perspetivas e evitar a sobrecarga de uma única pessoa, foi considerada potencialmente enriquecedora.

**Relação com o Cliente e Transição para Produção**

No decorrer da discussão deste tema foi apresentada uma proposta de integrar o cliente, de uma forma mais próxima, no processo de desenvolvimento. Essa proposta para inteirar o cliente do progresso da equipa consistia na realização de vídeos periódicos que mostrassem avanços significativos. Foi apresentada como um método para promover transparência e envolvimento ativo e mais frequente por parte do cliente.

Ainda sobre este tema, para além desses vídeos, e para que o cliente pudesse ter acesso direto ao progresso feito pela equipa, debatemos o envio das mudanças para o ambiente de *staging* ou produção no final de cada *Sprint*. Para que isso fosse concretizável, foi realçada a necessidade de uma estabilidade dos próprios ambientes e a inevitabilidade de testes rigorosos antes de efetuar qualquer movimento significativo para os mesmos.

Nesta nota, é preciso mencionar que na altura deste *focus group*, o ambiente de *staging* estava sob manutenção por um período indefinido graças à migração dos sistemas para *PHP 8* e o ambiente de produção continha a versão antiga e estável da plataforma *Campus*, que se mantém à data da escrita deste documento. Por isso, dado que o trabalho da equipa era o de implementar a modernização gráfica e técnica da plataforma *Web*, a passagem dessas mudanças para o ambiente de produção estava completamente fora de questão.

No entanto, os participantes concordaram unanimemente que numa fase posterior do projeto, uma vez que o ambiente de *staging* estivesse operacional e a modernização da plataforma presente no ambiente de produção, era saudável a passagem das mudanças no final de cada *Sprint* para um ou ambos os ambientes.

**Projeção para o Futuro: O que Fazer de Forma Diferente?**

Para finalizar, em jeito de exercício conclusivo do *Focus Group*, foi pedido aos participantes que refletissem sobre o que mudariam na nossa equipa, a nível metodológico ou qualquer outro, caso o projeto fosse reiniciado sem nenhum constrangimento financeiro, de recursos humanos, metodológico ou tecnológico. Isto gerou uma oportunidade para que os participantes pensassem sobre um contexto ideal onde gostariam de trabalhar (sendo que os projetos se manteriam os mesmos).

Os participantes mostraram imediatamente um desejo de ter pelo menos mais dois membros integrantes da equipa. Um que assumisse um papel exclusivamente ligado às metodologias de trabalho na equipa, como *Scrum Master*. Como justificação, foi mencionado que essa pessoa podia dedicar todo o seu tempo

à organização da equipa e, assim, libertar os outros membros que naquele momento se tinham de dividir entre o desenvolvimento e essa mesma função. Isso, na visão dos participantes, melhoraria a própria metodologia de trabalho, a gestão de tempo da equipa, e consequentemente a produtividade.

O outro papel que foi reconhecido como necessário foi o de *Chief Technology Officer*. Apesar de a equipa não funcionar numa estrutura semelhante a uma empresa, o termo CTO foi usado como referência para explicar a importância de ter alguém na equipa que dominasse totalmente a nossa infraestrutura de servidores e código. Na perspetiva dos participantes, ter alguém com este perfil na equipa era extremamente importante para a manutenção da nossa base de trabalho nessas áreas, ou seja, iria resultar numa melhor organização dos nossos micro serviços, atualizações de segurança regulares e uma comunicação mais próxima com os gestores dos nossos próprios servidores, localizados e sob a tutela da Direção de Infraestrutura e Tecnologia (DIT) da Altice.

Para além disto, foi manifestada também a preocupação com a rotatividade constante da equipa, o que se deve ao facto de ser constituída integralmente por bolseiros de investigação, com todas as limitações que daí advêm. Para melhorar a integração e a passagem de testemunho para os eventuais novos membros, foi sugerida a elaboração de uma apresentação que contemplasse a organização da nossa metodologia de trabalho, infraestrutura de sistema e código, quer a nível de *Front-End* como de *Back-End*.

Neste último ponto, foi exposto por vários membros o facto de, mesmo após dois anos fazendo parte da equipa, ainda existirem vários setores dos projetos que não eram completamente entendidos, e que existia pouca documentação para apoiar a sua compreensão. Isto levava a um dispêndio de tempo individual de cada um para aprender certos conceitos por tentativa-erro. Para evitar isto, foi acordado que dali para a frente, uma vez que se estava a reestruturar muitos componentes da plataforma *Campus*, sempre que alguém se deparasse com um conceito, componente ou sistema mais complexo da plataforma, se faria um esforço para documentar esse mesmo elemento de uma forma mais extensa para que, no futuro,

caso alguém encarasse o mesmo dilema, tivesse um apoio extra na documentação do projeto.

Neste cenário hipotético, foi salientado pelos participantes a necessidade de existir um processo mais sólido e simples de manutenção e controlo de versões nos diferentes ambientes. Isto é, existirem novas formas, com uma curva de aprendizagem menor, para gerir os diferentes projetos e realizar certas ações como o envio de novas alterações para produção com a segurança de testes, para que nada além do pretendido seja afetado pelas novas mudanças. Uma solução para este problema iria facilitar não só o envio de mudanças para produção, mas também o processo de desenvolvimento e testagem no ambiente de *staging* e o acesso a essas mudanças por parte da equipa de design, coordenadores e cliente, aumentando assim a velocidade e frequência de *feedback* por parte destes intervenientes. Nesta discussão todos concordaram que indicadores como as *DORA Metrics* seriam interessantes, no entanto, não foi algo muito aprofundado dado o reconhecimento por todos de que existiam prioridades mais básicas ainda por cumprir e tais indicadores seriam muito difíceis de implementar tendo em conta as restrições de infraestrutura que se faziam sentir na altura.

## *4.2.* Entrevistas com especialistas, ex-membros da equipa *Campus by Fundação Altice*

Analisando as gravações das entrevistas realizadas depois do *Focus Group,* durante os meses de abril e março (mediante a disponibilidade dos três entrevistados), foi possível identificar diversos pontos relevantes relacionados às metodologias de trabalho em equipas de tecnologias da informação, ferramentas de desenvolvimento e gestão de infraestruturas tecnológicas, tal como previamente estabelecido no guião. Nesta secção é feita uma análise aos pontos mais relevantes e às sugestões dadas pelos especialistas.

As soluções técnicas sugeridas nas entrevistas abrangem uma gama de práticas e ferramentas modernas, refletindo uma abordagem atualizada e eficiente no contexto do desenvolvimento de software.

### 4.2.1. Metodologias ágeis e gestão de equipa

Todos os entrevistados enfatizam a importância das metodologias ágeis no contexto das suas experiências profissionais. Falou-se sobre como a organização e a comunicação eficiente são cruciais, especialmente em equipas com diversidade de funções e competências. Foi relatada a aplicação de *Scrum* nas experiências profissionais dos entrevistados, destacando-se a importância de se ter as tarefas bem distribuídas e organizadas. Foi valorizada a *daily meeting* e a estrutura de *Sprints* para manter a equipa alinhada e os projetos e tarefas em andamento.

Segundo as opiniões dos entrevistados, aplicar a metodologia *Scrum* numa equipa como a do *Campus by Fundação Altice* pode trazer várias vantagens. A metodologia, de acordo com os mesmos, proporciona uma estrutura clara para a gestão de tarefas e projetos, promove uma comunicação eficiente e facilita a colaboração entre membros com diferentes competências e funções. No entanto, é consensual que é importante adaptar as práticas do *Scrum* às especificidades da equipa, como o tamanho da equipa, a natureza dos projetos e o nível de experiência dos membros com metodologias ágeis. A implementação bem-sucedida do *Scrum* requer um compromisso com a transparência, a adaptação contínua e o envolvimento ativo de todos os membros da equipa.

Esta convergência nas opiniões sublinha a relevância das metodologias ágeis na gestão eficaz de projetos de TI.

### 4.2.2. Ferramentas de Desenvolvimento e Infraestrutura

No que diz respeito às ferramentas de desenvolvimento, há um consenso sobre a importância de ferramentas de automação e *CI/CD*. Para tal, foi indicado que a estrutura da *pipeline* deve ser iniciada com um sistema de controlo de versões robusto, como o *Git*, utilizando as diferentes *branches* estrategicamente, como *feature branches*, para organizar o desenvolvimento e facilitar a revisão de código. A implementação de *builds* automáticas para cada *commit* ou *pull request* é essencial para garantir a boa execução do código e a ausência de erros básicos no mesmo.

Para prevenir também esses erros, quanto aos testes automáticos, realçou-se a importância de implementar testes unitários em componentes críticos, utilizando ferramentas adequadas como *Jest*. Os testes de integração são cruciais para validar a interação entre diferentes módulos ou serviços, enquanto, para os testes *end-to-end*, foram indicadas ferramentas como *Selenium* ou *Cypress*, que simulam a experiência do utilizador final.

Simultaneamente, para a integração contínua (CI), a utilização de ferramentas como *Jenkins, CircleCI,* ou as próprias funcionalidades de *CI/CD* da plataforma *GitLab* são recomendadas para automatizar a integração do código. Estas ferramentas devem ser configuradas para executar testes automaticamente e fornecer *feedback* rápido aos programadores da equipa.

O uso de *Docker* foi destacado como essencial para criar ambientes consistentes, garantindo a uniformidade do software em todos os ambientes de desenvolvimento. Para aplicações *multi-container*, já antevendo a migração total da infraestrutura do Campus para esta tecnologia, foi dada uma breve explicação e várias sugestões por um dos entrevistados, mencionando que o *Docker Compose* é uma solução prática para definir e gerir esse tipo de ambientes.

Em relação ao *Continuous Deployment/Delivery (CD)*, sugere-se a automação de *deploys* para o ambiente de *staging* assim que as alterações forem integradas e aprovadas.

Por fim, a importância da documentação clara e acessível para toda a equipa foi enfatizada, assim como a promoção de uma cultura de revisão de código e colaboração para melhorar a qualidade e a manutenção do código.

Em jeito de conclusão, na análise integrada das entrevistas sobressai a complexidade e a dinâmica do desenvolvimento de *software* em ambientes como o da equipa do Campus. As metodologias ágeis, particularmente o *Scrum*, surgem como fundamentais para a gestão eficiente de projetos, enquanto as ferramentas de automação e *CI/CD* são cruciais para otimizar processos e garantir a qualidade do produto final. A adaptação contínua às novas tecnologias e a importância de

uma cultura colaborativa são aspetos essenciais para o sucesso de uma equipa de desenvolvimento tecnológico. Estes *insights* são particularmente valiosos para equipas como a nossa, que procuram melhorar as suas práticas e adaptar-se às rápidas mudanças que se fazem sentir no mundo da tecnologia.

Para tal, todas estas práticas sugeridas pelos entrevistados, se bem implementadas, podem melhorar significativamente a eficiência, qualidade e confiabilidade do processo de desenvolvimento da nossa equipa, sendo fundamental que a *pipeline* seja adaptável e revista continuamente, alinhando-se com as necessidades em evolução do projeto e as novas tecnologias disponíveis.

# 5. Implementação e desenvolvimento

Neste capítulo, proceder-se-á à descrição da implementação das soluções anteriormente debatidas, seguindo os dois planos de ação definidos. Inicialmente, focar-se-á o nível das metodologias de desenvolvimento e, subsequentemente, a execução prática no que diz respeito ao desenvolvimento de ferramentas e infraestrutura.

## 5.1. Metodologias de desenvolvimento

Na sequência dos resultados obtidos no *Focus Group*, tornou-se imperativo avançar com reformulações que visassem otimizar os processos da nossa equipa de desenvolvimento. Entrámos, então, num regime de *Sprints* iterativos, aprimorando e calibrando as nossas metodologias a cada ciclo, com o intuito de aumentar a eficiência e eficácia do nosso trabalho, cumprindo com a metodologia escolhida inicialmente para a investigação, que prevê ciclos de avaliação iterativos.

### 5.1.1. A otimização iterativa dos *Sprints*

A descrição das medidas adotadas, embora apresentada sem uma hierarquia específica, enuncia as decisões tomadas de acordo com o que foi discutido no *Focus Group* e também com o *feedback* contínuo no final de cada *Sprint*.

#### 5.1.1.1. Reuniões diárias obrigatórias

Apesar de já ser um hábito da equipa, instituímos as "*Daily Stand-up Meetings*" como obrigatórias e essenciais ao bom funcionamento da equipa, garantindo um ponto de contacto diário onde se discutiam os avanços, obstáculos e planeamentos para o dia.

#### 5.1.1.2. Comunicação aprimorada com coordenadores

Intensificámos o diálogo com os coordenadores do projeto, realizando reuniões semanais às sextas-feiras (*Sprint Review*), onde expúnhamos os progressos e resultados alcançados. Estas reuniões já ocorriam previamente, mas

sem uma data fixa estabelecida e, algumas vezes, sem a estrutura que uma *Sprint Review* exige.

#### 5.1.1.3. Retrospetivas pós-*Sprint*

A cada final de *Sprint*, passámos a dedicar uma sessão na sexta-feira à tarde para uma análise sobre o que correu bem, o que podia ser melhorado, e como podemos implementar essas melhorias no próximo *Sprint*. A regularidade destas sessões, no final de cada *Sprint*, assegura que a equipa mantenha um ritmo de autoavaliação e adaptação, que são fundamentais para o crescimento e sucesso num ambiente ágil. Além disso, ao agendar estas reuniões para a sexta-feira à tarde, beneficia-se do encerramento natural do ciclo semanal de trabalho, permitindo que a equipa inicie o novo *Sprint* na segunda-feira com um plano de ação claro e um foco renovado nos objetivos estabelecidos.

#### 5.1.1.4. Padronização de tarefas

Refinámos a especificação das tarefas, tornando-as mais claras, realistas e, quando necessário, dividindo-as em micro tarefas para facilitar a sua execução. Assim, evitam-se situações como as que aconteciam anteriormente, em que algumas tarefas, por terem uma natureza tão complexa, se arrastavam por múltiplos *Sprints* consecutivos.

#### 5.1.1.5. Reorganização do *Trello*

O nosso quadro no *Trello (*Figura 5*)*, plataforma eleita para a gestão das tarefas da equipa, foi reestruturado, distinguindo claramente as tarefas do *Sprint* corrente das que estão em *Backlog* ou de outras tarefas de menor dimensão que estão guardadas para momentos de menos intensidade - tipicamente tarefas de estilização ou erros de pouco impacto para o utilizador final.

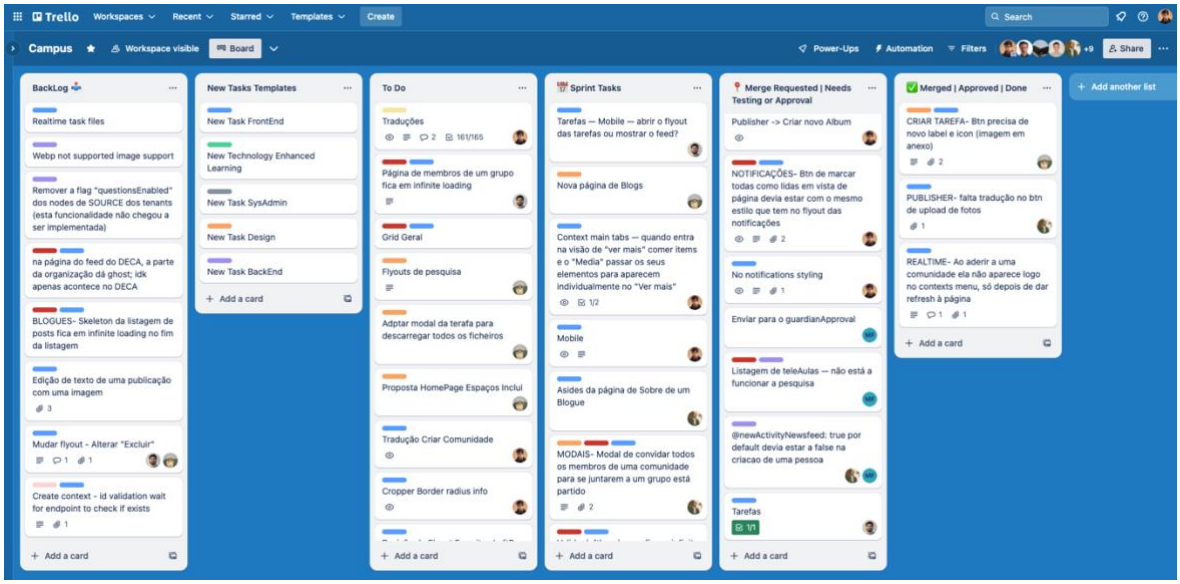



*Figura 5 - Visão da plataforma Trello*

### 5.1.1.6. Participação ativa da equipa de design

Envolveu-se a equipa de design no controlo de qualidade antes da integração final das funcionalidades na *branch* principal de desenvolvimento. Isto permitiu libertar mais tempo à equipa encarregue de fazer a revisão do código, uma vez que quando tem efetivamente de se dedicar a tal, é menos provável ainda existirem erros de interação ou compatibilidade em diferentes ecrãs. Apesar disto, é necessário notar que existiu uma ótima interação entre estas duas equipas neste processo, estando ambas ativamente a contribuir para essa avaliação. Isto possibilitou que a equipa de revisão do código se focasse mais nos aspetos técnicos, sem descurar a qualidade das funcionalidades desenvolvidas que, pelo contrário, aumentou graças à exigência e atenção ao detalhe da equipa de design no processo de avaliação.

