

**Desenvolvimento de uma plataforma
de gestão de projetos**

Blended4Future

1201367 Carlos Rodrigo Marques dos Santos

Orientador: Professor Nuno Escudeiro

Porto, Setembro, 2025

Agradecimentos

Ao professor Nuno Escudeiro, na qualidade de orientador, pelo convite para integrar este projeto e pela confiança demonstrada ao longo do mesmo.

Ao professor Ricardo Almeida, pelo apoio contínuo e valiosas orientações disponibilizadas durante o desenvolvimento do projeto.

À minha família e à minha namorada, pelo apoio incondicional, pelo carinho e atenção dedicados, e por me ouvirem e motivarem nos momentos mais difíceis.

Conteúdo

Lista de Figuras	v
Lista de Tabelas	vi
Lista de Código	vii
1 Introdução	1
1.1 Enquadramento	1
1.1.1 Apresentação da organização	1
1.2 Descrição do Problema	2
1.3 Objetivos	2
1.4 Abordagem	3
1.4.1 A equipa	3
1.4.2 Metodologia de trabalho	3
1.4.2.1 Divisão de sprints	3
1.4.3 Tecnologias Definidas	4
1.4.3.1 Ferramentas de Gestão de projetos	4
1.4.3.2 Tecnologias para desenvolvimento	6
1.5 Desafios enfrentados	9
2 Estado da Arte	10
2.1 Contexto	10
2.2 ESN - European Student Network	11
2.2.1 erasmusintern.org	11
2.3 PRAXIS	12
2.4 Spain Internship	13
3 Analise do problema e desenho de uma solução	17
3.1 O Dominio	17
3.2 Engenharia de requisitos	17
3.2.1 Requisitos funcionais	17
3.2.1.1 Diagram de User Flow	19
3.2.2 Requisitos não Funcionais	19
3.2.2.1 FURPS	19
3.3 Arquitetura do sistema	24
3.3.1 Solução	24
3.3.2 Nível 1 - Contexto	24

3.3.2.1	Diagrama Lógico	24
3.3.2.2	Diagrama Físico	24
3.3.3	Nível 2	25
3.3.3.1	Diagrama Lógico	25
3.3.3.2	Diagrama Físico	25
3.3.4	Nível 3	26
3.3.4.1	Diagrama Lógico	26
3.3.4.2	Diagrama Físico	26
3.3.5	Nível 4	27
3.3.5.1	Diagrama Físico	27
4	Implementação de uma solução	29
4.1	A Implementação	29
4.1.1	Backend	29
4.1.1.1	Spring	29
4.1.1.2	Estrutura de pastas	30
4.1.1.3	Entidades e relações	31
4.1.1.4	<i>BaseEntity</i>	32
4.1.1.5	<i>Value Objects</i>	33
4.1.1.6	Repositórios	35
4.1.1.7	Controladores e prevenção de erros	36
4.1.1.8	Serviços	39
4.1.1.9	Documentação	41
4.1.2	Frontend	41
4.1.2.1	React	41
4.1.2.2	Paginas Estáticas	42
4.1.2.3	Pagina de Visualização de Projetos	44
4.1.3	Controlo de Versões	45
4.1.4	Deployments	45
4.1.4.1	Definição dos agentes	46
4.1.4.2	Trabalhos da pipeline	46
4.2	Testes	48
4.2.1	Testes Unitários	48
4.2.2	Testes de Implementação	49
5	Conclusão	50
5.1	Desafios	50
5.2	Objetivos Concluídos	50
5.3	Apreciação Final	50
Bibliografia		52
Apêndice A Diagrama de User Flow		53

Apêndice B Script para conexão de uma máquina à agent pool do Azure Pipelines	56
Apêndice C Dockerfile do Agente Azure Pipelines	59
Apêndice D Pipeline criada em formato YAML	60

Listas de Figuras

1.1	Homepage do projeto	5
1.2	Exemplificação do uso das <i>boards</i> no DevOps	5
2.1	<i>Website</i> do erasmusintern.org	11
2.2	Filtros de pesquisa divididos por <i>Area de Estudo</i> do erasmusintern.org	12
2.3	<i>Home page</i> da praxisnetwork.eu	13
2.4	Página de pesquisa da PRAXIS	13
2.5	<i>Home Page</i> da <i>Spain Internship</i>	14
2.6	Página de pesquisa da <i>Spain Internship</i>	15
2.7	<i>Dropdown list</i> para as páginas de contacto	15
2.8	Página de contacto de <i>partners</i>	16
3.1	Diagrama de classes da lógica de negócio do Blended4Future 1	18
3.2	Diagrama Lógico de Nível 1	24
3.3	Diagrama Físico de Nível 1	24
3.4	Diagrama Lógico de Nível 2	25
3.5	Diagrama Físico de Nível 2	25
3.6	Diagrama Lógico de Nível 3	26
3.7	Diagrama Físico de Nível 3	27
3.8	Diagrama Físico de Nível 4	28
4.1	Estrutura de pastas do backend	30
4.2	<i>Webpage</i> do <i>Swagger</i>	41
4.3	Página <i>Elevator Pitch</i> de estudante - parte 1	42
4.4	Página <i>Elevator Pitch</i> de estudante - parte 2	43
4.5	Página <i>Elevator Pitch</i> de estudante - parte 3	43
4.6	Página de visualização de projetos	45
4.7	Script de inicialização do Container Docker para o agente do Azure Pipelines	46
4.8	Agente disponibilizado no Azure DevOps	47
4.9	Diagrama de atividade representante da Pipeline de construção de uma nova versão	47
A.1	Diagram de User Flow da Aplicação 1	53
A.2	Diagram de User Flow da Aplicação 2	54
A.3	Diagram de User Flow da Aplicação 3	55

Lista de Tabelas

1.1	Organização de sprints definida pela equipa	4
1.2	Levantamento das tecnologia que os elementos da equipa já utilizaram	7
3.1	Lista de requisitos funcionais	22
3.2	Lista de requisitos não funcionais	23

Listas de Código

4.1	Classe <i>Project</i>	31
4.2	Classe <i> BaseEntity</i>	32
4.3	Interface <i>Value Object</i>	34
4.4	Classe <i>ProjectDescription</i>	34
4.5	Inteface <i>BaseRepository</i>	35
4.6	interface <i>ProjectRepository</i>	36
4.7	Class <i>AppExceptionController</i>	36
4.8	Class <i>Error</i>	37
4.9	Class <i>ProjectController</i>	37
4.10	Exceção <i>AppException</i>	38
4.11	Implementação do metodo para criação de um novo projeto	39
4.12	Função <i>LandingPageContainer</i>	42
4.13	Função <i>ProjectLibraryPage</i>	44
4.14	Trigger - Pipeline	47
4.15	Class <i>CompanyTest</i> - Exemplificação de testes Unitários	48
4.16	Script de test da rota POST /company	49

Capítulo 1

Introdução

O presente capítulo visa fornecer uma visão geral que enquadra e organiza os diferentes aspetos que serão aprofundados nas secções subsequentes.