### 5.1.1.7. Ambiente de desenvolvimento

Com a inexistência de uma *pipeline* ou de um ambiente de *staging* funcional, o ambiente de desenvolvimento tornou-se crucial para testes pela equipa de design e para apresentações aos coordenadores. Como plano de contingência para mitigar a falta do ambiente de *staging* e a impossibilidade de fazer passagens para produção, ficou delineado que seriam utilizadas diferentes *branches* no nosso

repositório de código de *Front-End*. Assim, surgiram as *branches campus-redesign* e *WIP* (*Work in Progress*). Estas branches foram usadas como alternativa para substituir estes ambientes. Assim, a *WIP* era utilizada como o ambiente de *staging* e *campus-redesign* como produção. O trabalho diário de cada membro era feito em *feature branches* que, no final, eram *merged* para *WIP*. Este *merge* passava pelo processo de avaliação da equipa de design, e verificação técnica, tal como descrito previamente. A *branch develop* manteve-se inalterada com o código correspondente à versão presente no ambiente de produção.

#### *5.1.1.8.* Antecipação do trabalho de *Back-End*

A antecipação do trabalho de *Back-End* é uma estratégia proativa adotada pela equipa para garantir uma integração mais fluída e eficaz com o *Front-End*. Ao começar a trabalhar nas tarefas de *Back-End* antes da equipa de *Front-End*, cria-se uma base sólida e pronta a ser conectada à interface com o utilizador, evitando assim atrasos que possam surgir quando ambas as equipas tentam progredir em paralelo sem um ponto de integração definido.

Este ponto fez-se sentir no desenvolvimento do Fora d'Aulas. Muita da lógica, senão toda, referente ao Back-End em relação ao novo produto do Campus, foi iniciada e terminada antes mesmo de se ter começado a implementação do Front-End. Isto assegura que, num futuro próximo, o desenvolvimento desta ferramenta irá ser mais célere em relação à experiência de outros produtos do passado, sem que se tenha de aguardar pelos programadores do Back-End para que se possa implementar toda a camada de visualização na interface do Campus.

#### 5.1.1.9. Documentação no *Notion*

A documentação no *Notion* (uma plataforma de documentação online) surgiu como uma medida estratégica para a gestão do conhecimento dentro da equipa (Figura 6). Esta plataforma tornou-se um repositório centralizado onde são documentados tanto os desafios encontrados no decorrer do projeto quanto as soluções adotadas e as melhores práticas desenvolvidas. O objetivo principal desta iniciativa é simplificar o processo de acolhimento de novos membros da equipa,

proporcionando-lhes acesso imediato a um vasto leque de informações e recursos essenciais para a sua integração e produtividade. A documentação favorece a continuidade do projeto a longo prazo, pois mesmo na ausência de membros que tenham contribuído com soluções importantes, a equipa mantém-se autossuficiente, com acesso às informações necessárias para avançar.


*Figura 6 - Organização da documentação no Notion*

#### 5.1.1.10. Registo em tempo real no *Teams*

Implementámos um registo automático dos *commits*, promovendo transparência e mantendo toda a equipa informada acerca das atualizações da base de código. Este registo em tempo real dos *commits* no *GitLab* através do *Microsoft Teams* (plataforma de eleição para a comunicação da equipa) é um exemplo de como a integração de ferramentas pode potencializar a transparência e a comunicação numa equipa de desenvolvimento. Utilizando *webhooks* do *GitLab*, configurámos um sistema automatizado que publica notificações, num canal dedicado para esse efeito, sempre que ocorre um *commit*, assegurando que

toda a equipa está a par do trabalho que está a ser efetuado pelos outros membros, especialmente em ocasiões de trabalho remoto.

#### 5.1.1.11. Comunicação regular com o cliente

Tal como foi discutido no *Focus Group*, houve uma tentativa de aproximação ao cliente, demonstrando o nosso compromisso em entregar produtos de excelência, fortalecendo as relações da equipa com os *stakeholders*.

Apesar da proposta no *Focus Group* ter estado direcionada à realização de vídeos periódicos que mostrassem avanços significativos, foi demonstrada a preferência do lado do cliente pela marcação mínima de um contacto bimestral com a equipa. Esta solução satisfez a intenção de nos aproximarmos da Fundação Altice, mantendo assim a nossa intenção de um alinhamento mais próximo com as expectativas e necessidades do cliente.

#### 5.1.1.12. Ronda de testes antes da *Sprint Review*

Em *Sprints* em que o progresso feito pela equipa correspondia a funcionalidades importantes, ou havia muitas mudanças significativas, ou seja, *sprints* que correspondiam a um *epic,* passaram a ser feitas sessões adicionais, no dia anterior à reunião de revisão do *Sprint*, com o objetivo de garantir uma versão estável e pronta a ser apresentada aos *stakeholders*. Estas sessões têm a presença da equipa de design e a equipa de programação, nomeadamente os membros responsáveis pelas mudanças que vão estar sob escrutínio.

#### 5.1.1.13. Deliberação da duração do *Sprint*

No processo de desenvolvimento após o *Focus Group*, foi testada a duração dos *Sprints* de catorze ou de sete dias para a conclusão de um conjunto de tarefas. Durante o teste dessas duas abordagens, ficou evidente que os *Sprints* de sete dias apresentaram vantagens significativas em relação aos *Sprints* de catorze dias. A principal razão para essa preferência foi a maior frequência de comunicação e ajuste de objetivos entre os programadores, coordenadores e outras partes envolvidas no processo.

Nos *Sprints* de sete dias, as equipas tinham a oportunidade de rever e adaptar os seus objetivos e estratégias com muito mais frequência. Isso permitia uma resposta ágil a mudanças nas prioridades, requisitos do cliente ou novos *insights* surgidos durante o desenvolvimento. Além disso, a comunicação regular era mais eficaz, pois as reuniões de acompanhamento e retrospetivas aconteciam em intervalos mais curtos.

Por outro lado, os *Sprints* de catorze dias mostraram-se menos ágeis e mais suscetíveis a desalinhamentos e problemas de comunicação, embora se poupasse tempo nos rituais da metodologia, como as retrospetivas e *reviews*. A longa duração desses *Sprints* tornava mais difícil a adaptação rápida a mudanças inesperadas, o que podia levar a atrasos no alcance dos objetivos definidos, que pelo *Sprint* ser mais longo, eram mais complexos ou em maior número.

A escolha de *Sprints* de sete dias em vez de *Sprints* de catorze dias foi motivada pela necessidade de uma comunicação mais frequente e pela capacidade de ajustar metas e objetivos de forma mais ágil. A maior frequência dos *Sprints* mantinha a equipa de desenvolvimento altamente envolvida e comprometida e os membros sentiam que estavam sempre próximos de alcançar os resultados, o que aumentava a motivação e a responsabilidade. Essa abordagem provou ser mais eficaz para manter o desenvolvimento alinhado com as necessidades da equipa e do projeto, garantindo uma entrega mais eficiente e de alta qualidade.

### 5.1.2. Teoria dos vidros partidos

No decorrer destas iterações, identificámos um ponto de falha que a equipa reconheceu como intrínseco e anterior ao teor deste processo de mudança na metodologia de trabalho da equipa. Uma vez que sempre se batalhou contra a falta de tempo e necessidade constante de desenvolver novas funcionalidades para as plataformas, eram muitas vezes consideradas completas, tarefas isentas de testes de qualidade. Foi consensual que esta prática introduzia demasiados *bugs* que impactavam negativamente a experiência de utilizador nas plataformas.

A teoria dos "vidros partidos" é uma teoria criminal que se baseia na ideia de que o crime e o comportamento antissocial podem ser influenciados e até mesmo exacerbados pelo ambiente físico e social de uma determinada área geográfica. Esta teoria parte da metáfora de uma janela partida num edifício abandonado. A ideia central é que se uma janela partida não for reparada imediatamente, envia um sinal de desleixo e desordem para a comunidade. Isso, por sua vez, pode levar a um aumento da criminalidade e do comportamento destrutivo na área. A teoria argumenta que a visibilidade de sinais de desordem, como graffiti, lixo nas ruas, janelas partidas e outros pequenos delitos, cria um ambiente propício para crimes mais graves. (Kelling et al., 1982)

O paralelismo entre a teoria dos "vidros partidos" e uma plataforma *Web* onde existem bugs é fácil de entender. Neste contexto, os "vidros partidos" podem ser comparados aos *bugs* não resolvidos na plataforma. Assim como janelas partidas e sinais visíveis de desordem criam uma impressão de negligência num bairro, a presença de *bugs* não resolvidos no código-fonte da plataforma *Web* pode criar uma impressão de descuido no desenvolvimento. Os programadores podem começar a desvalorizar o código existente e o processo de desenvolvimento quando veem que os *bugs* não estão a ser tratados adequadamente. Podem interpretar a presença de *bugs* como um sinal de que os responsáveis não se importam nem com a qualidade do código nem com a qualidade do processo de desenvolvimento.(Holvitie et al., 2018)

A falta de resolução de *bugs* pode ainda levar a problemas mais graves no desenvolvimento, como a acumulação de problemas técnicos e a dificuldade de manutenção do código. Isso pode afetar negativamente a eficiência e a produtividade dos membros da equipa de desenvolvimento.

Assim como a teoria dos vidros partidos sugere que a prevenção do crime se deve concentrar na correção imediata de sinais de desordem, a prevenção de *bugs* deve ser uma prioridade no desenvolvimento. É essencial que os programadores identifiquem e resolvam os *bugs* nas funcionalidades em desenvolvimento antes de prosseguirem para novas tarefas. A conclusão efetiva

de cada funcionalidade, com a devida correção de erros, é prioritária antes do avanço para outras atividades.

É também importante que os responsáveis incentivem uma cultura de qualidade e cuidado no desenvolvimento. Isso significa que os programadores devem ser reconhecidos pelo trabalho bem feito e pela criação de código de alta qualidade, incluindo a correção de *bugs*.

Da mesma forma que a vigilância comunitária envolve a colaboração com a comunidade, os programadores devem estar abertos ao *feedback* dos *stakeholders* para identificar problemas e *bugs*. A comunicação eficaz entre ambos é essencial para resolver problemas rapidamente.

O paralelismo enfatiza a importância de uma manutenção constante do código da plataforma *Web* para garantir que os problemas sejam abordados prontamente. Isso ajuda a manter o brio e o cuidado dos programadores ao desenvolver novas funcionalidades, pois sabem que o código existente está em boas condições.

Em resumo, a presença de *bugs* não resolvidos no processo de desenvolvimento de uma plataforma *Web* pode afetar o brio e o cuidado dos membros da equipa no processo de desenvolvimento de novas funcionalidades na plataforma, assim como na teoria dos "vidros partidos" a desordem pode afetar a perceção da qualidade de uma área. Portanto, a resolução ágil de *bugs* e a promoção de uma cultura de qualidade são fundamentais para manter a motivação e o profissionalismo da equipa.

Este exercício de paralelismo foi exposto à equipa como forma de sensibilizar os membros a mudar a sua abordagem ao desenvolvimento, priorizando a correção de *bugs* e mais cuidado aquando do desenvolvimento e introdução de novos elementos à nossa base de código.

### 5.1.3. Resultados intercalares

Depois de algumas iterações, aplicando sucessivamente estas mudanças à nossa metodologia, o *Sprint* que teve lugar na segunda semana de junho de 2023 foi marcante para toda a equipa porque foram identificadas melhorias palpáveis nos resultados alcançados. Este *Sprint* teve a particularidade de ser próximo de um ponto de situação com os nossos parceiros da Fundação Altice, o que também contribuiu para uma maior seriedade no trabalho efetuado e para o perfecionismo do mesmo.

Este momento destacou-se porque o empenho de todos os membros, o rigor e respeito pelas cerimónias relativas à metodologia e a entreajuda entre as equipas de trabalho foi excecional ao ponto de alcançarmos os objetivos a que nos tínhamos proposto no início do *Sprint*. Esses objetivos diferiram de um *Sprint* comum, visto que marcavam o fim de uma série de tarefas que eram essenciais para que tivéssemos um MVP (*Minimum Viable Product*) do *redesign* da plataforma *Campus*.

Por todas estas razões, foi destacada na reunião de retrospetiva a satisfação dos coordenadores pelo sucesso desse *Sprint*, mas também a preocupação em, a partir daquele momento, manter aquela forma de trabalho. Foi um momento necessário e marcante para toda a equipa a nível de moral e sentimento de objetivo concluído.

## 5.2. Desenvolvimento prático

### 5.2.1. Estudo de viabilidade técnica com parceiros

Como forma de validação de possíveis soluções, foi decidido reunir com os nossos parceiros da *Direção Tecnologias Informação* (DIT) da Altice. Esta reunião serviu essencialmente para debater soluções e saber a opinião dos gestores dos nossos servidores quanto à exequibilidade técnica das mesmas, segundo as limitações presentes nos nossos sistemas. Para além disso, também tínhamos como objetivo perceber como é que esses parceiros e as suas equipas gerem outros projetos e se existiam outras soluções possíveis em cima da mesa.

Nessa conversa, foi dado o aval para procedermos à melhoria dos nossos sistemas, sendo que teríamos o seu apoio para as intervenções necessárias mediante um pedido formal através da plataforma interna do nosso projeto. No entanto, ficámos pela primeira vez a saber que o tipo de servidores que usamos eram, já à data, considerados uma exceção relativamente à norma que outros projetos geridos pela DIT usavam.

Os nossos servidores atualmente funcionam com o sistema operativo *Debian*, uma distribuição *Linux* de código aberto amplamente reconhecida pela sua estabilidade e flexibilidade. No entanto, descobrimos uma distinção significativa em relação à maioria dos outros projetos da organização, que utilizam a licença corporativa *RedHat*.

O *Debian*, por ser uma distribuição de código aberto, oferece uma gama diversificada de pacotes de software que podem ser instalados de acordo com as necessidades específicas do nosso projeto. No entanto, a sua manutenção e atualizações requerem intervenção manual por parte da nossa equipa, o que pode ser mais demorado e propenso a erros. A sua utilização está relacionada com o histórico do projeto, sendo que numa fase anterior a infraestrutura tecnológica utilizada era do SAPO, sendo o *Debian* o sistema operativo de eleição para essa equipa.

Por outro lado, a licença corporativa *RedHat*, representada pelo *Red Hat Enterprise Linux* (*RHEL*), traz consigo diversas vantagens. O *RHEL* é conhecido pelo seu suporte técnico dedicado, garantia de segurança e atualizações frequentes. A sua maior vantagem reside na automação abrangente, que simplifica a gestão de servidores, a implementação de políticas de segurança e as atualizações em grande escala.

Além disso, os utilizadores do *RHEL* têm acesso garantido a recursos de segurança e correções de *bugs*, garantindo que os servidores permanecem seguros e estáveis ao longo do tempo. A automação desses processos também torna a manutenção mais eficiente, poupando tempo e reduzindo os riscos de erros humanos.

No entanto, é importante sublinhar que a escolha entre *Debian* e *RedHat* depende das necessidades específicas do nosso projeto e das preferências da nossa equipa. Ambas as distribuições têm as suas vantagens, e a decisão final deve ser cuidadosamente ponderada à luz das circunstâncias individuais do nosso projeto.

À data desta reunião, essas circunstâncias não eram favoráveis para se pensar numa transição imediata para *RedHat*. O facto de estarmos ainda a meio da atualização para versão de *PHP 8* com o ambiente de *staging* inoperacional era uma das premissas que impediam essa hipótese. Para além disso, uma vez que os nossos serviços estavam dispersos por vários servidores, cada um deles com necessidades diferentes, era expectável que a transição para outro sistema, que não o *Debian*, não fosse uma tarefa de fácil execução. Esta possibilidade surgiu porque foi mencionada a nossa vontade de colocar toda a nossa infraestrutura em *Docker*. Caso isso fosse alcançado, a dificuldade de migração para *RedHat* seria significativamente menos complexa e mais natural, uma vez que traria todas as vantagens acima descritas sem grande investimento de tempo nem de recursos humanos, dada a natureza automática de maior parte dos processos do lado das equipas de gestão dos nossos servidores.

Assim, ficou decidido que a abordagem correta seria tentar replicar a nossa infraestrutura em *Docker*, para depois se migrar a mesma para os sistemas corporativos da Altice em *RedHat*. Resumindo, a reunião com os parceiros da Altice abriu a porta para uma possível transição para o *RHEL*, mas essa migração devia ser feita com base em critérios bem definidos e alinhados com os objetivos e recursos do nosso projeto.

### 5.2.2. Plano de implementação

Com a constante evolução das tecnologias e metodologias de desenvolvimento, é imperativo que a infraestrutura acompanhe estas mudanças, garantindo assim uma otimização contínua dos processos e uma maior eficiência na entrega de soluções. Neste contexto, o presente subcapítulo aborda o plano de implementação elaborado considerando as possibilidades existentes, apuradas até

a este momento da investigação, através do *Focus Group*, entrevistas a especialistas e consulta da literatura.

#### 5.2.2.1. Estudo das possibilidades

Com base na informação recolhida previamente, foi necessário delinear esse plano de implementação de mudanças significativas na infraestrutura dos nossos projetos. Para isso, de entre as várias opções que estavam em discussão, foi essencial selecionar aquela que traria mais benefícios à equipa, mas que ao mesmo tempo, fosse realista e alcançável de acordo com os recursos e o tempo que tínhamos à nossa disposição.

**Opção 1 - Infraestrutura atual com upgrades de *CI/CD***

À data, a infraestrutura que estava a ser utilizada, apresentava potencialidades que podiam ser maximizadas com a incorporação de *upgrades* focados em *CI/CD*. Uma vez que todos os nossos repositórios se encontram guardados numa instância do *GitLab*, existe suporte nativo para a adição de *pipelines* na plataforma. Estas *pipelines* permitiriam uma maior automatização e flexibilidade nos processos de desenvolvimento com múltiplas funcionalidades que estão disponíveis na plataforma.

É necessário esclarecer que, apesar de maior parte dos repositórios serem capazes de suportar uma *pipeline*, aquele onde mais se justifica a sua aplicação é no repositório que aloja o código fonte da camada visual do *Front-End*, ou seja, o de *React*. Isto porque é o repositório que sofre mais alterações diariamente e onde o número de membros da equipa a contribuir para o seu desenvolvimento é maior, estando na altura alocados quatro membros para o *Front-End* e apenas um para o *Back-End*. Por essa razão, as funcionalidades da criação de uma pipeline e os seus efeitos positivos iriam ser mais notórios neste repositório. Dessas funcionalidades destacam-se algumas que foram por consideradas essenciais, para uma fase inicial de implementação:

- **Implementação de estados de *build* e *deploy***. Esta funcionalidade permite a configuração de múltiplos estados no processo da *pipeline,* otimizando a gestão do ciclo de vida da aplicação. Através da definição destes estados, como *'build'*, *'test'*, e *'deploy'*, é possível controlar com precisão cada fase do desenvolvimento e a otimização das mesmas com a utilização de cache, por exemplo para a fase de *build*, onde são instaladas as dependências do projeto. A automatização destes estados contribui para a redução de erros humanos e para a agilização da disponibilização de novas versões das plataformas;

- **A implementação de testes automáticos**. A integração de testes automáticos nas pipelines do *GitLab* é importante para garantir a qualidade e a estabilidade do código. Estes testes podem ser executados em cada *commit*, assegurando que as alterações recentes não comprometem as funcionalidades existentes. Além disso, facilitam a identificação e correção de *bugs* em estados iniciais do desenvolvimento, economizando tempo e recursos que seriam gastos em fases posteriores;

- **Monitorização em tempo real de estados da pipeline e ambientes**. A capacidade de monitorizar o estado das pipelines em tempo real possibilita uma visão clara e atualizada do processo de *CI/CD*. Esta visibilidade permite a deteção imediata de falhas e a rápida intervenção para a sua resolução, para além de permitir o acompanhamento da progressão de cada fase da pipeline, desde a construção até ao *deploy*;

- **A possibilidade de reverter automaticamente para uma versão anterior em caso de falha no *deploy***. A funcionalidade de *rollback* automático é um mecanismo de segurança crucial que assegura a estabilidade dos vários ambientes, principalmente o de produção. Em caso de falhas detetadas após a implementação de novas versões, o sistema pode reverter automaticamente para o último estado estável, minimizando o tempo de inatividade e o impacto no utilizador final.

- **O lançamento de versões em ambientes estáticos e dinâmicos**. Isto significa que é possível fornecer uma pré-visualização automática das alterações

efetuadas numa *feature branch*, criando um ambiente dinâmico para os *merge requests*. Isto permite que a equipa de design, a equipa de testes e os coordenadores possam ver as alterações implementadas sem precisarem de verificar a *branch* específica que, caso esta funcionalidade não exista, tem de passar por um processo manual de *deploy* para o ambiente de desenvolvimento ou *staging*. Esse processo, naturalmente, consome tempo e recursos humanos, nomeadamente a pessoa encarregue pela administração do sistema.

Apesar de todas estas vantagens, é necessário indicar que esta e outras abordagens semelhantes, que visem o repositório do *Front-End*, sofrerão com uma dificuldade previsível relacionada com a nossa infraestrutura de servidores. Atualmente, os servidores dos ambientes de desenvolvimento e *staging* não têm uma conexão aberta à internet. Qualquer pedido que necessite de um recurso online, tem obrigatoriamente de passar num *proxy* de gestão que nos é fornecido pelos parceiros do departamento de cibersegurança da DIT/Altice. Esta limitação trará desafios acrescidos a todo o processo, como por exemplo a instalação nos servidores dos *runners* da pipeline do *GitLab* ou até a simples instalação de todas as dependências de *React* do nosso projeto.

Apesar desta limitação previsível, esta solução, quando comparada às próximas a serem apresentadas, é mais pragmática, exigindo um menor esforço do ponto de vista de investimento de tempo e recursos humanos, uma vez que é a mais compatível com a infraestrutura atual, já que não é previsível que haja a necessidade de efetuar muitas mudanças ao funcionamento atual dos nossos servidores.

**Opção 2 - *Docker***

A migração da infraestrutura para a utilização de *containers* de *Docker* representaria um marco significativo na modernização e otimização dos processos de desenvolvimento e operações. Esta transição para containers permitiria encapsular os serviços usados nos ambientes de desenvolvimento, *staging* e produção, facilitando a transição de novas versões entre os mesmos. Por isso, ao

adotar o *Docker*, beneficiaríamos de uma consistência entre os ambientes, reduzindo as discrepâncias que frequentemente ocorrem devido a configurações divergentes. A uniformidade proporcionada pelos containers *Docker* também simplificaria o processo de *CI/CD*, pois cada serviço é desenvolvido, testado e implementado dentro de seu próprio container, independentemente de outros serviços. Assim, a migração para *Docker* não só fomentaria a agilidade e eficiência operacional, como também fortaleceria a infraestrutura com robustez e flexibilidade.

Esta opção surge, não só no seguimento dos resultados obtidos nas entrevistas, mas também de uma solução parcial, previamente desenvolvida por um ex-membro da equipa do *Campus*, focada na transição da infraestrutura relativa ao *Back-End* para containers de *Docker*. Essa solução parcial contemplava parte do *upgrade* para *PHP 8* dos repositórios de *Back-End*, a atualização de diversos pacotes e dependências necessárias ao funcionamento desses repositórios e a modernização da base de código para acomodar as necessidades da nova versão de *PHP*.

Quando comparada com a solução apresentada na Opção 1, a continuação deste recurso aqui proposto apresentou-se como uma tarefa significativamente mais complexa e trabalhosa, especialmente tendo em conta a pouca ou total ausência de experiência prévia apresentada pelos membros atuais da equipa em *Docker*. Por essa razão, foi considerada a opção de dividir esta solução em várias complementares, mas funcionais individualmente, focando-se cada uma delas num serviço ou parte da infraestrutura.

A **primeira**, seria a continuação do recurso já desenvolvido, focado no *Back-End* e nos serviços que suportam a infraestrutura dos vários *tenants* do Campus. Isto implicaria a atualização da base de código de PHP 5.6 para PHP 8 de todos os repositórios e microserviços. Esta tarefa acarreta uma enorme complexidade, visto que a evolução do PHP entre estas duas versões incluiu alterações significativas na linguagem e na forma como o PHP é executado, o que implica uma revisão abrangente do código existente. Por exemplo, muitas funções que eram comuns no PHP 5.6 foram descontinuadas ou tiveram seu comportamento alterado no PHP 8. Isso significa que o código que dependia dessas funções precisaria de ser

atualizado ou substituído por alternativas modernas, o que pode ser um processo trabalhoso e propenso a erros.

Além disso, a compatibilidade dos módulos representa um grande desafio. Módulos que eram amplamente utilizados em versões antigas não foram atualizados para suportar PHP 8, ou podem ter sido substituídos por outras soluções mais atuais. Isto requer que se encontrem ou desenvolvam alternativas adequadas para esses módulos, o que pode ser complicado se a documentação for escassa ou inexistente.

Por fim, o Campus tem dependências e integrações específicas que foram construídas para trabalhar com a versão 5.6 do PHP e que não são diretamente transferíveis ou compatíveis com o ambiente do PHP 8, como é o caso do SAPO *Broker*, que gere toda a camada lógica de eventos dos vários *tenants* da tecnologia *Campus*. A atualização deste tipo de dependência requer um trabalho cuidadoso de reengenharia e testes extensivos para garantir que todas as funcionalidades da plataforma são mantidas após a migração.

A **segunda** opção, à semelhança do que foi descrito no ponto anterior, seria focada no *Front-End*. Esta vertente seria complementar e de igual forma necessária para que, no futuro, se pudesse migrar a infraestrutura para *RedHat*, tal como foi abordado no ponto 5.2.1. Esta abordagem focada na camada de visualização do *Front-End* seria muito semelhante à abordagem da Opção 1, partilhando dos mesmos pontos fortes e possíveis funcionalidades, mas com o acréscimo de todos os benefícios que os *containers* de *Docker* nos trariam. Isto deve-se ao facto do próprio *GitLab* ser compatível com *Docker*, podendo os *runners* das *pipelines* serem executados dentro de *containers.*

A adoção de uma *pipeline* de *GitLab* com um *runner* configurado para *Docker* traz benefícios consideráveis em relação a um ambiente de execução tradicional. Primeiramente, destaca-se o isolamento e a consistência que os *containers Docker* oferecem. Cada tarefa é realizada num ambiente isolado, o que elimina a possibilidade de conflitos entre etapas devido a diferentes configurações ou dependências. Isso traduz-se num controlo mais apurado dos processos e na

garantia de que as execuções são consistentes, independentemente do local. A configuração e manutenção simplificadas são outra vantagem significativa. Utilizando imagens de *Docker*, é possível evitar a instalação manual e a configuração de dependências para cada projeto, o que poupa tempo e reduz o potencial para erros. Este aspeto é ainda mais valioso quando consideramos a escalabilidade que o *Docker* oferece. A capacidade de lançar ou desativar *runners* com rapidez em resposta às necessidades é um fator chave para otimizar recursos e eficiência operacional. Para além disso, o uso eficiente de recursos é uma característica intrínseca ao *Docker*. A capacidade de executar múltiplos containers simultaneamente numa única máquina sem a sobrecarga de sistemas operativos completos economiza recursos e poder de processamento aos servidores. Não podemos esquecer ainda a rapidez com que os *containers* podem ser iniciados e parados, o que acelera o processo de desenvolvimento.

Tal como são partilhados os pontos fortes das *pipelines* do *GitLab*, também é partilhada a previsível dificuldade em fazer a configuração desta solução nos nossos servidores de desenvolvimento ou de *staging*, graças à necessidade de especificar todo o tráfego que sai desses servidores através do *proxy* de gestão dos ambientes.

**Resumindo,** o plano de implementação de *Docker* nos nossos projetos, pode ter duas fases distintas:

**Fase 1 a):** Implementação da primeira opção focada na infraestrutura do *Back-End*.

**Fase 1 b):** Implementação da segunda opção focada no repositório do *Front-End*, com recurso às pipelines do *GitLab*.

(A primeira fase é dividida nas duas soluções, visto que estas podem ser desenvolvidas em qualquer ordem porque são complementares e ambas necessárias para que se possa eventualmente avançar para a segunda fase)

**Fase 2:** Migração da infraestrutura para *RedHat*. Com o auxílio dos nossos parceiros da *DIT*, esta fase tem como objetivo transferir a solução já testada

rigorosamente para um ambiente de produção, garantindo assim uma maior estabilidade e a disponibilidade do serviço aos utilizadores finais.

#### 5.2.2.2. Solução implementada e constrangimentos encontrados

Após uma análise cuidada das possibilidades apresentadas, optou-se pela implementação de *Docker*, particularmente na segunda opção, correspondente ao *Front-End (CI/CD)*. Apesar de não existir um conhecimento prévio de *Docker* por parte dos membros da equipa atual, o maior obstáculo que se previu existir era mesmo os limites de conexão dos ambientes de desenvolvimento e *staging* através do *proxy* de gestão. Sendo assim, este obstáculo é comum à solução do *Front-End* com e sem *Docker*, pelo que nesta solução escolhida com *Docker* temos as regalias adicionais relacionadas com essa mesma tecnologia. Para além desta razão, na escolha da solução pesaram ainda fatores decisivos como a futura integração com o *Back-End* e migração para *RedHat*, o tempo de execução disponível e o impacto diário nos membros da equipa e na sua experiência de desenvolvimento.

Numa fase inicial do desenvolvimento foi efetuada uma prova de conceito, tendo sido desenvolvida uma pipeline com recurso a *Docker* com as funcionalidades estabelecidas na Opção 1. Dado que a experiência prévia na tecnologia era bastante limitada, este exercício foi executado numa abordagem exploratória, para que fossem interiorizados os conceitos básicos de *Docker*, mas também para testar as funcionalidades possíveis numa *pipeline* do *GitLab*. Estes objetivos foram alcançados.

Para iniciar o desenvolvimento, é necessário compreender que o objetivo prático é, após a realização de um *push* para uma *branch* específica, um *runner*, localizado num dos nossos servidores, iniciar uma *pipeline*. Esta *pipeline* é responsável por criar um container de *Docker* onde o projeto *React* será executado. Para isso, primeiramente é necessário realizar a instalação desse *runner* nos nossos servidores. Começou-se por fazer essa tentativa de instalação no ambiente de desenvolvimento. No entanto, tiveram de ser ultrapassados dois problemas. O primeiro estava relacionado com o facto de não termos conexão à internet e, por

essa razão, a instalação convencional do *runner* não seria uma opção. Para contornar isto, teve de se descarregar um ficheiro de instalação específico para o sistema operativo do nosso servidor, que neste caso era o *Debian 6*. No entanto, esta versão de *Debian* já não era suportada pelo *GitLab* – o segundo problema. Por isso, graças aos *upgrades* mencionados anteriormente no ambiente de *staging,* esses servidores já contavam com a versão 11 do mesmo sistema operativo. Assim, tal como pode ser verificado na Figura 7, foi decidido usar apenas o ambiente de *staging* como base para a execução dos *runners* e respetivas *pipelines*. O mesmo problema foi detetado e resolvido para a instalação do *Docker* nos servidores.

Este desenlace implica que, caso o ambiente de destino seja o de desenvolvimento, a informação da base de dados terá de ser consultada num IP correspondente ao servidor desse ambiente[5], e o resultado final da pipeline terá de ser enviado por *FTP (File Transfer Protocol) para o servidor cmp2-dev-fe01* (servidor responsável por alojar e servir a camada de visualização do *Front-End* no ambiente de desenvolvimento).