No seu conteúdo, apresenta-se o enquadramento e a relevância do projeto desenvolvido no âmbito da unidade curricular de Projeto Estágio (PESTI). Este trabalho integrou-se no programa internacional Blended4Future, uma iniciativa que reúne estudantes de diversas instituições europeias com o propósito de promover o desenvolvimento de competências técnicas e colaborativas através da realização de projetos aplicados a necessidades reais de empresas parceiras.

Serão descritos o contexto em que o projeto foi concebido, o problema central identificado e os objetivos definidos para a sua resolução. Além disso, explicita-se a organização da equipa de trabalho e a metodologia adotada para guiar o processo de desenvolvimento.

Por último, apresentam-se as principais tecnologias selecionadas para a implementação da solução e faz-se referência aos desafios mais significativos enfrentados ao longo da execução do projeto.

1.1 Enquadramento

Este relatório foi desenvolvido com base num projeto enquadrado no âmbito da unidade curricular de Projeto Estágio (PESTI) da Licenciatura em Engenharia Informática (LEI) do Instituto Superior de Engenharia do Porto (ISEP).

Este projeto ocorreu no enquadramento do Projeto BlendED (também referido como Blended4Future ou BlendedMobility¹). Este curso dá a estudantes a oportunidade de desenvolverem as suas *soft* e *hard skills* num projeto com alunos de diferentes culturas e países trabalhando num projeto para empresas reais.

1.1.1 Apresentação da organização

BlendedMobility é uma iniciativa educativa internacional que promove o desenvolvimento de projetos colaborativos no contexto do ensino híbrido. Esta visa combinar

¹<https://blendedmobility.com>

experiências de aprendizagem presencial e *online*, proporcionando aos estudantes uma formação mais flexível, personalizada e orientada para a prática.

O programa reúne alunos de diferentes universidades europeias que, ao longo de quatro meses, trabalham em conjunto no desenvolvimento de projetos reais para empresas parceiras. A metodologia adotada assenta em práticas ativas e colaborativas, potenciando competências essenciais como trabalho em equipa, comunicação intercultural e resolução de problemas num ambiente profissional simulado.

O percurso inicia-se com uma semana presencial no Instituto Universitário da Maia (ISMAI), onde as equipas se conhecem, recebem o enquadramento do projeto e planeiam as etapas de trabalho. O desenvolvimento dos projetos decorre num regime híbrido, combinando sessões *online* e trabalho autónomo. Ao final do ciclo, todas as equipas reúnem-se presencialmente na Universidade de Trier, na Alemanha, para apresentar os resultados dos seus projetos a um painel de avaliadores e representantes das empresas envolvidas.

1.2 Descrição do Problema

O Website do curso Blended4Future estava muito aquém do esperado, muitos elementos seguiam um design inconsistente e antiquado, ou não estavam completamente funcionais.

A organização desejava uma plataforma onde se pudesse automaticamente adicionar projetos, alunos, universidades e empresas numa só plataforma. Por tal foi colocada uma proposta de desenvolvimento de uma nova aplicação *web* que substituiria esta anterior.

Nesta aplicação, estudantes professores e representantes de empresas poderiam ver os projetos em que estavam envolvidos e fazer posts sobre os seus respetivos projetos.

Foi ainda requisitado uma maneira de ver todas as edições do Blended4Future e todos os projetos neste envolvido.

1.3 Objetivos

A aplicação web a desenvolver deverá incluir um sistema de autenticação com diferenciação entre perfis de utilizador, nomeadamente administradores e utilizadores comuns, assegurando uma gestão adequada de permissões.

Adicionalmente, deverá permitir a criação de novos projetos e a associação de diferentes intervenientes a cada um, consoante o seu papel. Além disso, deverá ser implementada uma biblioteca de projetos, acessível a qualquer utilizador da plataforma, onde estarão disponíveis todos os projetos desenvolvidos no âmbito do curso, promovendo a sua consulta e divulgação.

A interface da aplicação deverá seguir as diretrizes de design definidas previamente por um membro da equipa dedicado ao design visual da aplicação.

1.4 Abordagem

1.4.1 A equipa

A equipa foi composta por um grupo de estudantes provenientes de várias universidades europeias, com a seguinte constituição:

- 5 Desenvolvedores
- 1 Designer
- 1 Marketer

A equipa de IT, constituída por cinco desenvolvedores, foi organizada em duas subequipes:

- 3 estudantes dedicados ao desenvolvimento de *Backend*
- 2 estudantes dedicados ao desenvolvimento de *Frontend*, incluindo o autor deste relatório, que integrou esta subequipa para participar no desenvolvimento da interface de utilizador.

Esta divisão teve como objetivo garantir um maior avanço na lógica de negócio durante a fase inicial do projeto. Numa etapa posterior, quando a lógica estivesse próxima da sua conclusão, alguns dos estudantes poderiam transitar para a subequipa de *Frontend*, focando-se então na criação da interface de utilizador.

É importante salientar que, no âmbito do projeto Blended4Future, cada estudante participou com um número distinto de créditos (ECTS), definidos em função do seu respetivo curso. Assim, tornou-se necessário organizar a distribuição das tarefas de forma proporcional, garantindo que a carga de trabalho atribuída a cada elemento refletisse adequadamente o valor dos créditos em que estava inscrito.

1.4.2 Metodologia de trabalho

Para uma melhor organização do trabalho, foi adotada a metodologia Scrum, com sprints de duas semanas de duração. Além disso, foi estabelecida, por consenso do grupo, a realização de reuniões semanais para atualizar o progresso das tarefas e promover um ambiente de trabalho mais colaborativo e comunicativo. Estas reuniões não possuíam uma data definida tendo em conta os fusos horários de cada um dos membros.