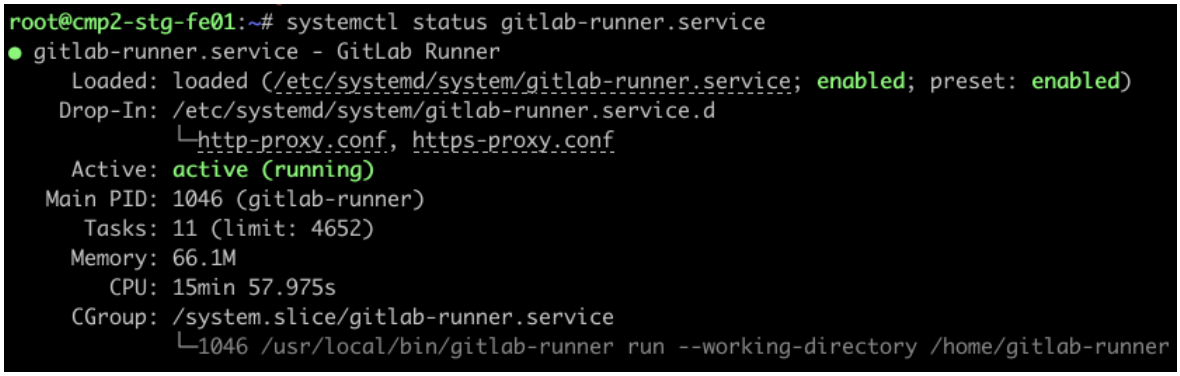



*Figura 7 – Exemplo de um GitLab Runner instalado no servidor cmp2-stg-fe01*

Tendo sido alcançado este ponto de partida, era necessário configurar a *pipeline*. Para isso, as instruções de seguida expostas estão localizadas no ficheiro *gitlab-ci.yml*.

[5] A construção do *bundle* de um tenant implica a consulta de informação de configuração geral do respetivo tenant, que se encontra na base de dados.

Como se pode observar na Figura 8, começando pela escolha da imagem *docker:latest*, esta opção assegura que os trabalhos na pipeline estão sempre a operar com a versão mais recente do *Docker*. Esta escolha é fundamental para garantir que está a usufruir das funcionalidades mais atuais, assim como das melhorias mais recentes em termos de segurança.

As variáveis de ambiente como **HTTP_PROXY**, **HTTPS_PROXY** e **NO_PROXY** são essenciais ao funcionamento da *pipeline* porque sem elas, qualquer tentativa de ligação à internet, seria barrada e iria falhar. Isto é essencial em ambientes corporativos que exigem definições específicas de proxy para acesso à Internet. A variável **REACT_APP_BD_PASS** guarda a senha da base de dados, necessária para a recolha das variáveis de ambiente durante o processo de *build* do projeto de *React*, para que o mesmo tenha acesso às variáveis de ambiente específicas de cada *tenant*.

O uso do serviço ***docker:dind*** (*Docker* dentro de *Docker*) permite que as imagens *Docker* sejam construídas dentro da própria pipeline. Isto é particularmente útil e necessário para criar um ambiente isolado e consistente para a construção de imagens, garantindo que os processos de construção sejam replicáveis e independentes do ambiente do *host*.

A definição das fases ***build*** e ***deploy*** na pipeline oferece uma separação clara e lógica entre as tarefas de construção e *deploy* nos servidores. Isto permite uma melhor organização e gestão do fluxo de trabalho, assegurando que as etapas sejam executadas de uma forma ordenada.

```yaml
image: docker:latest

variables:
  HTTP_PROXY: "http://**.***.*.**:****"
  HTTPS_PROXY: "https://**.***.*.**:****"
  REACT_APP_BD_PASS: $REACT_APP_BD_PASS
  NO_PROXY: "gitlab.****.****.pt"

services:
  - docker:dind

stages:
  - build
  - deploy
```

*Figura 8 - Imagem, variáveis, serviços e fases do ficheiro gitlab-ci.yml*

A configuração de ***cache***, tal como mostra a Figura 9, permite a reutilização da pasta ***node_modules*** entre os *jobs* e é uma estratégia para economizar tempo e poder de processamento. Ao armazenar em cache dependências que raramente mudam, reduz-se significativamente o tempo de construção, uma vez que não é necessário reinstalar essas dependências a cada execução da *pipeline*.

```yaml
.cache: &cache
  key: "$CI_COMMIT_REF_NAME"
  paths:
    - node_modules/

before_script:
  - mkdir -p $HOME/.docker/

.buildDevAndDeploy: &buildDevAndDeploy
  stage: build
  when: manual
  allow_failure: true
  script:
    - docker compose build --build-arg HTTP_PROXY=$HTTP_PROXY --build-arg
    HTTPS_PROXY="$HTTPS_PROXY --build-arg REACT_APP_BD_PASS=$REACT_APP_BD_PASS

  cache:
    <<: *cache
    policy: pull
```

*Figura 9 – Cache e execução da pipeline do ficheiro gitlab-ci.yml*

O *script* de configuração inicial (***before_script***) que prepara o ambiente ***docker*** e configura os *proxies*, garante que cada *job* comece com as definições corretas, otimizando a consistência e a eficiência de todo o processo.

Finalmente, na Figura 10, os *jobs* específicos de *deploy* para diferentes plataformas (**Campus**, **miOne**, **ColLab**, **GPS**) concedem adaptabilidade à *pipeline*. Ao definir estas variáveis e ao reutilizar a configuração de construção e *deploy*, a pipeline torna-se versátil, permitindo *deploys* personalizados para cada *tenant*. Para além disso, uma vez que a fase de *build* está definida como manual, só irá ser iniciada por ordem de alguém na própria página do *GitLab*, como mostra a Figura 11 . Esta solução poderá sofrer ainda mais desenvolvimento para que ser executado automaticamente em certas *branches* ou outro tipo de *triggers.*


*Figura 10 - Triggers manuais para cada tenant no ficheiro gitlab-ci.yml*

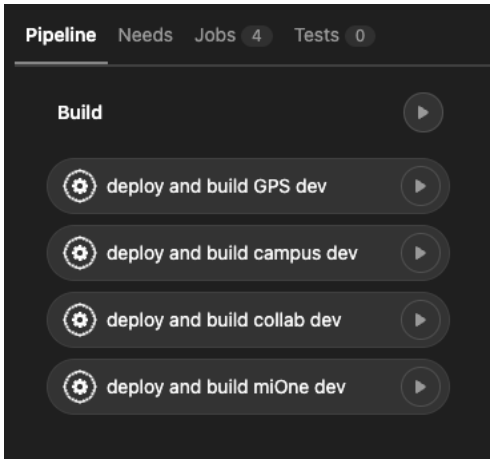
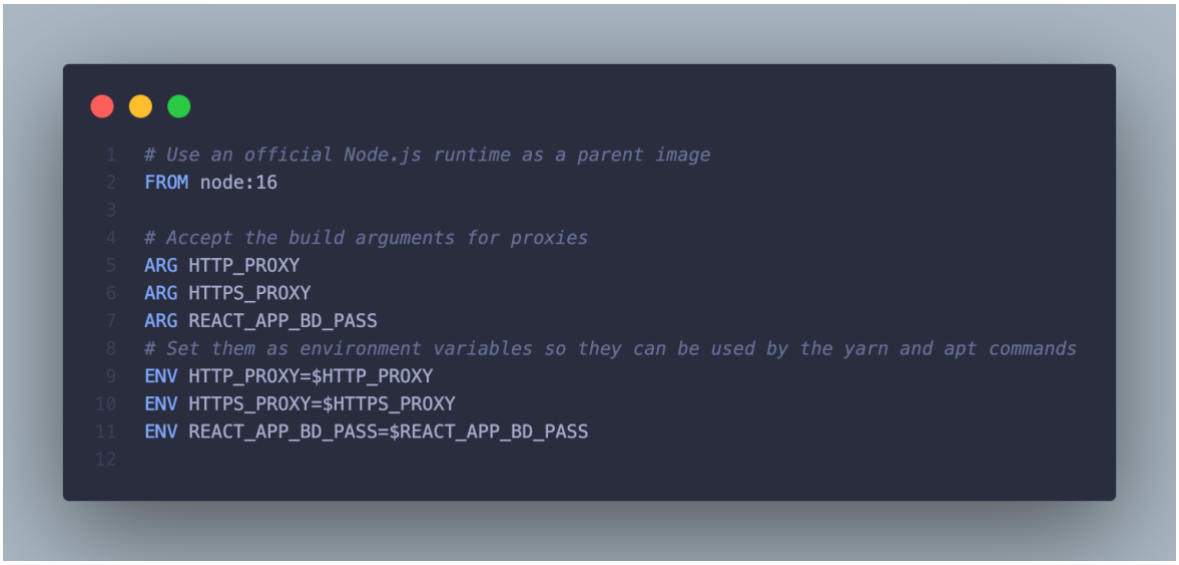



*Figura 11 - Resultado do script na interface do GitLab*

A integração do ***Docker*** na pipeline de *GitLab* cria um elo poderoso entre a automação do processo de integração e entrega contínua (*CI/CD*) e a construção de um *container Docker* personalizado e otimizado para a aplicação. Na *pipeline*, cada *commit* ou mudança no repositório dispara automaticamente a execução de *jobs* que utilizam o *Dockerfile* para construir uma imagem *Docker*. Esta abordagem garante que, a cada alteração, a aplicação é reconstruída num ambiente consistente, refletindo as definições e as dependências mais recentes especificadas no *Dockerfile*. Assim, a *pipeline* não só automatiza o processo de teste e construção do código, mas também garante que a imagem *Docker* da aplicação esteja sempre atualizada e pronta para um *deploy* para qualquer um dos ambientes. Esta sinergia entre a *pipeline* de *CI/CD* e a construção de imagens *Docker* assegura um desenvolvimento ágil, eficiente e alinhado com as melhores práticas de *DevOps*, minimizando os riscos do processo de *deploy* e otimizando o tempo de lançamento de novas versões da aplicação.

Começando pela base, tal como mostra a Figura 12, o *Dockerfile* usa a imagem oficial do *Node.js* versão 16, proporcionando um ambiente estável e atualizado para aplicações *Node.js*. Este comando garante a utilização da versão correta, a mesma que é utilizada pelo repositório da camada visual do *Front-End*. Assim, é ultrapassada uma dificuldade que atualmente existe nos nossos servidores do ambiente de desenvolvimento com o sistema operativo *Debian 6*, que

não é compatível com esta versão de *Node.js*, tornando até agora impossível a execução do código fonte do repositório diretamente nesse ambiente.



```
# Use an official Node.js runtime as a parent image
FROM node:16

# Accept the build arguments for proxies
ARG HTTP_PROXY
ARG HTTPS_PROXY
ARG REACT_APP_BD_PASS
# Set them as environment variables so they can be used by the yarn and apt commands
ENV HTTP_PROXY=$HTTP_PROXY
ENV HTTPS_PROXY=$HTTPS_PROXY
ENV REACT_APP_BD_PASS=$REACT_APP_BD_PASS

```

*Figura 12 - Configurações iniciais do Dockerfile*

A incorporação de argumentos de construção como HTTP_PROXY, HTTPS_PROXY e REACT_APP_BD_PASS, que provêm da sua mesma instanciação no ficheiro *gitlab-ci.yml*, resultou de vários problemas durante o desenvolvimento desta solução. Estas variáveis garantem agora que operações que dependem de rede, como instalação de dependências e acesso a recursos externos, funcionem sem problemas no ambiente de *Docker* que está a ser criado.

O *Dockerfile*, na Figura 13, estabelece ***/app*** como o diretório de trabalho. Este passo simples, mas crucial, assegura que todas as operações subsequentes sejam executadas no contexto correto.

Além disso, o *Dockerfile* executa a cópia do ficheiro ***package.json*** e, quando disponível, do ***yarn.lock***, para dentro do container. Esta etapa é inteligente porque aproveita a *cache* de camadas do *Docker*. Se o ***package.json*** não mudar entre as *builds*, o *Docker* usará camadas em *cache* para acelerar o processo de construção, não sendo necessário reinstalar dependências que não mudaram.

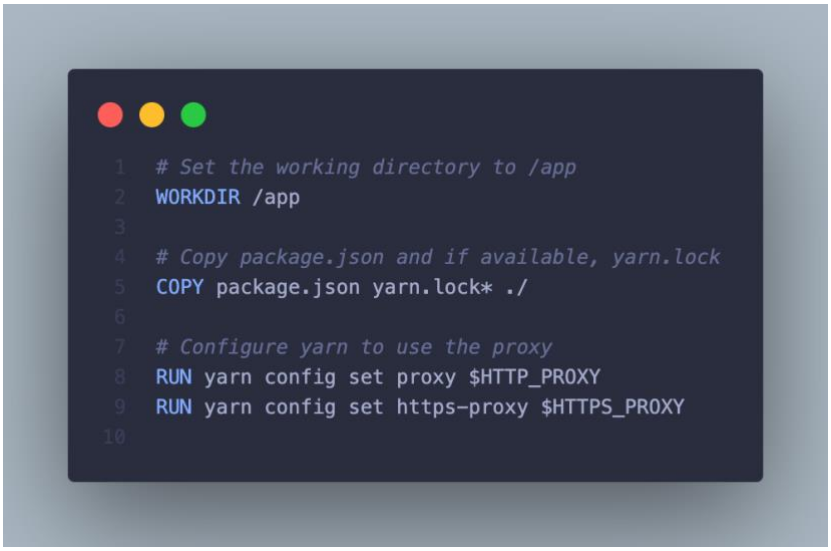



*Figura 13 - Base da pasta de trabalho, cache e configuração do Yarn no Dockerfile*

A configuração do ***proxy*** do *Yarn* dentro do container é outro aspeto cuidadosamente pensado, e mais uma vez resultante de erros com que nos deparamos durante o desenvolvimento. Ao ajustar o *Yarn* para utilizar os proxies configurados, o *Dockerfile* garante que todas as dependências são descarregadas corretamente, mesmo em ambientes com restrições de acesso à Internet, como é o caso dos ambientes de desenvolvimento e *staging*.


*Figura 14 - Instalação de pacotes no Dockerfile*

O processo de instalação de **pacotes globais**, incluindo o *Git*, é fundamental para o processo de *build*, especialmente para aplicações que dependem de ferramentas externas ou módulos do *Node.js*, como é o caso específico do nosso repositório. Esta instalação assegura que todas as ferramentas necessárias estão

disponíveis para a próxima etapa, caso contrário não seria possível avançar com o processo de *build* do projeto.

O comando ***yarn install*** no *Dockerfile* é vital para garantir que todas as dependências necessárias para a aplicação estejam instaladas. Este passo não só prepara o ambiente de execução da aplicação, mas também assegura que a construção da aplicação pode ser realizada com todas as suas dependências corretamente instaladas. Apesar de parecer trivial, este foi o passo que mais tempo consumiu em todo este processo de desenvolvimento. Este facto explica-se, mais uma vez, com as limitações de acesso à rede, impostas nos ambientes de desenvolvimento e *staging*. Apesar de termos um *proxy* de gestão destes mesmos ambientes gerido pelos nossos parceiros, é necessário fornecer-lhes uma lista detalhada de todos os domínios a que esses ambientes precisam de aceder. Ora, uma vez que este processo de instalação requer a ligação a inúmeros domínios distintos para cada pacote que é especificado no ficheiro *package.json,* foi levado a cabo um trabalho demorado de tentativa e erro para que o processo de instalação fosse bem sucedido. Para cada tentativa de instalação falhada, no erro apresentado pelo *container*, era mostrado o domínio a que o pacote necessitava de aceder para ser instalado. Por sua vez, para cada um destes erros, foi necessário contactar os nossos parceiros para adicionarem esse domínio ao nosso *proxy* de gestão, e voltar a tentar. A elevada burocracia, que é exigida para este tipo de pedido junto dos nossos parceiros, fez com que cada um destes erros demorasse cerca de um a três dias a resolver, o que no final resultou num atraso bastante significativo do processo de desenvolvimento desta etapa.