1.4.2.1 Divisão de sprints

A tabela 1.1 demonstra a divisão de sprints feita. Foi ainda definido objetivos para os primeiros quatro sprints.

Objetivos Sprint 1–2

- Documentação geral do projeto
- Configuração dos repositórios de Backend/Frontend
 - Pipelines de CI/CD
- Configuração de DevOps/Azure
- Análise SWOT do negócio

Objetivos Sprint 3–4

- Finalização da documentação geral
- Início do desenvolvimento

Sprint	Semanas	Data de início	Data de fim
1	1-2	24/02	02/03
2	3-4	10/03	23/03
3	5-6	24/03	06/04
4	7-8	07/04	20/04
5	9-10	21/04	04/05
6	11-12	05/05	18/05
7	13-14	19/05	01/06
8	15-16	02/06	15/06

Tabela 1.1: Organização de sprints definida pela equipa

A tabela 1.1 demonstra a organização de sprints que foi escolhida pela equipa durante a primeira semana de *kickoff* do projeto.

1.4.3 Tecnologias Definidas

Durante a semana de kickoff tornou-se necessário deliberar sobre as tecnologias e ferramentas a adotar para o desenvolvimento do projeto. Ao longo da execução, foram incorporadas soluções adicionais visando otimizar processos e aumentar a eficiência do trabalho realizado.

1.4.3.1 Ferramentas de Gestão de projetos

Primeiramente foi necessário definir uma plataforma central para onde cada elemento do grupo pudesse ver o estado do projeto, incluindo ainda a totalidade do *product backlog*, as suas respetivas tarefas, duração dos sprints e onde se pudesse ter acesso a todos os repositórios necessários ao projeto.

Por aconselhamento dos professores membros dos projeto foi escolhido a aplicação da *Azure DevOps*. (Ver fig. 1.1 e 1.2)

Esta ferramenta destaca-se como uma plataforma integrada que oferece um equilíbrio entre funcionalidades avançadas e facilidade de utilização, suportando todo o ciclo de vida do desenvolvimento de software. A sua interface intuitiva, combinada com um elevado grau de personalização dos painéis de trabalho, automações eficazes e relatórios detalhados, facilita a gestão ágil e a integração com o ecossistema Microsoft, bem como com diversas outras ferramentas DevOps.

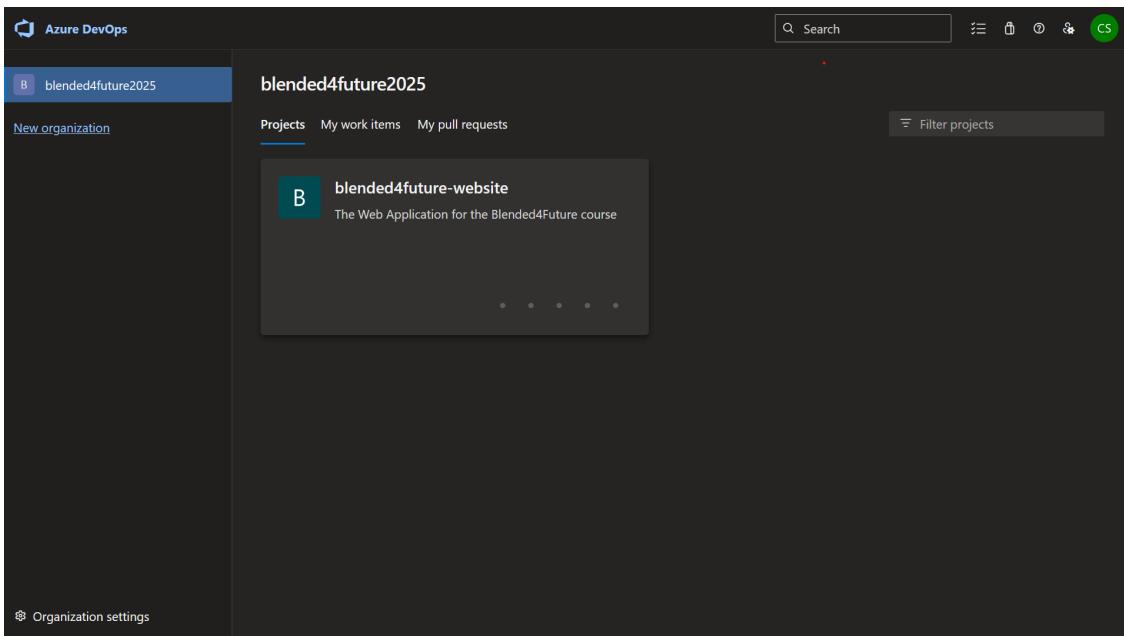


Figura 1.1: Homepage do projeto

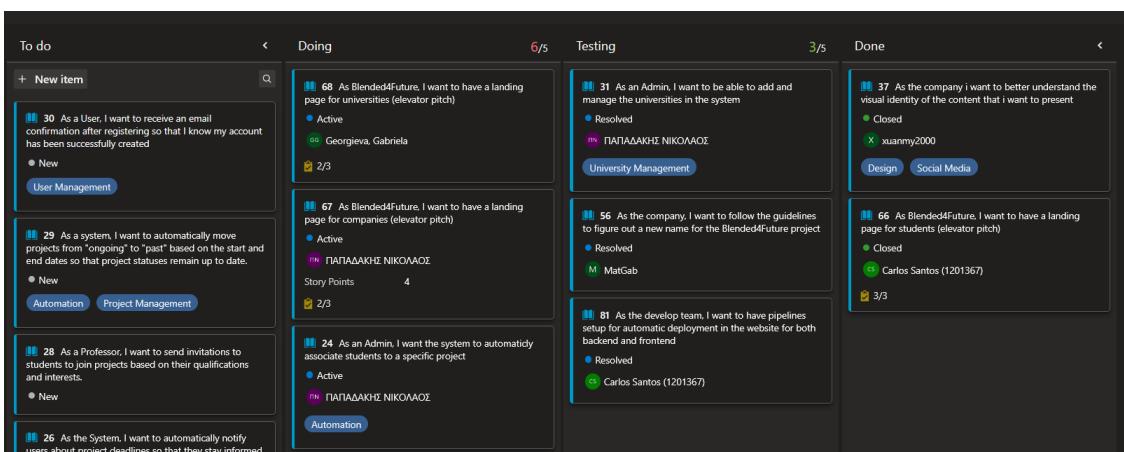


Figura 1.2: Exemplificação do uso das *boards* no DevOps

Outras possíveis e válidas escolhas para este tipo de ferramenta seriam, por exemplo:

GitHub Projects Conhecido pela sua simplicidade e integração direta no ambiente de desenvolvimento, o GitHub Projects permite um acompanhamento natural e ágil das tarefas dentro dos próprios repositórios de código. Esta característica potencia a colaboração entre programadores e torna a ferramenta particularmente útil para

projetos de pequena escala ou equipas que privilegiam a rapidez e a facilidade de uso. No entanto, apresenta limitações em termos de funcionalidades de gestão ágil e geração de relatórios detalhados, pelo que pode não ser adequado para projetos com requisitos organizacionais mais rigorosos ou estruturas de equipa complexas.

Jira O Jira apresenta-se como uma ferramenta de gestão de projetos altamente robusta, especialmente orientada para equipas que adotam metodologias ágeis, como Scrum e Kanban. Destaca-se pela capacidade de personalização detalhada dos fluxos de trabalho, pela integração aprofundada com ferramentas DevOps e pela produção de relatórios analíticos complexos, que facilitam o acompanhamento rigoroso do progresso em projetos de elevada complexidade. Contudo, a sua curva de aprendizagem é acentuada e a interface pode revelar-se excessivamente densa para utilizadores menos experientes, o que implica um investimento significativo em configuração e formação inicial, podendo limitar a sua eficiência em equipas pequenas ou com recursos reduzidos.

1.4.3.2 Tecnologias para desenvolvimento

Atendendo à heterogeneidade da equipa, composta por indivíduos com diferentes experiências, origens académicas e nacionalidades, tornou-se imperativo definir consensualmente as tecnologias a utilizar no projeto. Para tal, procedeu-se ao levantamento sistemático das ferramentas previamente utilizadas pelos elementos da equipa, tendo-se optado, em média, pelas tecnologias mais frequentemente referidas e consideradas mais adequadas à criação de interfaces de utilizador e à implementação de APIs do tipo REST.

Houve uma clara preferência quanto ao uso de linguagens com tipagem forte. Estas permitem o melhor rastreamento de erros e garantem o formato dos objetos no programa. Esta preferência teve como base principal a inexperiência de vários elementos da equipa.

A tabela 1.2 demonstra este levantamento que permitiu à equipa fundamentar a decisão sobre as tecnologias a adoptar:

Java Linguagem de programação orientada a objetos amplamente utilizada no desenvolvimento de aplicações empresariais e sistemas robustos. A sua integração com o ecossistema Spring possibilita a criação eficiente de APIs REST, suportando a escalabilidade e manutenção do software. A sua tipagem forte permite que erros sejam mais fáceis de evitar, principalmente para desenvolvedores com pouca experiência.

Spring Framework para a plataforma Java caracterizada pela sua modularidade e flexibilidade, que simplifica o desenvolvimento de aplicações empresariais. Entre as suas principais vantagens técnicas destacam-se a Inversão de Controlo e a Injeção de Dependência, que promovem baixo acoplamento e facilitam a manutenção e testabilidade do código. Suporta também Programação Orientada a objetos, permitindo

Tecnologia	Nº de referências
Java	5
Python	4
CSS	4
MySQL	4
HTML	5
SQL Server	3
C	2
JavaScript	5
TypeScript	2
C++	2
Bash Scripting	1
R	1
ReactTS	1
C#	1
.NET (ASP)	1
Entity Framework	1
H2	1
MongoDB	1
PHP	1

Tabela 1.2: Levantamento das tecnologia que os elementos da equipa já utilizaram

isolar funcionalidades transversais como segurança e logging. Entre as suas várias funcionalidades destaca-se o Spring Data, que abstrai o acesso a repositórios e bases de dados, facilitando a interação com diversos sistemas de armazenamento por meio de interfaces padronizadas; assim, os repositórios do Spring permitem a execução simplificada de operações CRUD e consultas complexas sem necessidade de escrever código SQL.

Junit Framework amplamente utilizada no desenvolvimento Java para a criação de testes unitários, essencial para garantir a qualidade e fiabilidade do código ao permitir a verificação automática e recetível do comportamento de unidades individuais numa aplicação. Este framework proporciona uma estrutura simples e eficiente para escrever, organizar e executar testes, suportando anotações que facilitam a definição de métodos de teste, fases de inicialização e limpeza, bem como a gestão de exceções esperadas.

TypeScript Linguagem de programação baseada em JavaScript, que integra tipagem estática e recursos avançados, facilitando o desenvolvimento de interfaces de utilizador complexas e a manutenção de grandes bases de código. A sua adoção é relevante na construção de aplicações web modernas e escaláveis.

React TS Framework de desenvolvimento frontend baseada em componentes, que alia a flexibilidade do React à segurança oferecida pela tipagem forte do TypeScript. Esta combinação permite construir interfaces de utilizador dinâmicas e com maior robustez, suportando os modernos paradigmas de desenvolvimento web com principal foco na reusabilidade dos referidos componentes.

TailwindCSS Framework utilitária que simplifica a estilização ao permitir aplicar classes diretamente nos componentes de React, eliminando a necessidade de escrever CSS tradicional. A sua integração assegura uma personalização rápida e responsiva, facilitada pela configuração flexível e pelo suporte a interfaces modulares e tipadas com TypeScript, promovendo assim maior produtividade e manutenção eficiente do código em projetos modernos de frontend.

MySQL : Sistema de gestão de bases de dados relacionais (SGBDR) amplamente utilizado, reconhecido pela sua natureza *open source* e pelo desempenho eficiente em operações de leitura e escrita, particularmente em aplicações web. Entre as suas vantagens técnicas destaca-se a facilidade de uso, a escalabilidade para lidar com grandes volumes de dados e o suporte a transações ACID. Adicionalmente, o MySQL oferece mecanismos de segurança sólidos, como autenticação, autorização e criptografia, reforçando a proteção dos dados armazenados.