*Figura 15 - Processo de build da plataforma no Dockerfile*

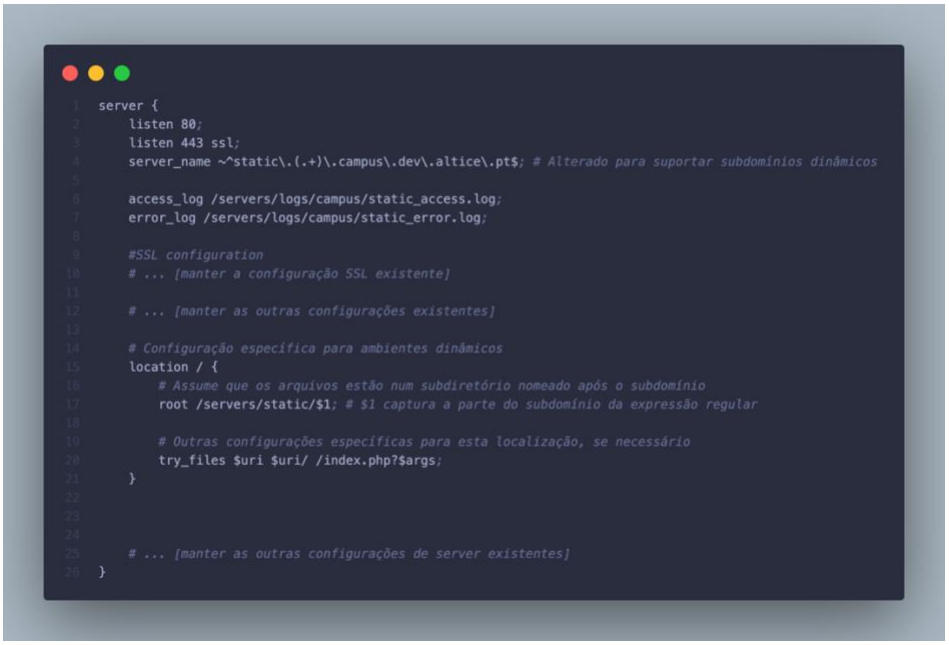
Este constrangimento foi eventualmente ultrapassado e, finalmente, tal como é exemplificado na Figura 15, o *Dockerfile* conclui com a cópia do restante do código-fonte para o *container* e a execução de um comando ***gulp build*** para construir a aplicação, comando este próprio do nosso repositório e amplamente usado diariamente no decorrer do desenvolvimento das plataformas. Este comando especifica o ambiente de desenvolvimento e a plataforma alvo. Este passo final compila e prepara a aplicação para execução ou *deploy*, garantindo que a versão construída reflete as mais recentes alterações no código-fonte. Dito isto, foi nesta etapa que nos deparámos com um problema de ligação à nossa própria base de dados, que impede este processo de *build* de terminar com sucesso. As razões para este problema de ligação ainda não foram compreendidas, mas tudo aponta para que seja uma limitação nas configurações de rede entre o *container* de Docker e o servidor onde o *runner* está alojado. Esta ligação é necessária para que o *tenant* consiga pedir as suas variáveis de ambiente à base de dados e com elas, determinar as funcionalidades que vão ou não ser compiladas nessa versão.

#### 5.2.2.3. Trabalho futuro

À data de escrita, o problema mencionado no ponto anterior ainda não foi ultrapassado, o que impediu o avanço para os próximos passos. Estes, supondo que a *build* seria concluída sem falhas, seriam a migração da lógica do servidor *Web (NGINX)* atualmente a atuar diretamente no servidor de Front-End do

ambiente de *staging*, para dentro do *container* de *Docker* e, de seguida, a disponibilização da versão da plataforma nesse *container,* com uma configuração que permitisse o acesso direto a partir de qualquer browser, desde que o utilizador estivesse ligado através da VPN corporativa que toda a nossa equipa utiliza no seu dia a dia.

Depois disso, a principal funcionalidade que seria desejável executar é a que diz respeito ao lançamento de versões em ambientes estáticos e dinâmicos. Tal como explicado previamente, isto significaria que seria possível fornecer uma pré-visualização automática das alterações efetuadas numa *feature branch*, criando um ambiente dinâmico automaticamente, assim que essa *branch* fosse publicada. Para isso, seria necessário modificar as configurações do próprio *NGINX.* Fica na Figura 16 uma proposta para essa configuração do ficheiro *static.conf,* que faz referência à camada de visualização do *Front-End* que temos desenvolvido ao longo deste capítulo. Esta proposta tem alguma sustentação, visto que é baseada na solução desenvolvida na prova de conceito, feita na fase inicial deste processo de implementação.



```
server {
    listen 80;
    listen 443 ssl;
    server_name ~^static\.(.+)\.campus\.dev\.altice\.pt$; # Alterado para suportar subdomínios dinâmicos

    access_log /servers/logs/campus/static_access.log;
    error_log /servers/logs/campus/static_error.log;

    #SSL configuration
    # ... [manter a configuração SSL existente]

    # ... [manter as outras configurações existentes]

    # Configuração especifica para ambientes dinâmicos
    location / {
        # Assume que os arquivos estão num subdiretório nomeado após o subdomínio
        root /servers/static/$1; # $1 captura a parte do subdomínio da expressão regular

        # Outras configurações especificas para esta localização, se necessário
        try_files $uri $uri/ /index.php?$args;
    }


    # ... [manter as outras configurações de server existentes]
}
```

*Figura 16 - Previsão de uma configuração para o servidor de NGINX com suporte a ambientes dinâmicos*

# 6. Avaliação das soluções implementadas

## 6.1. Resultados do questionário

Conforme descrito no capítulo dos Instrumentos de recolha de dados, a última fase da investigação debruça-se sobre a avaliação das mudanças implementadas na metodologia de trabalho de desenvolvimento na equipa do projeto *Campus*, através de um questionário. Como tal, neste capítulo é feita a análise às respostas a esse questionário dadas pelos próprios membros da equipa entre os dias 2 e 6 de outubro de 2023.

### 6.1.1. Perfil dos inquiridos

Assim como já foi referido previamente, a equipa do *Campus* é bastante penalizada com a frequente rotatividade dos seus membros, visto que os seus membros usufruem de uma bolsa de investigação e têm de obedecer a limites de tempo máximos por cada membro, dependendo do seu grau de estudos. Em setembro de 2023, antecipando a saída dos sete membros até então integrantes da equipa, foram contratados mais quatro estudantes/bolseiros para darem continuidade aos projetos em curso. Dos sete membros seniores na equipa, seis são da área das Ciências e Tecnologias da Comunicação e, à data desta investigação, estão todos a terminar o seu percurso académico com a entrega da dissertação no âmbito do Mestrado em Comunicação e Tecnologias Web da Universidade de Aveiro. A exceção é apenas um membro que é da área da Educação, tendo concluído o doutoramento há poucos meses.

Os novos membros foram e, à data da escrita deste documento, continuam a ser alvo de um processo de formação e introdução às ferramentas, metodologias e projetos da equipa. Por ainda não terem um conhecimento tão profundo sobre a equipa e por não terem acompanhado o processo desde o início, foi ponderada a aplicação deste questionário aos mesmos. No entanto, visto que estão a ser alvo de um processo de aprendizagem rápido e com bastante proximidade aos restantes membros, foi acordado que a sua opinião relacionada com algumas questões poderia ser valiosa e um contributo positivo para a análise e resultados desta

investigação. Por isso, foi decidido elaborar um segundo conjunto de questões, diferentes daquelas aplicadas aos outros membros, adaptadas à sua situação de recém-chegados, que podia oferecer uma visão diferente dos restantes.

Assim, como pode ser observado no Gráfico 1, numa amostra de 10 pessoas elegíveis, foram inquiridos 6 membros que acompanharam o processo desta investigação desde o início (membros seniores) e 4 membros novos. A faixa etária da amostra encontra-se entre os 21 e os 24 anos, sendo que 5 membros são do sexo masculino e 3 do sexo feminino.

De notar que, apesar de os coordenadores também terem acompanhado o processo, à semelhança do que aconteceu com o *Focus Group*, não foram selecionados para a participação neste questionário. Pela mesma razão, os especialistas e antigos membros da equipa que foram alvo das entrevistas, também não entram no leque de participantes do questionário, uma vez que apenas os membros da equipa atual têm contacto diário com o objeto de estudo desta investigação e têm a sua experiência no processo como alvo deste questionário. Para além disso, os coordenadores da equipa e dos projetos coincidem com os orientadores desta mesma investigação, pelo que não seria adequado aplicar-lhes o questionário.

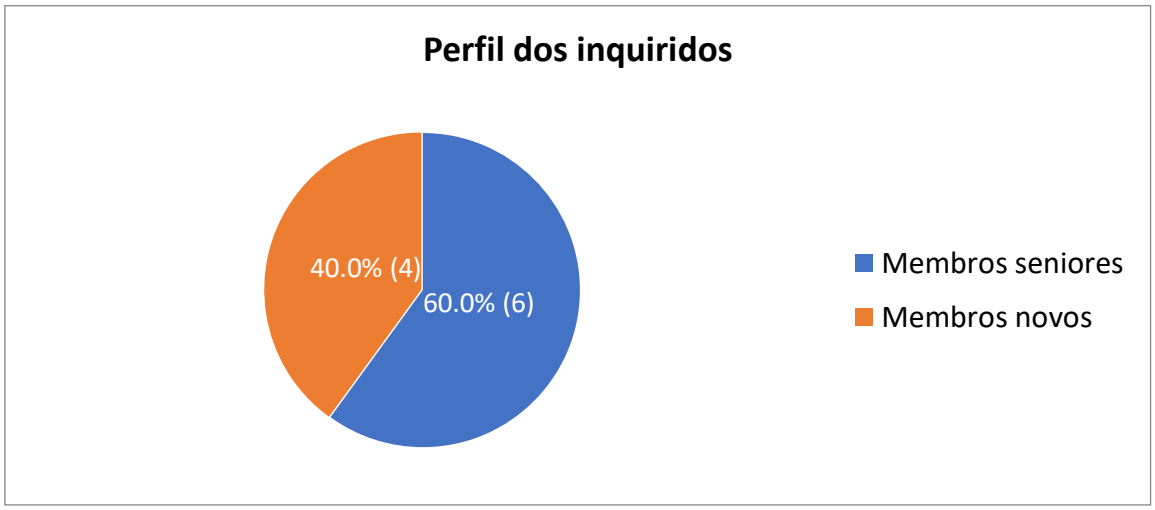



*Gráfico 1 - Perfil dos inquiridos no questionário*

### 6.1.2. Análise do questionário aos membros seniores

#### 6.1.2.1. Produtividade da equipa

Ao analisar as respostas à pergunta sobre a perceção dos inquiridos quanto à variação da sua produtividade individual e de equipa, como podemos observar no Gráfico 2, a maior parte (83,3%) indicou que sentiu um aumento na produtividade após a implementação da nova metodologia, seja esse aumento mais ou menos considerável. Nenhum inquirido considerou que a produtividade tenha diminuído pouco nem que tenha diminuído consideravelmente.

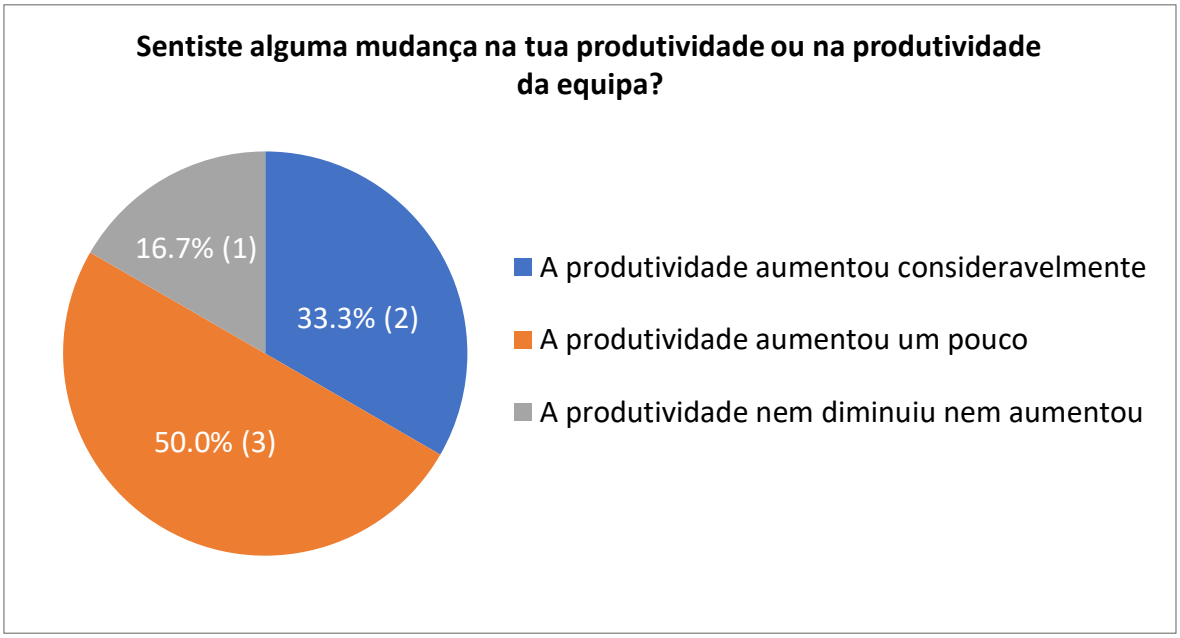



*Gráfico 2 - Perceção dos membros quanto à variação da produtividade*

Os comentários deixados no seguimento desta questão fornecem-nos informação valiosa sobre as mudanças na produtividade sentidas pelos inquiridos. Sumariando esses comentários, em áreas de atuação distintas:

1. **Definição clara de tarefas**: A utilização extensiva das funcionalidades do *Trello* e da mais detalhada definição das tarefas na plataforma com recurso a imagens e *checklists* foi apontada como uma razão para o aumento na produtividade.

2. **Comunicação**: A comunicação constante e eficaz entre os membros da equipa foi frequentemente mencionada como um fator que contribuiu para um melhor desempenho.

3. **Motivação através de prazos**: A definição de prazos específicos para os *Sprints* foi vista como uma motivação adicional para completar as tarefas atempadamente.

4. **Identificação e solução de problemas**: A nova metodologia permitiu uma melhor perceção dos problemas enfrentados pelos membros da equipa, facilitando a identificação e resolução coletiva destes.

5. **Variações na Produtividade**: Algumas respostas mencionam que, apesar de geralmente sentirem uma melhoria na produtividade, esta ainda varia devido a fatores externos, como por exemplo épocas de entrega de trabalhos curriculares.

6. **Relações da Equipa**: Foi destacada uma melhoria nas relações interpessoais e na coesão da equipa como um todo. Foi também mencionado que o sentimento de pertença ao grupo e ao projeto em curso foi um fator importante para essa melhoria.

7. **Estruturação do Dia**: O recurso às reuniões diárias de *stand-up*, e o seu aprimorar para que fossem mais rápidas e focadas no essencial, parece ajudar na organização diária das tarefas, o que pode ter contribuído para uma melhor eficiência.

8. **Gestão de Qualidade**: A nova abordagem na gestão e manutenção do código com a separação em *branches* específicas e um maior controlo de qualidade por mais intervenientes, permite uma melhor monitorização da qualidade e produtividade da equipa.

Dito isto, existem respostas que sugerem uma dificuldade em quantificar a produtividade individual e de equipa ou que a mudança na produtividade não foi

muito significativa, mas admitindo que outros aspetos como a comunicação e organização melhoraram.

#### 6.1.2.2. Avaliação geral da execução dos *Sprints*

Como se pode observar no Gráfico 3, no que toca à avaliação da execução dos *Sprints* semanais, a maioria (83,3%) dos membros da equipa considera o processo de execução dos *Sprints* como "adequado". Apenas uma resposta é "neutra". Nenhum membro avaliou a execução dos *Sprints* como nada adequado, pouco adequado nem muito adequado. Estes dados sugerem que o método utilizado é, em geral, eficaz, mas deixa espaço para possíveis melhorias ou ajustes.

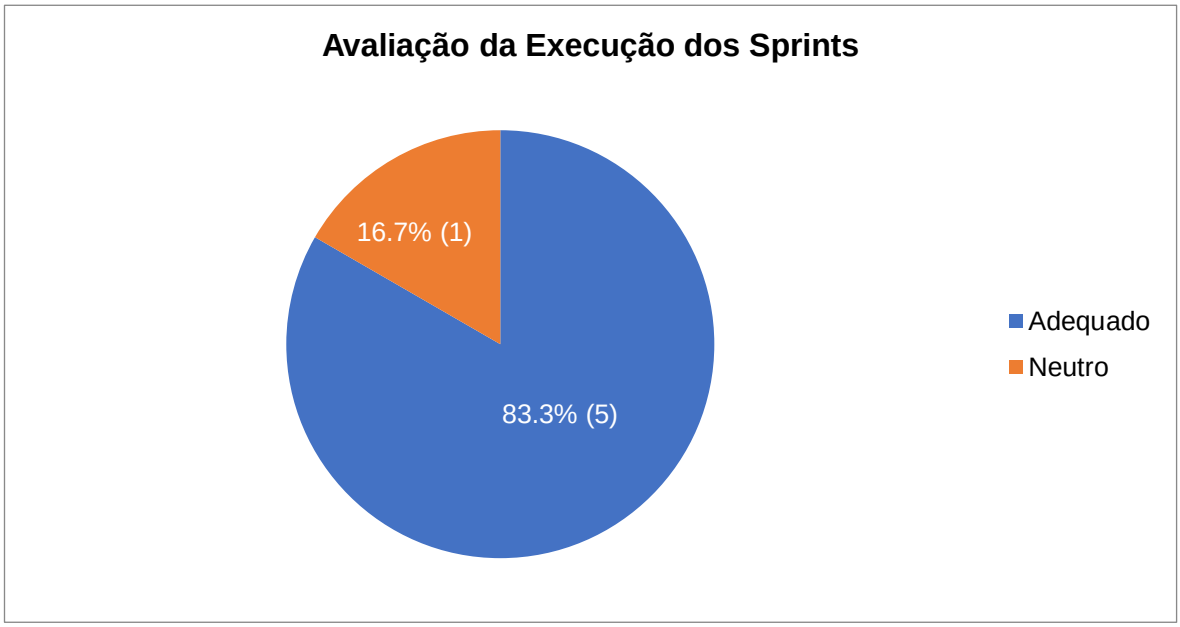



*Gráfico 3 - Avaliação dos inquiridos à execução dos Sprints*

Essas melhorias são imediatamente enunciadas nos comentários, justificando as respostas dadas pelos inquiridos. Resumindo, aqueles que acham a execução adequada realçam a clareza das tarefas e a organização em geral como pontos positivos, mas referem algumas falhas nas reuniões de *daily stand-up* e de retrospetiva. É referido que, por vezes, estes momentos são descurados e saltados por várias razões e fatores externos, como a falta de membros presentes no momento de reunir. A razão para isto decorre da complexidade inerente em

acomodar os diversos horários que os membros possuem para assistir às suas aulas.

A resposta do inquirido que avaliou a execução com a opção neutra justificou a sua resposta indo mais além e explicando em detalhe estas mesmas falhas. Explicou dizendo que sentiu que no momento de planeamento do *Sprint* e na definição de tarefas, era muitas vezes ignorado ou mal calculado o tempo estimado que essa tarefas levariam a ser concluídas. Isso fez com que este inquirido sentisse que muitas vezes a carga de trabalho podia ser desajustada entre membros da equipa. Referiu ainda que, no seu ponto de vista, era necessário um maior controlo do trabalho por parte dos coordenadores para aumentar a transparência do trabalho realizado por cada membro. Mencionou ainda que a execução do *Sprint* estava dependente da responsabilidade individual de cada um dos membros da equipa, sem que houvesse nada que a promovesse. No final, fundamentou a sua resposta dizendo que considerava que tinha havido uma melhoria muito significativa no processo de execução dos *Sprints*, no entanto, estas tinham sido desaproveitadas pelos motivos anteriormente expostos.

Em contrapartida, um dos inquiridos realçou especificamente as reuniões prévias à Sprint Review. Apesar de terem sido executadas pontualmente, estas reuniões tinham o objetivo de juntar a equipa de programação para uma rápida verificação pós *merge* para a *branch* WIP. Neste ponto, destacou os benefícios trazidos por este momento para o rigor, transparência, controlo e eficácia do trabalho semanal. Justificou ainda que, na sua opinião, a tendência pontual em haver um dispêndio excessivo de tempo com tarefas inesperadas que acabavam por consumir tempo útil à finalização daquelas planeadas para o *Sprint*, levou este inquirido a não optar pela opção “muito adequado”.

#### 6.1.2.3. Rigidez e controlo sob a execução da metodologia

As respostas a esta pergunta e respetivos comentários são diversas e bastante detalhadas, o que sugere que os membros da equipa têm opiniões fortes e bem fundamentadas sobre o tema. Algumas tendências podem ser observadas no Gráfico 4:

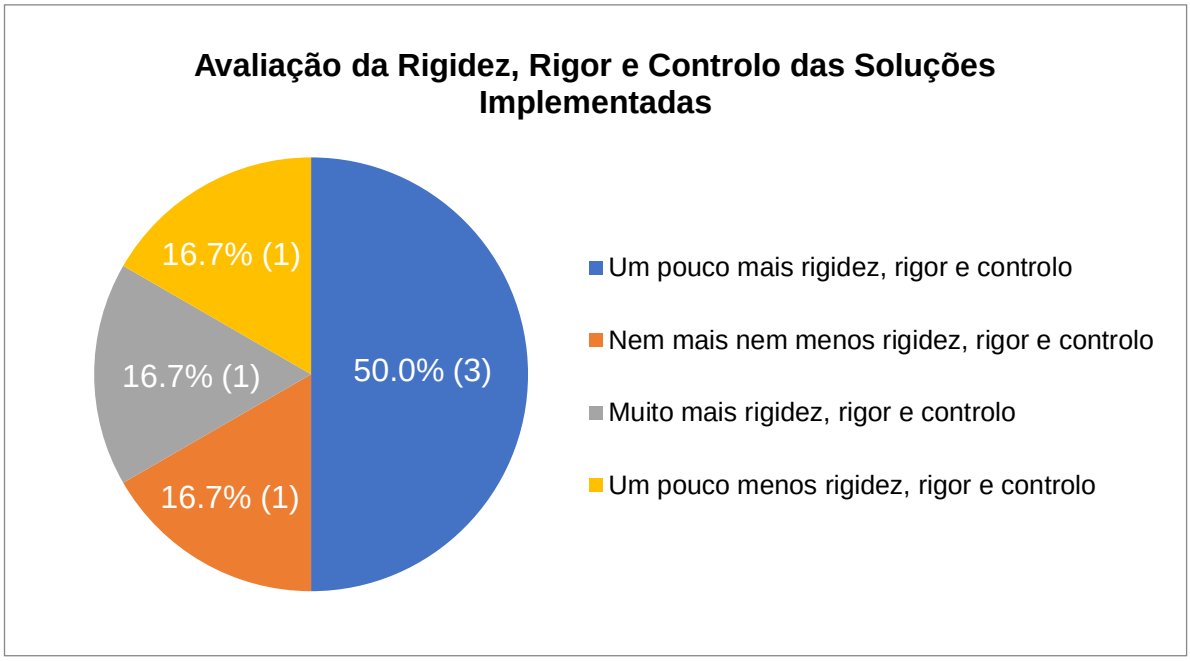



*Gráfico 4 - Avaliação da rigidez, rigor e controlo das soluções implementadas*

A maioria (66,7%) concorda que algum nível de rigidez e controlo é necessário. No entanto, dois inquiridos (33,3%) não se revêm nesta opinião e assumem que o nível de controlo atual é adequado ou deve mesmo ser menor.

Para percebermos os diferentes pontos de vista dos inquiridos, agrupámos os comentários pela avaliação que fazem em cima. Desta forma:

1. **Um pouco mais de rigidez, rigor e controlo**: Os membros que optaram por esta opção concordam que um certo grau de flexibilidade deve ser mantido, especialmente porque os horários da equipa são diversos. Sugerem a implementação de medidas adicionais como reuniões secundárias para os que não podem participar nas primeiras. Mencionam também o autocontrolo, sugerindo que se cada membro for rigoroso consigo mesmo, a produtividade da equipa irá melhorar. Nesta avaliação para mais rigor, dá-se mais ênfase à consistência e à necessidade de estratégias para manter essa consistência, como autoavaliações e liderança rotativa em diversos papeis como o de Scrum Master, por exemplo.

2. **Nem mais nem menos rigidez, rigor e controlo**: A resposta sublinha que a equipa ainda está em fase de desenvolvimento e que as “falhas” são naturais. Há uma aceitação dos erros como uma oportunidade de aprendizagem mútua para todos os membros integrantes da equipa.

3. **Muito mais rigidez, rigor e controlo**: Esta resposta é uma crítica detalhada das práticas atuais. Argumenta que a falta de rigor e controlo em várias áreas, como a revisão do *Sprint* e a autogestão e autonomia de cada um, está a impedir a melhoria contínua da equipa. Pede mais controlo por parte dos coordenadores e menos autonomia de cada membro, mais rigor na execução e um melhor planeamento de tarefas de acordo com o tempo que se estima que as mesmas irão demorar. Apesar disso, são feitas algumas notas positivas quanto à integração da equipa de design no controlo de qualidade e experiência do utilizador; à utilização de *branches* em substituição dos ambientes de *staging* e produção; e também à criação do canal dedicado a *updates* em tempo real no Microsoft Teams para estimular a transparência no trabalho de cada membro.

4. **Um pouco menos de rigidez, rigor e controlo**: Esta resposta sugere que a equipa já tem suficiente rigidez e que mais flexibilidade poderia beneficiar o ambiente de trabalho, abrindo a possibilidade de trabalhar remotamente, acordando que a autogestão e autonomia no trabalho são essenciais.

Em termos gerais, a tendência da equipa parece inclinar-se para um aumento do rigor e controlo, mas com diferentes níveis de intensidade. Enquanto alguns membros valorizam a flexibilidade e a autogestão, outros pedem o oposto, com uma estrutura mais rigorosa e sistemática. O fator comum parece ser um desejo de equilíbrio entre rigor e flexibilidade, adaptado às necessidades e ao perfil da equipa. A sugestão de implementação de estratégias concretas como reuniões secundárias, autoavaliações e liderança rotativa também evidencia uma vontade de tornar esses ideais práticos e mensuráveis.

Esta divergência de opiniões pode revelar uma diferença nas preferências de cada um enquanto profissional, numa fase muito próxima da chegada ao mercado de trabalho. Há pessoas que preferem uma maior autonomia e um

sentimento de mais confiança, outros preferem trabalhar num ambiente mais controlado, sob a tutela de alguém num cargo superior. A diferença de ideias quanto a esta questão vem também mostrar que a equipa, apesar de ter membros da mesma área de estudos em idades similares, é heterogénea na forma como trabalha ou gostaria de trabalhar, fazendo com que o dia a dia no local de trabalho seja um ambiente propício ao debate, troca de ideias e aprendizagem mútua.

#### 6.1.2.4. Satisfação com a metodologia implementada

Como é apresentado no Gráfico 5, podemos inferir que a maioria (83,3%) dos inquiridos está satisfeito ou muito satisfeito com a nova metodologia implementada. Apenas um expressou neutralidade, não se inclinando nem para satisfação nem para insatisfação.

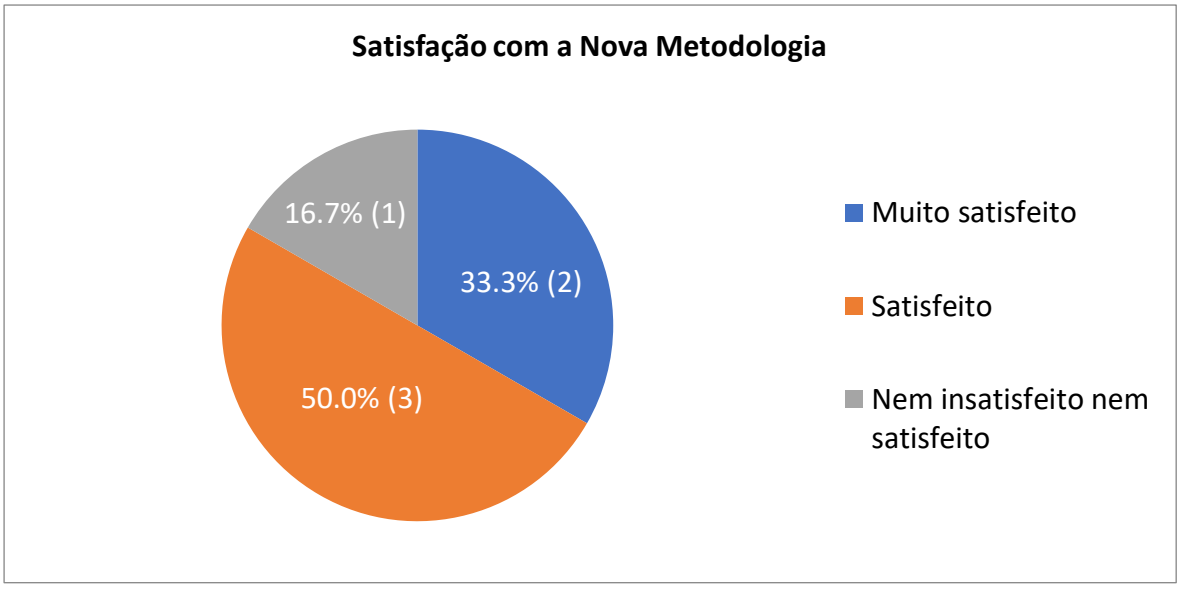



*Gráfico 5 - Satisfação com a nova metodologia*

Analisando os comentários recebidos, são reconhecidas melhorias significativas na metodologia de trabalho, mas acredita-se que ainda existem problemas a serem resolvidos. Destaca-se a eficácia da metodologia, salientando-se também a melhoria na comunicação entre os membros da equipa. Contudo, vê-se a necessidade de tornar algumas práticas, como as reuniões de *stand-up* diárias e as retrospetivas, mais consistentes. A flexibilidade nos horários de trabalho é vista por um participante como uma potencial melhoria. Existe ainda um testemunho de

profunda satisfação com a equipa e a metodologia, salientando a valorização dos contributos individuais e a colaboração aprimorada entre os membros da equipa.

Resumindo, a análise sugere uma receção positiva da nova metodologia ágil entre os membros da equipa. O feedback sobre a metodologia é amplamente positivo, com ênfase na melhoria da comunicação e na colaboração entre os membros da equipa. No entanto, existem sugestões para aperfeiçoamentos, incluindo a adoção de práticas mais consistentes e a introdução de maior flexibilidade nos horários.

### 6.1.3. Análise do questionário aos novos membros

#### 6.1.3.1. Experiência prévia com metodologias ágeis

Observa-se no Gráfico 6 uma divisão equilibrada entre os inquiridos que já tinham e os que não tinham experiência prévia com metodologias ágeis.

Com base nos comentários daqueles que indicaram ter experiência anterior com metodologias ágeis, podemos extrair algumas informações.

Um dos novos membros menciona ter usado Scrum e Kanban. No contexto de trabalho remoto, foi adotada uma abordagem focada em objetivos semanais, dividindo tarefas em listas de "*to do*", "*doing*" e "*done*", o que se assemelha a um *Sprint* semanal. O outro membro com experiência prévia destaca que, durante o terceiro ano da sua licenciatura, foi introduzido a metodologias ágeis como Scrum e começou a usar o JIRA para gestão de tarefas. Desde essa experiência, passou a aplicar estes métodos em projetos de maior envergadura.

Resumindo, os inquiridos que possuíam experiência prévia com metodologias ágeis demonstram algum entendimento destas práticas, variando entre a aplicação em contextos de trabalho remoto e a integração destes métodos em projetos académicos. Estas respostas indicam uma familiaridade com ferramentas e práticas comuns em ambientes ágeis, como o uso de *Sprints*, Kanban e ferramentas de gestão de tarefas como o JIRA.

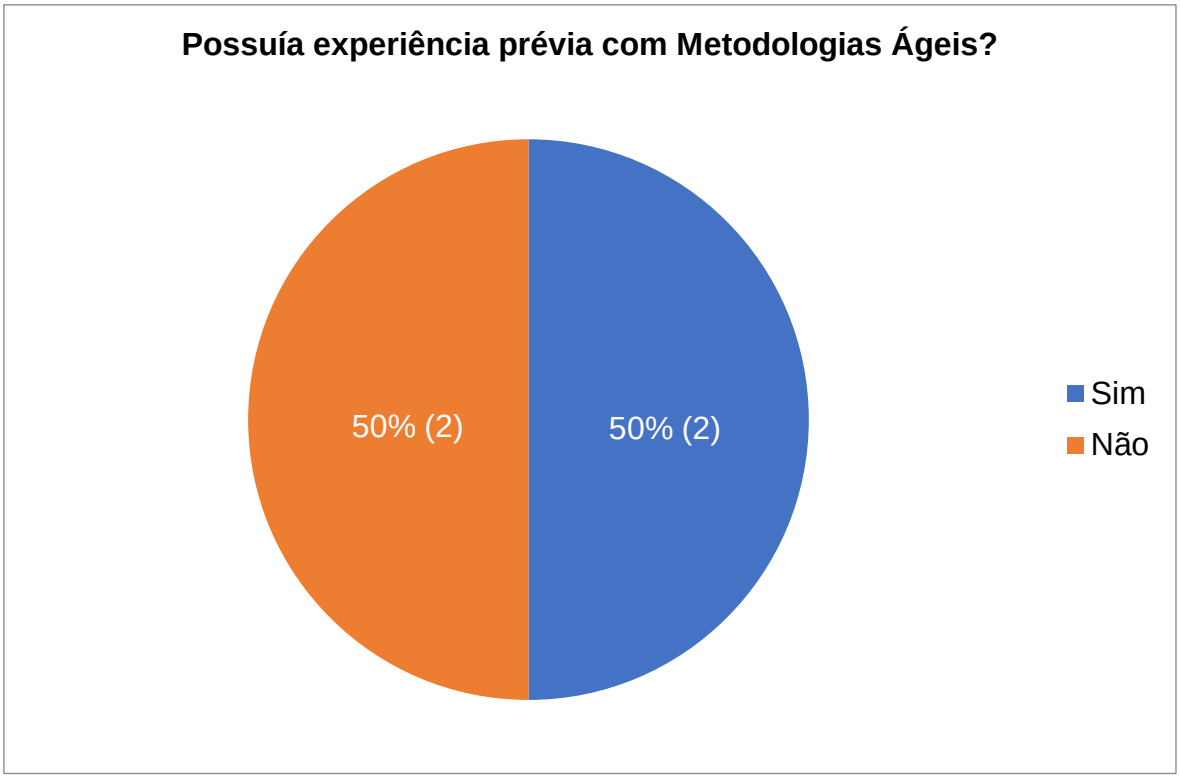



*Gráfico 6 - Experiência prévia em metodologias ágeis dos novos membros da equipa*

### 6.1.3.2. Experiência na transição e perceção da metodologia da equipa

Quanto à perceção de cada um dos novos membros da equipa relativamente à metodologia praticada, como se pode aferir no Gráfico 7, a maioria (75%) dos participantes indica ter um bom ou muito bom conhecimento da metodologia de trabalho da equipa, com duas respostas indicando "Conheço bem a metodologia" e uma afirmando ter "total conhecimento sobre a metodologia". Apenas um membro admitiu ainda não ter grande conhecimento sobre a mesma. Isto pode justificar-se potencialmente devido à posição que essa pessoa pode desempenhar na equipa. Caso esse cargo não interaja tão de perto com a metodologia de trabalho da equipa, como é o caso do que acontece com a equipa de design, essa perceção é justificável uma vez que não participa nas questões relacionadas com o código fonte dos projetos.