1.5 Desafios enfrentados

Esta secção é apresentada na perspetiva do autor do relatório, procurando não apenas identificar os principais obstáculos encontrados ao longo do projeto, mas também refletir sobre as aprendizagens decorrentes da experiência, em especial no que respeita à gestão de equipa e à adaptação em contextos de incerteza.

Um dos desafios mais marcantes prendeu-se com a falta de comunicação entre os membros da equipa. Esta dificuldade comprometeu a coordenação das atividades e levou, em diversas ocasiões, ao incumprimento de tarefas previamente atribuídas. Acresce que a ausência de contributos regulares por parte de alguns elementos originou constrangimentos adicionais, dificultando a execução contínua de certas fases do projeto.

Como tentativa de mitigar estas dificuldades, procurou-se aumentar a frequência de reuniões e envolver mais ativamente os professores responsáveis pela orientação. No entanto, estas medidas revelaram-se insuficientes, o que levou, sensivelmente a meio do projeto, à necessidade de reduzir significativamente o âmbito (*scope*) inicialmente definido, ajustando a complexidade do domínio de trabalho.

Em retrospectiva, é importante reconhecer que alguns elementos da equipa não demonstraram capacidade ou disponibilidade para contribuir de forma consistente. Enquanto *Scrum Master*, teria sido desejável adotar uma postura mais preventiva e, principalmente, assertiva, intervindo mais cedo quando estes sinais começaram a emergir, de modo a minimizar o impacto na progressão do projeto.

Capítulo 2

Estado da Arte

Esta secção tem como objetivo apresentar o estado da arte, analisando um conjunto de ferramentas e/ou produtos que partilham funcionalidades, ou que assentam em modelos de negócio semelhantes ao *Blended4Future*. Em particular, privilegia-se a exploração de plataformas digitais concebidas para a divulgação, gestão e acompanhamento de projetos ou estágios de carácter internacional, onde a colaboração entre instituições, empresas e estudantes assume um papel central.

Para além de identificar soluções com finalidades comparáveis, pretende-se também realçar as suas principais características distintivas, apontando os pontos fortes, limitações e contextos de utilização.

2.1 Contexto

As ferramentas de procura de estágios *online* direcionadas para estudantes Erasmus desempenham um papel central na mobilidade académica e profissional dentro do espaço europeu. Estas plataformas surgem como mediadoras entre estudantes que procuram uma experiência internacional de trabalho e organizações que disponibilizam oportunidades de estágio, funcionando como pontos de encontro digitais que facilitam a divulgação, a pesquisa e a seleção de estágios.

A sua principal vantagem reside na capacidade de centralizar num único espaço inúmeras ofertas provenientes de diferentes países, permitindo ao estudante filtrar as oportunidades de acordo com critérios como a área de estudo, a duração do estágio, o país de destino ou o tipo de instituição de acolhimento. Este mecanismo é particularmente relevante no contexto do programa Erasmus+, uma vez que os estudantes têm frequentemente requisitos específicos a cumprir, quer em termos académicos (reconhecimento de créditos ECTS), quer em termos de apoio logístico (alojamento, bolsas ou compatibilidade com calendários universitários).

É importante compreender que o website do *Blended4Future* tem como principal função a apresentação dos projetos que foram ou que se encontram em desenvolvimento no âmbito do programa. Por sua vez, as plataformas analisadas neste estado da arte centram-se sobretudo na oferta de estágios internacionais. Apesar desta diferença de enfoque, a análise destas ferramentas revela-se pertinente, uma vez

que partilham um modelo de negócio semelhante, baseado na interligação entre estudantes, instituições de ensino e empresas, visando a promoção de oportunidades de desenvolvimento académico e profissional em contexto internacional.

2.2 ESN - European Student Network

A Erasmus Student Network (ESN) é uma organização sem fins lucrativos fundada em 1989 e atualmente presente em mais de 40 países europeus. O seu objetivo principal é apoiar e enriquecer a experiência de estudantes em mobilidade internacional, nomeadamente no âmbito do programa Erasmus+, mas também de outros programas de intercâmbio académico e profissional. Através de uma vasta rede de secções locais, a ESN promove atividades culturais, sociais e profissionais que favorecem a integração dos estudantes nos países de acolhimento.

Para além das suas atividades presenciais, a ESN tem desenvolvido várias plataformas digitais que servem de apoio à comunidade estudantil, oferecendo acesso a informações sobre oportunidades de estágio, bolsas de estudo, e até serviços de voluntariado internacional. Estas ferramentas constituem um importante meio de ligação entre estudantes, instituições e empresas, permitindo assim fomentar uma experiência de mobilidade mais completa e enriquecedora.

2.2.1 erasmusintern.org

Dentro da ampla rede de iniciativas desenvolvidas pela Erasmus Student Network (ESN), destaca-se particularmente o projeto *ErasmusIntern.org*, que constitui o departamento dedicado à divulgação de oportunidades de estágio em formato online.

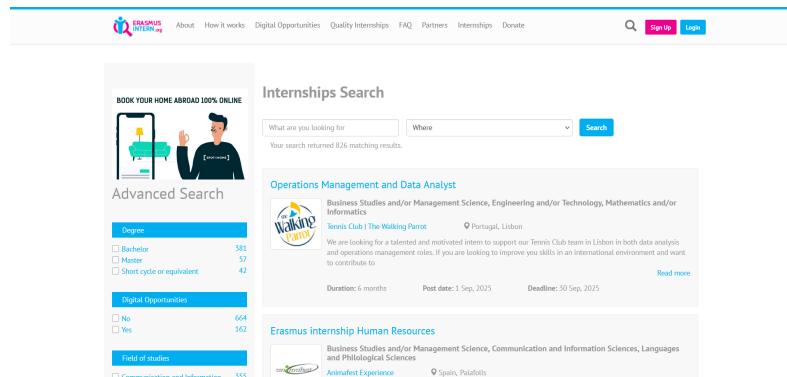


Figura 2.1: Website do erasmusintern.org

Num contexto desta natureza, em que a diversidade académica e cultural dos utilizadores é extremamente elevada, a disponibilização de mecanismos de pesquisa eficazes assume um papel central. A existência de filtros específicos, nomeadamente por área de estudo, setor de atividade, localização geográfica ou duração do estágio, torna-se fundamental para garantir que os estudantes encontram rapidamente as oportunidades que melhor correspondem ao seu perfil e às suas expectativas.

A plataforma *ErasmusIntern.org* destaca-se pela abrangência, oferecendo oportunidades em diversas áreas de estudo e permitindo ao utilizador recorrer a um vasto conjunto de filtros de pesquisa para personalizar a sua procura (ver Figura 2.2).

Em contraste, o *Blended4Future* adota uma abordagem mais focada, centrando-se sobretudo em três áreas específicas: Informática, Marketing e Design. Ao invés de promover uma dispersão de oportunidades, o projeto visa potenciar a interdisciplinaridade entre estas áreas, reunindo estudantes de diferentes especializações num mesmo contexto de colaboração. Esta estratégia procura não apenas responder às necessidades de mercado nestes domínios, mas também tende a refletir projetos reais e a treinar os estudantes para um mercado de trabalho exigente, proporcionando-lhes a oportunidade de colaborar diretamente com empresas parceiras.



Figura 2.2: Filtros de pesquisa divididos por *Area de Estudo* do *erasmusintern.org*

2.3 PRAXIS

A PRAXIS uma função bastante semelhante à desempenhada pela *ErasmusIntern.org*, centrando-se igualmente na promoção e intermediação de estágios internacionais para estudantes universitários. A PRAXIS Network resulta de uma colaboração entre diversas instituições de ensino superior europeias, procurando criar uma rede sólida que facilite a mobilidade estudantil e, simultaneamente, aumente a cooperação entre universidades e entidades empregadoras.

Na página inicial da plataforma PRAXIS evidencia-se, desde logo, a centralidade atribuída aos dois atores fundamentais deste tipo de iniciativa: os estudantes, enquanto beneficiários diretos das oportunidades de estágio, e as empresas, que se apresentam como entidades promotoras. Esta lógica de funcionamento assenta na criação de um ponto de encontro entre oferta e procura, permitindo um processo de intermediação simples e acessível.

De forma semelhante, o Blended4Future identifica como público-alvo tanto os estudantes como as empresas, reconhecendo, no entanto, um terceiro interveniente igualmente relevante: as universidades. A inclusão das instituições de ensino superior representa um diferencial importante, uma vez que estas desempenham um papel central na mediação académica e na validação das experiências desenvolvidas. Para além de atuar como intermediário entre alunos e organizações, o projeto procura também atrair um número crescente de universidades parceiras, reforçando a credibilidade da plataforma e ampliando o alcance das oportunidades disponibilizadas.

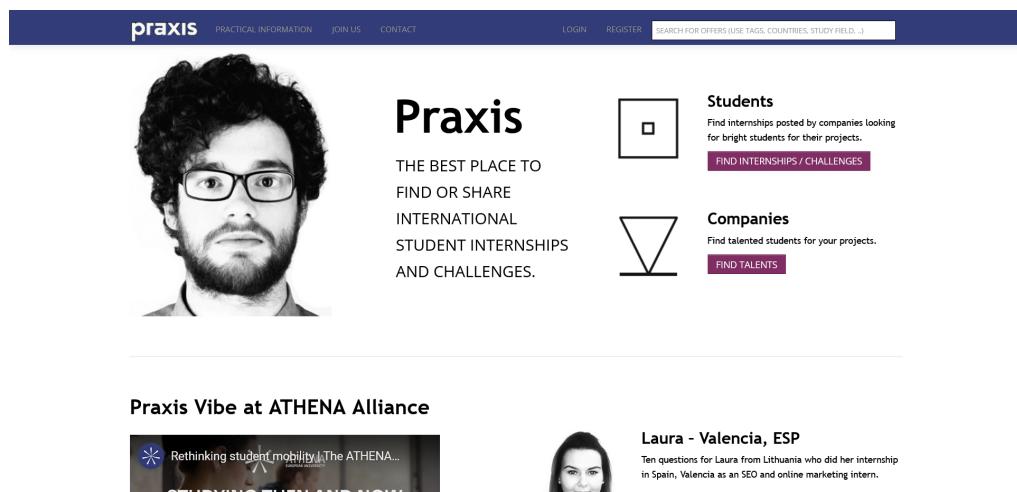


Figura 2.3: *Home page* da praxisnetwork.eu

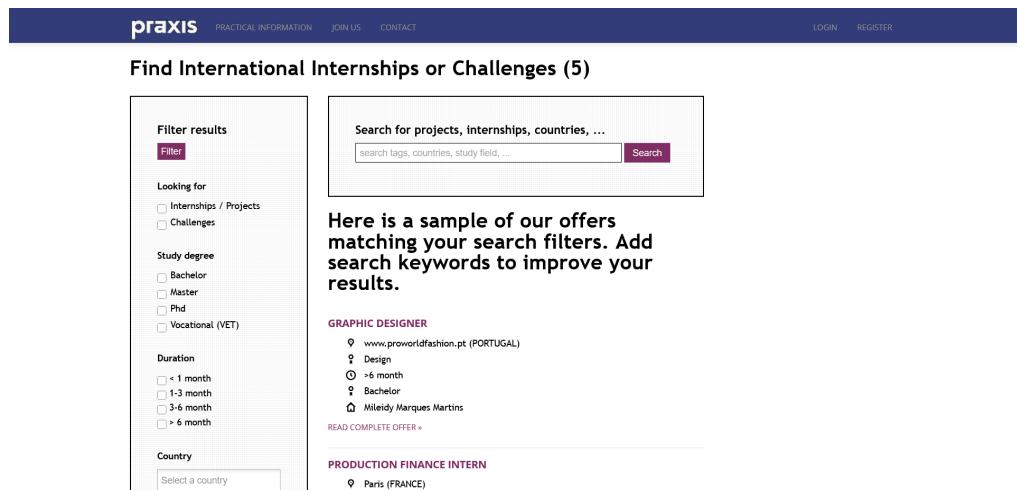


Figura 2.4: Página de pesquisa da PRAXIS

2.4 Spain Internship

A *Spain Internship* é uma organização dedicada à oferta de estágios internacionais, com enfoque particular no território espanhol, mas aberta a candidatos de toda a

Europa. Tal como as referidas anteriormente, esta plataforma funciona como mediadora entre estudantes e empresas, fornecendo serviços de recrutamento, acompanhamento e suporte administrativo, sendo frequentemente utilizada por estudantes em programas de mobilidade europeia.