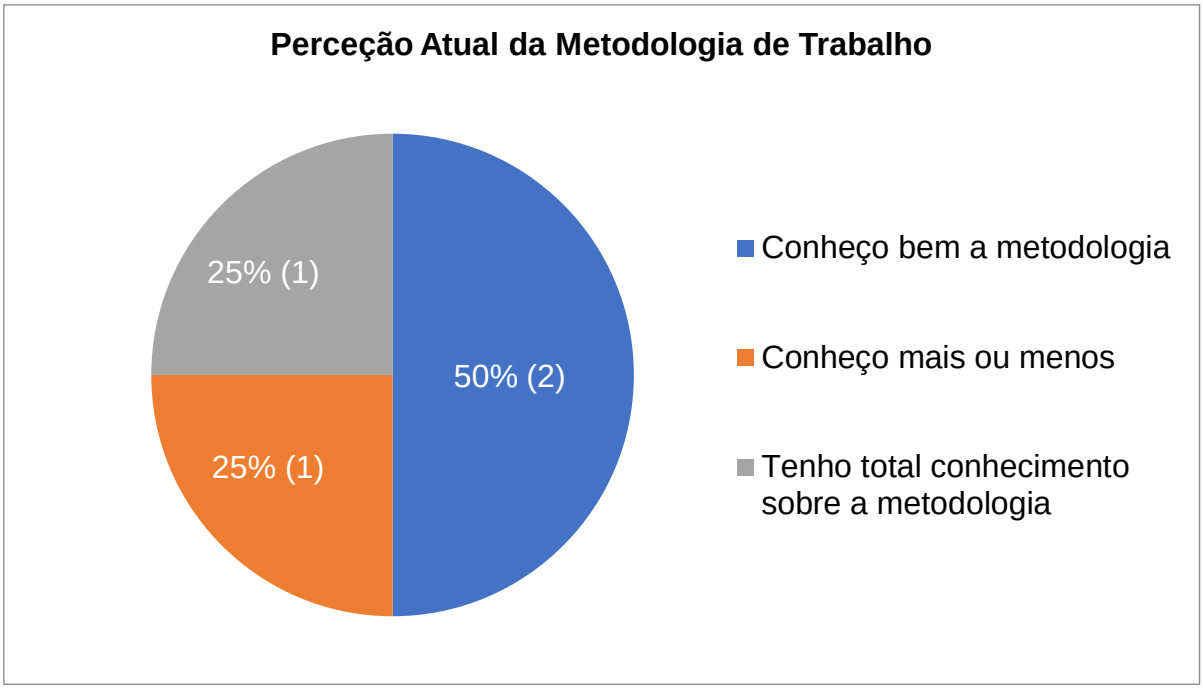



*Gráfico 7 - Perceção dos novos membros sobre a metodologia atual da equipa*

No entanto, e tal como foi referido anteriormente, é necessário relembrar que estes membros, à data das suas respostas ao questionário, faziam parte da equipa há menos de um mês, pelo que é perfeitamente normal que não tenham um vasto conhecimento da metodologia ou outros fatores relacionados com a equipa e o seu funcionamento. Aliás, era até expectável que os mesmos ainda sentissem algum desconforto no que diz respeito a estes temas. Assim, os resultados obtidos podem ser indicativos de uma boa integração por parte dos membros seniores e fruto do esforço prévio em organizar e documentar os vários processos e metodologias relativos aos vários projetos da equipa.

Dito isto, analisando a experiência dos novos membros na transição para a metodologia em vigor na equipa aquando da sua chegada à mesma, o Gráfico 8 mostra que essa experiência é predominantemente positiva, com três participantes a classificarem como "Fácil" e um como "Muito fácil".

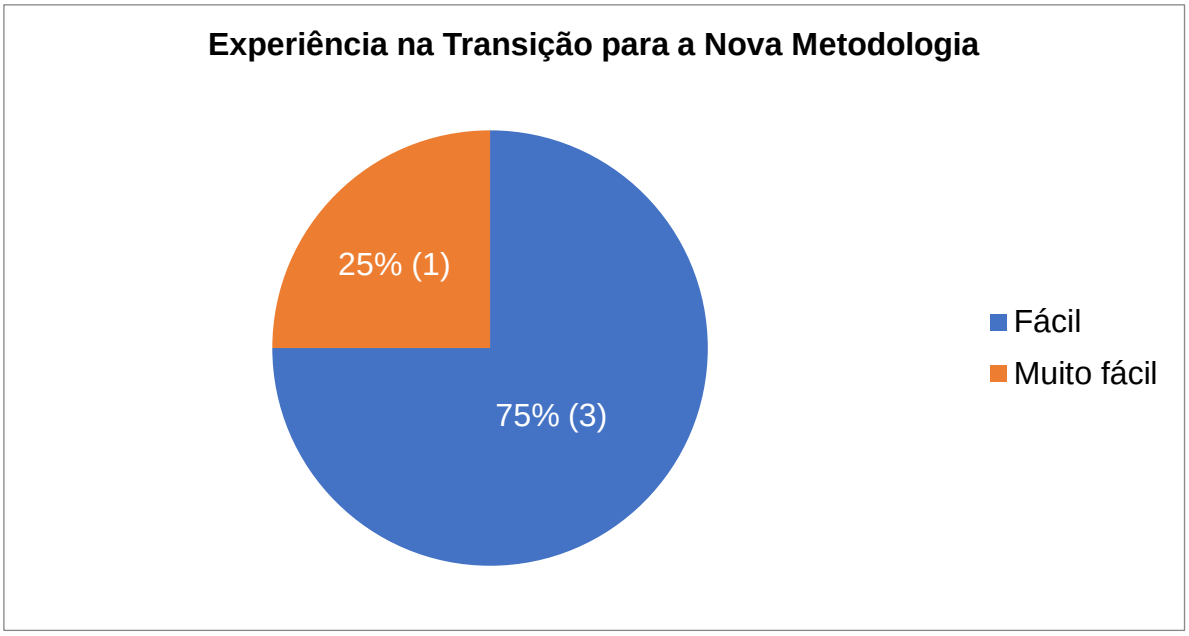



*Gráfico 8 - Experiência dos novos membros na transição para a metodologia da equipa*

Para justificar as respostas dadas, um dos comentários refere que, embora a transição esteja a ocorrer de forma suave e seja facilitada para a integração dos novos elementos, ainda há um caminho de aprendizagem pela frente. É também mencionado que a equipa trabalha de forma colaborativa e integra bem as várias áreas, como o Back-End, Front-End e Design. Foi apontado que a realização diária de uma reunião é um ponto positivo para garantir que todos estejam informados sobre o trabalho uns dos outros. Há ainda uma referência à boa integração e suporte da equipa por parte dos membros seniores, bem como à definição clara dos métodos de trabalho graças à comunicação eficaz que se faz sentir diariamente.

Nenhum dos membros decidiu responder ao último campo de resposta aberta do questionário, que dava aos mesmo a oportunidade de sugerir pontos para melhorar a metodologia de trabalho atual, a colaboração ou comunicação entre os membros da equipa.

Sumariando, os resultados indicam que a maioria dos participantes está bastante familiarizada com a metodologia da equipa e encarou facilmente a

transição para esta metodologia. Os comentários fornecidos sublinham a importância da colaboração e comunicação dentro da equipa, que parecem ser pontos fortes. Também é evidente que, embora a maioria sinta que a transição foi fácil, há um reconhecimento de que o processo de aprendizagem e integração é contínuo. Esta consciência é positiva, pois sugere que os membros da equipa estão abertos a aprender e evoluir continuamente nas suas funções.

# 7. Conclusões e trabalho futuro

Em todo o processo de investigação, é importante mencionar que os resultados alcançados tiveram a preciosa ajuda de todos os membros pertencentes à equipa do Campus by Fundação Altice, principalmente através da vontade que demonstraram em contribuir para esta investigação, correspondendo aos desafios propostos e colaborando com as suas opiniões e ideias nas várias fases da investigação.

Dito isto, para que se possa analisar este estudo, é necessário relembrar os objetivos definidos previamente:

- Pesquisar, identificar e averiguar a aplicabilidade de novas tecnologias que permitam melhorar a infraestrutura que suporta o Campus;
- Desenvolver essa solução, em conjunto com a equipa de I&D;
- Propor um conjunto de práticas, metodologias, ferramentas e técnicas de desenvolvimento para melhorar as vigentes na equipa;

Os resultados obtidos demonstram avanços significativos, embora também revelem áreas que requerem mais desenvolvimento. O primeiro objetivo foi satisfatoriamente alcançado através de múltiplos instrumentos de recolha de dados, incluindo diálogos com especialistas e antigos membros da equipa, bem como a realização de um Focus Group com os membros da equipa e ainda conversas com os nossos parceiros do grupo Altice. Estas abordagens permitiram identificar tecnologias emergentes que têm o potencial de melhorar substancialmente a infraestrutura em questão.

O progresso no segundo objetivo foi significativo, ainda que incompleto. Foram implementadas melhorias consideráveis nos servidores, proporcionando uma base sólida para futuros esforços de desenvolvimento. Este objetivo permanece em andamento e representa uma oportunidade para investigação futura.

Embora tenha sido concluído com sucesso, o terceiro objetivo revelou a existência de margens para aperfeiçoamento. Os resultados dos questionários identificaram áreas específicas onde a metodologia da equipa poderia ser aprimorada.

Em síntese, o estudo alcançou maioritariamente os seus objetivos, contribuindo assim para o avanço do conhecimento e práticas na área de gestão da infraestrutura tecnológica e metodologias de trabalho na nossa equipa. As lacunas identificadas apontam para necessidades de pesquisa futura, reiterando a importância da evolução contínua neste domínio.

Assim, conclui-se que os esforços de investigação foram em grande medida bem-sucedidos, mas também reveladores de novas questões e desafios, que poderão ser abordados em investigações subsequentes. Recomenda-se que o foco destas investigações seja na continuação do trabalho já realizado, visando alcançar com êxito o objetivo de migrar a infraestrutura do Campus by Fundação Altice para *RedHat*.

## Bibliografia


Araújo, I., Pedro, L., Santos, C., & Batista, J. (2017). Crachás: como usar em contexto educativo? *Atas Da X Conferência Internacional de Tecnologias de Informação e Comunicação Na Educação. Challenges 2017: Aprender Nas Nuvens, Learning in the Clouds*, 159–176. https://ria.ua.pt/handle/10773/21672

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software*, *33*(3), 94–100. https://doi.org/10.1109/MS.2016.68

Fokina, A. (2013). *Metodologia Scrum e força de vendas no turismo: Ubiwhere*. https://ria.ua.pt/handle/10773/12240

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate : building and scaling high performing technology organizations*.

Holvitie, J., Licorish, S. A., Spínola, R. O., Hyrynsalmi, S., MacDonell, S. G., Mendes, T. S., Buchan, J., & Leppänen, V. (2018). Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology*, *96*, 141–160. https://doi.org/10.1016/J.INFSOF.2017.11.015

Humble, J., & Farley, D. (2010). *Continuous delivery : reliable software releases through build , test , and deployment automation*.

James, M. (2010). *Scrum Reference Card*. http://blogs.danube.com/estimation-game

Kelling, G. L., Wilson, J. Q., & others. (1982). Broken windows: The Police and Neighborhood Safety. *Atlantic Monthly*, *249*(3), 29–38.

Kim, G., Behr, Kevin., & Spafford, George. (2018). *The Phoenix Project, 5th Anniversary Edition : a Novel about IT, DevOps, and Helping Your Business Win.* 351.

Martin, R. C. (2009). *Clean Code: A Handbook of Agile Software Craftsmanship20092Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship .* (Issue 6).

Pathirage, M., Perera, S., Kumara, I., Weerasiri, D., & Weerawarana, S. (2012). A Scalable Multi-Tenant Architecture for Business Process Executions. *International Journal of Web Services Research*, *9*(2), 21–41. https://doi.org/10.4018/jwsr.2012040102

Reason, P., & Bradbury, H. (2008). *The SAGE Handbook of Action Research*. SAGE Publications Ltd. https://doi.org/10.4135/9781848607934

Santos, F. (2013). *Sistema de recomendações no SAPO Campus: desenvolvimento e avaliação: estudo de caso do mecanismo implementado no SAPO Campus*. https://ria.ua.pt/handle/10773/12312

Schwaber, K., & Sutherland, J. (2011). *The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game*.

Silva, J. H. (2021). “Gamification For All”: planning and designing a community-oriented gamification strategy. *GAME-ON’2021 - 22nd International Conference on Intelligent Games and Simulation*, 94–101. https://ria.ua.pt/handle/10773/35337

van den Akker, J. (1999). Principles and Methods of Development Research. *Design Approaches and Tools in Education and Training*, 1–14. https://doi.org/10.1007/978-94-011-4255-7_1

# Apêndices

## Apêndice 1 - Consentimento Informado Focus Group

**Tecnologias e metodologias de desenvolvimento Web:**

O caso da plataforma Campus by Fundação Altice

**Investigador**

Samuel Pinto dos Santos (samuelfmatias@ua.pt)

**Objetivo da Investigação**

O investigador (Samuel Pinto dos Santos) é aluno do mestrado de Comunicação e Tecnologias Web que está a conduzir o presente estudo no contexto da sua dissertação, com a coordenação dos professores Luís Pedro e Carlos Santos. O estudo tem como principal finalidade auxiliar no desenho e desenvolvimento de uma nova ferramenta tecnológica, capaz de ser integrada na plataforma Campus by Fundação Altice.

**Procedimento**

O Focus Group terá o objetivo de abordar a metodologia de trabalho atual e possíveis melhorias à mesma, com possibilidade de expansão a outros temas ou opiniões pertinentes ao estudo.

**Privacidade**

A informação recolhida será processada e divulgada apenas para fins científicos. Apenas os investigadores envolvidos na investigação terão acesso aos registos.

**Consentimento Informado**

Eu declaro que:

i. Recebi uma cópia deste documento;
ii. Li e percebi toda a informação presente neste documento e que fui informado de forma clara sobre os objetivos e condições de participação deste estudo;
iii. Tive a oportunidade de colocar questões e de ser informado de outros aspetos;
iv. Concordo em participar voluntariamente neste estudo.

Data: _____/_____/_____

O participante: _____

Obrigado pela sua participação nesta investigação.

## Apêndice 2 - Estrutura do Focus Group

### Abordagem geral a metodologias de desenvolvimento

- Apresentação e explicação genérica das metodologias Scrum e Kanban;
- Exposição em abstrato das vantagens e desvantagens de cada uma.

### Metodologia atual do Campus

- Vantagens e desvantagens da metodologia atualmente praticada na equipa;
- Será que a implementação pura de uma das metodologias apresentadas compensaria? Porquê?
- Quais as necessidades tecnológicas, organizativas, recurso-humanísticas e processuais da equipa ?
- Abrir a discussão sobre os ambientes atuais de desenvolvimento e se questionar sobre o ideal aproveitamento do ambiente de *staging*, caso o mesmo estivesse operacional;
- Qual deve ser a frequência das passagens para o ambiente de produção?
- E o cliente? Independentemente da metodologia, é suposto ele ser mais interventivo?
- Valerá a pena ter métricas (ex. DORA Metrics) e quais as necessidades ao nível de alarmística que isso implicaria?
- Valerá a pena definir regras e papéis concretos para as etapas da metodologia, nomeadamente na conclusão dos *Sprints* (verificação dos commits) ou para a gestão e conclusão dos tickets?
- Haveria mais facilidade em implementar estas ideias e metodologias se o Back-End trabalhasse um *Sprint* à adiantado em relação ao Front-End?
- O que mudariam na nossa equipa, a nível metodológico ou qualquer outro, caso o projeto fosse reiniciado sem nenhum constrangimento financeiro, de recursos humanos, metodológico ou tecnológico?