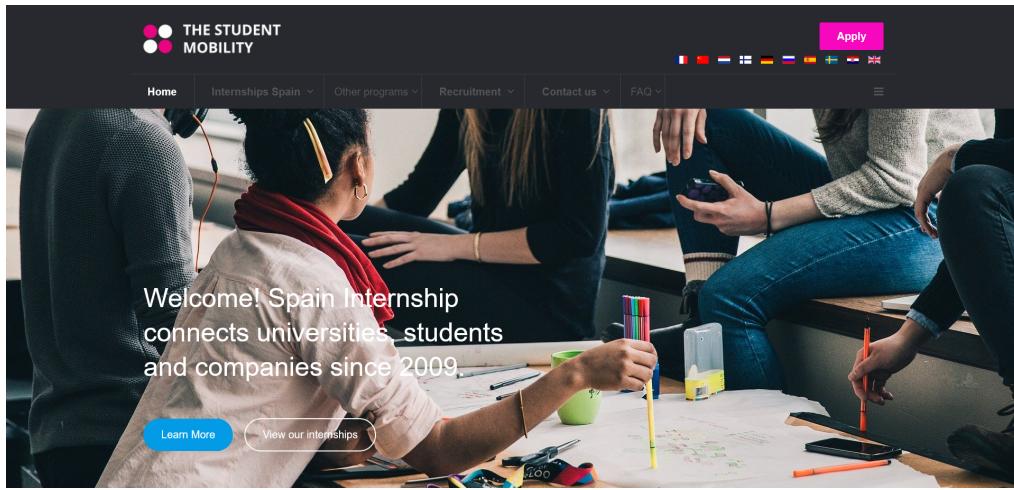


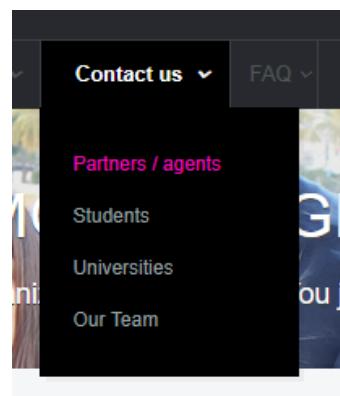
Figura 2.5: *Home Page* da *Spain Internship*

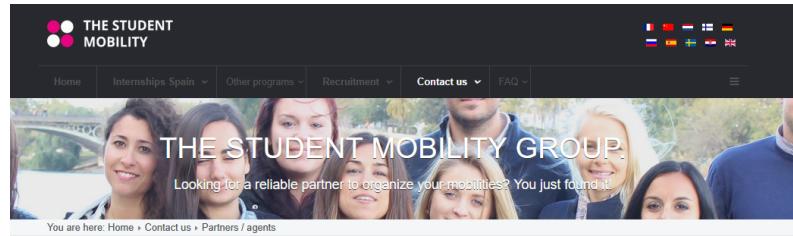
Mais uma vez, observa-se um forte investimento em ferramentas de filtragem, concebidas para tornar a pesquisa de oportunidades o mais simples e direcionada possível. Paralelamente, estas plataformas disponibilizam mecanismos de contacto direto que facilitam a comunicação entre os três principais atores do ecossistema — estudantes, empresas e universidades —, precisamente os mesmos que constituem o núcleo do modelo de negócio do Blended4Future, como referido anteriormente.

The screenshot shows the homepage of THE STUDENT MOBILITY website. At the top, there is a navigation bar with links for Home, Internships Spain (selected), Other programs, Recruitment, Contact us, FAQ, and a language dropdown showing flags for various countries. A prominent pink 'Apply' button is located in the top right corner. Below the navigation, a breadcrumb trail indicates the user's location: You are here: Home > Internships Spain > IT, Engineering & Design Internships > Super User. The main content area features a search form titled 'FIND AN INTERNSHIP' with fields for Keyword, Reference number, Field, Stipend, Other benefits, City, Country, and Cost. To the right of the search form are four internship listings:

- Paid IT Internship in an accommodation services company in Dublin**: Our collaborator is a short and long term accommodation services specialist. Founded in 2018 in Dublin with the aim of providing students and tourists with reliable and value for...
IT, Engineering & Design | 300€ - 400€ | --- | Other | Ireland | Free | Thursday, 28 December 2024
- Paid technical support internship in an enterprise mobility platform in Italy (Italian or Chinese speakers)**: Our collaborator is a leading enterprise mobility platform-as-a-service, empowering businesses across delivery, rental, sharing, and more to streamline their last-mile operations. They partner with a diverse range of clients, from...
IT, Engineering & Design | 500€ + | --- | Firenze | Italy | Free | Tuesday, 17 September 2024
- Architecture Internship in an Architecture and Interior Design Studio in Barcelona**: Our collaborator is an Architecture Studio developing mostly interior design and refurbishing in Barcelona and surroundings. They strive to achieve ethical architecture and believe in crafting spaces that bring harmony...
IT, Engineering & Design | Unpaid | --- | Barcelona | Spain | Free | Wednesday, 07 August 2024
- Remote Full Stack Developer position in a Software Development Agency**: Our client is a software development agency headquartered in Lisbon, Portugal. They specialize in collaborating with a diverse range of clients, particularly within the investment and ecommerce industries. They are ...
IT, Engineering & Design | 500€ + | --- | Remote | Others | Free

Below the listings, a small image of a person in a striped shirt is visible.

Figura 2.6: Página de pesquisa da *Spain Internship*Figura 2.7: *Dropdown list* para as paginas de contacto



THE STUDENT MOBILITY GROUP. PARTNERS AREA.

Looking for a reliable partner to organize your mobilities? You just found it!

Founded in 2009, Spain Internship, a brand of The Student Mobility group (www.internships-germany.com, www.internships-portugal.com, www.internships-italy.com) is an expert provider in organising internship programs for companies, universities and vocational schools as well as many other student services in different locations. We have long-term partnerships with more than 700 educational institutions providing complete internship services for students from all around the world.



Figura 2.8: Página de contacto de *partners*

Capítulo 3

Analise do problema e desenho de uma solução

3.1 O Dominio

Este projeto teve como objetivo principal o desenvolvimento de uma plataforma ajustada às necessidades da organização e aos desafios por ela apresentados. Para tal, tornou-se essencial definir e estruturar cuidadosamente a lógica de negócio subjacente.