## Apêndice 3 - Guião das Entrevistas

### Participantes

Três ex-membros da equipa, especialistas a nível da infraestrutura tecnológica

### Introdução (5 min)

Antes de mais, gostaria de agradecer por ter aceitado o convite para participar nesta entrevista. Chamo-me Samuel Pinto dos Santos, sou aluno de Mestrado em Comunicação e Tecnologias Web na Universidade de Aveiro, e no âmbito da minha dissertação estou a estudar o desenvolvimento de uma nova ferramenta tecnológica, capaz de ser integrada na plataforma Campus by Fundação Altice e a aplicação de metodologias de trabalho em equipa nesse processo.

Dada a sua experiência passada na equipa do Campus by Fundação Altice e de desenvolvimento web em contexto profissional, foi convidado a participar neste estudo para que eu possa obter mais informações sobre a utilização de metodologias de trabalho em equipa e ferramentas que auxiliem essas metodologias ou o desenvolvimento técnico.

Esta entrevista terá a duração aproximada de 50 minutos. Assim, antes de prosseguir, gostaria de perguntar se este tempo não é um problema para si e se posso gravar a sessão para registar as suas respostas às minhas perguntas, de modo a poder analisá-las mais tarde. Podemos prosseguir? Tem alguma pergunta antes de começarmos?

### Ice-breaking (10min)

Para começar, gostaria de ouvir uma pequena introdução sobre si e sobre a sua experiência na equipa do Campus, as que áreas em estava a trabalhar nesse momento, e depois o seu percurso e aquisição de novos conhecimentos que o levaram à posição que tem agora.

**Questões (30min)**

**Como é que vê a aplicação de metodologias ágeis (ou outras) em equipa de desenvolvimento de software?**

- No passado, quando integrava a equipa do Campus, eram aplicadas algum tipo de metodologias? Se sim, quais e qual o seu benefício?
- Como é que são organizadas as equipas em que se insere e como é que elas comunicam entre si?
- Atualmente, na sua equipa de trabalho, aplica algum tipo de metodologia? *(se aplicável) Como se compara ao que era praticado no Campus?*

- Na sua equipa de trabalho utilizam alguma plataforma ou ferramenta como auxílio a essa metodologia? Qual, em que situações e de que maneira?
- Com a sua experiência, acha que a integração dessa metodologia e dessas ferramentas seria viável numa equipa como a do Campus?

**Atualmente, no papel que desempenha, tem contacto com alguma prática ou ferramenta de CI/CD (ex. Pipelines, Jenkins, Docker, etc.)?**

- Acha que, se durante o seu período na equipa do Campus, teria sido benéfico existir algum tipo de ferramenta DevOps?
- Alguma vez esteve encarregue de montar soluções destas? Como é que balança as prioridades entre a implementação dessas práticas e o desenvolvimento Web da equipa?
- Do que conhece da infraestrutura do Campus, consegue prever algum risco na implementação dessas práticas?
- Como é que mede o sucesso de uma implementação de DevOps?

**Agradecimentos (5min)**

Quero agradecer imensamente pela sua participação nesta entrevista para a minha investigação. A sua contribuição é muito valiosa. A informação que forneceu irá ajudar-me a compreender melhor o tema da minha investigação e a chegar a conclusões mais significativas. Muito obrigado pela sua colaboração.

## Apêndice 4 - Consentimento Informado das Entrevistas

**Tecnologias e metodologias de desenvolvimento Web:**

O caso da plataforma Campus by Fundação Altice

**Investigador**

Samuel Pinto dos Santos (samuelfmatias@ua.pt)

**Objetivo da Investigação**

O investigador (Samuel Pinto dos Santos) é aluno do mestrado de Comunicação e Tecnologias Web e está a conduzir o presente estudo no contexto da sua dissertação, com a coordenação dos professores Luís Pedro e Carlos Santos. O estudo tem como principal finalidade auxiliar no desenho e desenvolvimento de uma nova ferramenta tecnológica, capaz de ser integrada na plataforma Campus by Fundação Altice.

**Procedimento**

A entrevista será semiestruturada, que pressupõe a resposta a três perguntas, com a possibilidade de expansão a outros temas ou opiniões pertinentes ao estudo.

**Privacidade**

A informação recolhida será processada e divulgada apenas para fins científicos. As gravações de áudio das entrevistas semiestruturadas não serão divulgadas e serão mantidas em segredo. Apenas os investigadores envolvidos na investigação terão acesso aos registos.

**Consentimento Informado**

Eu declaro que:

i. Recebi uma cópia deste documento;

ii. Li e percebi toda a informação presente neste documento e que fui informado de forma clara sobre os objetivos e condições de participação deste estudo;

iii. Tive a oportunidade de colocar questões e de ser informado de outros aspetos;

iv. Concordo em participar voluntariamente neste estudo e aceito a gravação áudio da entrevista.

Data: _____/_____/_____

O participante: _____

Obrigado pela sua participação nesta investigação.

## Apêndice 5 - Questionários para a equipa do Campus by Fundação Altice

**Questionário para equipa Campus:**

No Focus Group de dia 21 de março foram levantadas necessidades de melhoria na metodologia de trabalho na nossa equipa. Desde então, foram implementadas algumas medidas para responder aos problemas encontrados. Para isso:

- Estabelecemos a *daily stand-up meeting* como imperativa;

- Melhorámos a comunicação com os coordenadores do projeto, expondo os resultados e o progresso semanal em reuniões todas as sextas-feiras (Sprint Review);

- Instituímos um momento de retrospetiva do *Sprint*, tipicamente sexta-feira à tarde, onde refletimos sobre o *Sprint* que acabou de ser concluído e identificamos maneiras de melhorar nosso trabalho no próximo Sprint;

- Estandardizámos as tarefas que eram propostas a cada pessoa para que fossem mais específicas, realistas de acordo com o tempo disponível, e fragmentadas em várias micro tarefas se necessário;

- Organizámos o *board* do Trello em colunas para melhor organizar as tarefas que devem ser executadas em cada Sprint vs tarefas em *backlog* ou *minor bugs/ fixes*;

- Desafiámos a equipa responsável pelo design a intervir no controlo de qualidade das soluções implementadas a cada *merge request*, libertando a equipa responsável pela *review* do código;

- Utilizámos o ambiente de dev para que a equipa de Design pudesse efetuar esses testes e os coordenadores poderem ter fácil acesso ao estado atual da plataforma, apesar de ainda não existir uma pipeline nem o ambiente de staging estar operacional;

- Determinámos que a equipa de BackEnd trabalhava nas tarefas com antecedência em relação à equipa da Front-End;

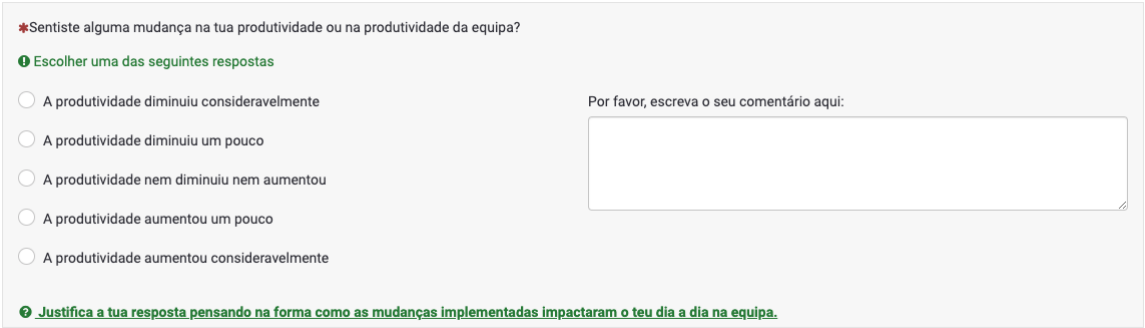
- Aprimorámos a documentação do projeto no Notion, à medida que nos deparávamos com certos problemas, em antecipação e para apoiar os novos membros da equipa;

- Criámos um log no Teams, em tempo real, dos commits efetuados nos repositórios de Front-End e Back-End, para transmitir transparência e para que cada membro estivesse a par do trabalho do resto da equipa;

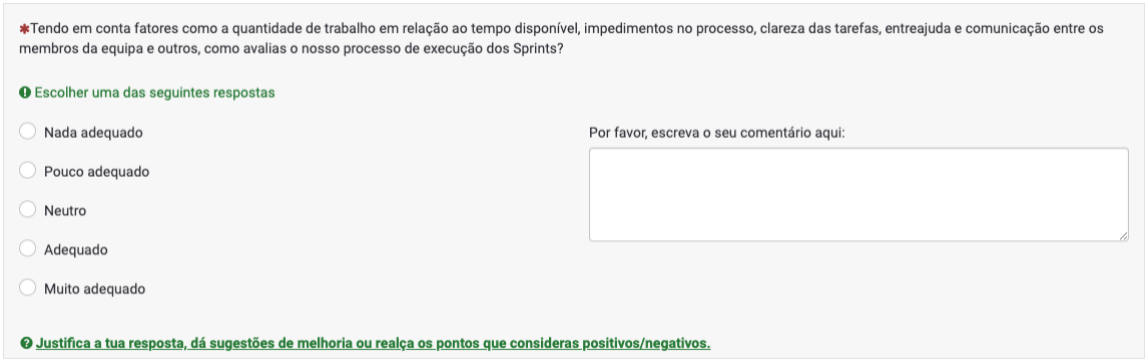
- Fundámos um hábito rotineiro de comunicação com o cliente, tendo reuniões frequentes com a Fundação Altice (tipicamente de 2 em 2 meses);

Tendo em conta a tua experiência na equipa, a metodologia de trabalho antes destas mudanças e todo o progresso que foi feito desde então:

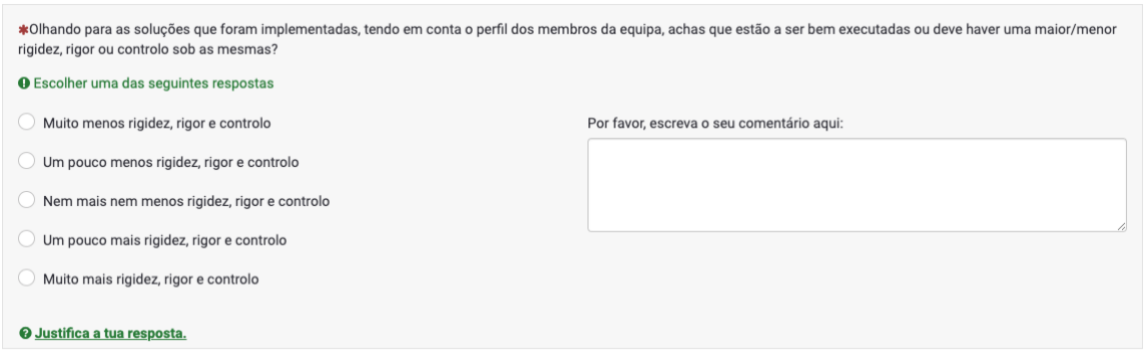
**a) Sentiste alguma mudança na tua produtividade ou na produtividade da equipa?**



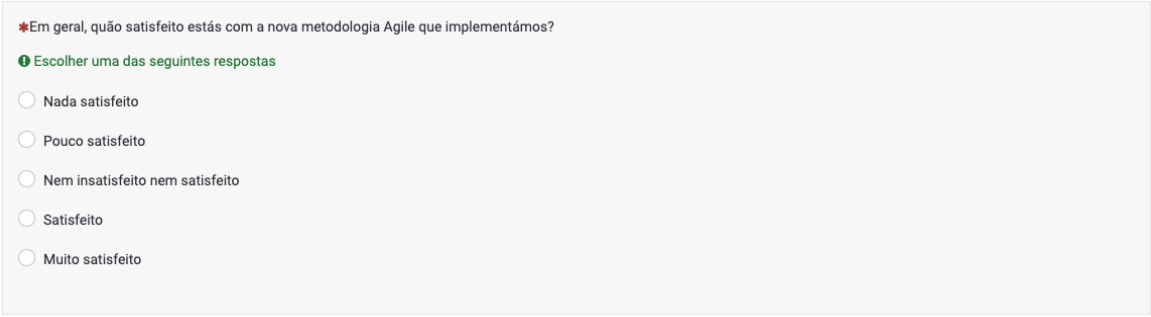

**b) Tendo em conta fatores como a quantidade de trabalho em relação ao tempo disponível, impedimentos no processo, clareza das tarefas, entreajuda e comunicação entre os membros da equipa e outros, como avalias o nosso processo de execução dos *Sprints*?**




**c) Olhando para as soluções que foram implementadas, tendo em conta o perfil dos membros da equipa, achas que estão a ser bem executadas ou deve haver uma maior/menor rigidez, rigor ou controlo sob as mesmas?**




**d) Em geral, quão satisfeito estás com a nova metodologia Agile que implementámos?**

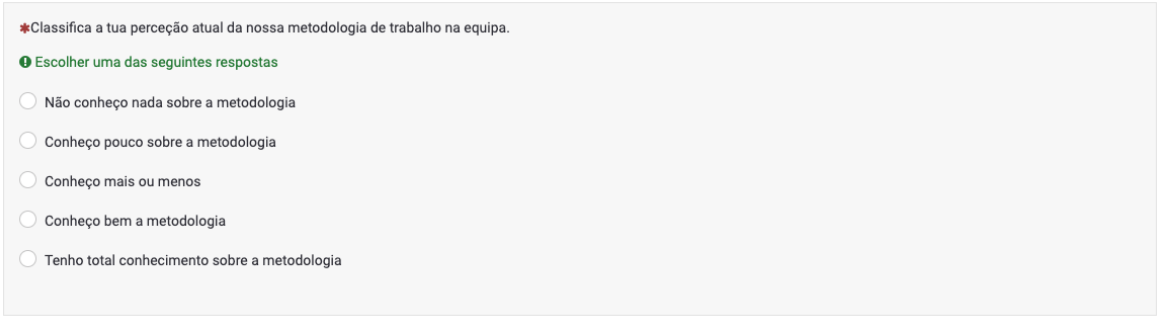



**e) Tens alguma sugestão de melhoria ou outro comentário em relação à metodologia aplicada atualmente na nossa equipa? Se sim, qual ou quais?** (Resposta Aberta)

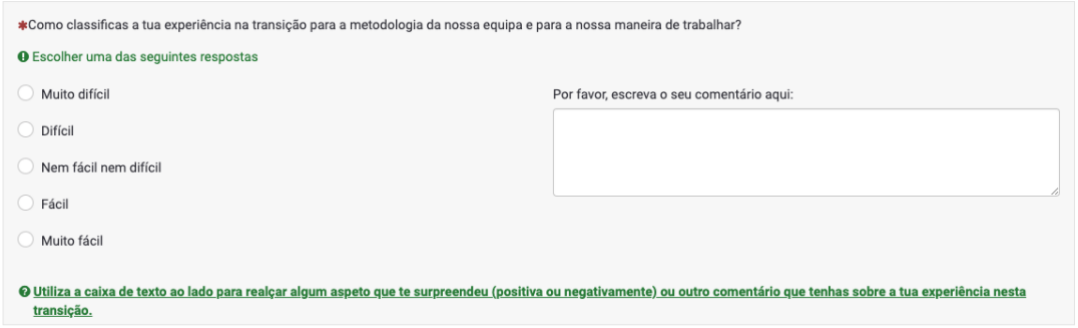
**Questionário para os novos membros da equipa Campus:**

Antes de mais, bem-vindo à equipa do Campus!
Este questionário serve para aferir a pertinência da atual metodologia de trabalho na nossa equipa. Para tal, tem em consideração o teu tempo na equipa e a forma como comunicamos entre nós, organizamos o nosso trabalho e planeamos as tarefas em cada Sprint.

a) **Antes de entrar para a equipa, já tinhas experiência em trabalhar com metodologias ágeis?** (Sim ou Não)
b) (Se sim) **Que metodologia ou processos é que eram usados na tua posição anterior?**
c) **Classifica a tua perceção atual da nossa metodologia de trabalho na equipa.**




d) **Classifica a tua perceção atual da nossa metodologia de trabalho na equipa.**




e) **Tens alguma sugestão para melhorar a metodologia de trabalho atual, a colaboração ou comunicação entre os membros da equipa?** (Resposta Aberta)