Após várias sessões de discussão com os professores envolvidos, foi possível clarificar o domínio do problema e estabelecer uma visão mais sólida sobre a solução pretendida. Ao longo dos meses, essa visão foi sendo progressivamente aperfeiçoada, permitindo alinhar melhor os requisitos com os objetivos do projeto.

A figura 3.1 refere-se ao diagrama finalizado que foi definido.

3.2 Engenharia de requisitos

A Engenharia de Requisitos é uma área da Engenharia de Software que se dedica à identificação, análise, especificação, validação e gestão das necessidades e expectativas das partes interessadas relativamente a um sistema, com o objetivo de transformar essas necessidades, muitas vezes vagas ou incompletas, em requisitos claros, compreensíveis e verificáveis.

Esta secção dedica-se à apresentação dos requisitos estabelecidos para o sistema no arranque do projeto. Para uma melhor organização, estes requisitos foram divididos em dois grupos distintos: funcionais, que descrevem as operações fundamentais que a aplicação deve garantir, e não funcionais, que representam as condições e restrições que asseguram o correto desempenho dessas operações.

3.2.1 Requisitos funcionais

Durante os primeiros dois sprints, em conjunto dos professores, foi estabelecido um *backlog* de *user stories* que refletia a experiência desejada dos atores.

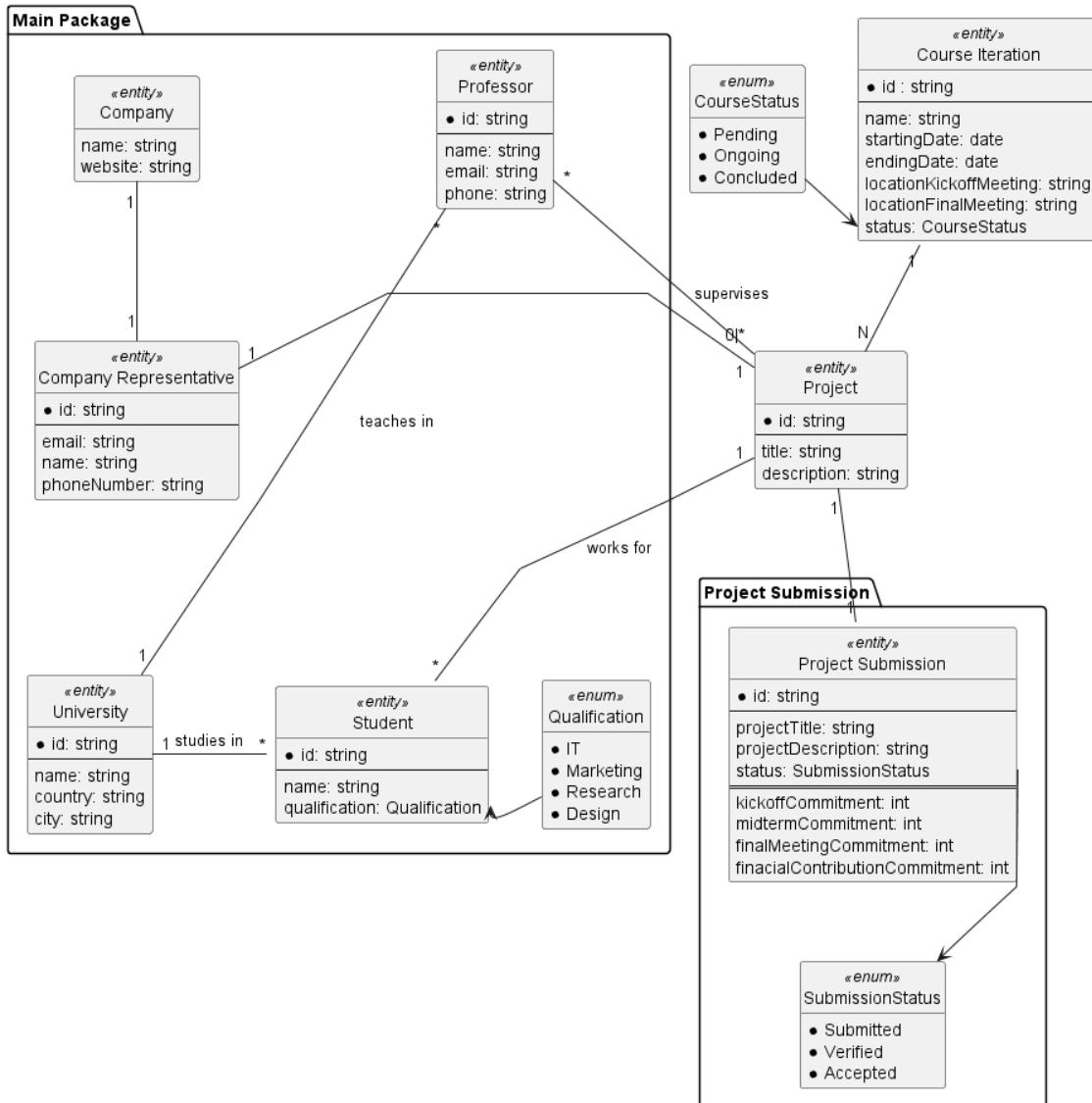


Figura 3.1: Diagrama de classes da lógica de negócio do Blended4Future 1

A tabela 3.1 reflete esta decisão.

3.2.1.1 Diagram de User Flow

Um Diagrama de *User Flow* é uma representação visual que descreve o caminho que um utilizador percorre dentro de um sistema ou aplicação para atingir um objetivo específico. Para uma melhor experiência de desenvolvimento, um destes foi elaborado. Este pode ser encontrado no anexo A.

3.2.2 Requisitos não Funcionais

3.2.2.1 FURPS

Para organizar e clarificar os requisitos do sistema, recorreu-se ao modelo **FURPS**, que permite classificar os requisitos em cinco categorias:

Functionality (Funcionalidade)

- Funcionalidades de backoffice e frontoffice asseguradas com a respetiva autenticação.

Usability (Usabilidade)

- Interface intuitiva e consistente, que siga as normas de design.
- Navegação simples, permitindo localizar facilmente projetos e conteúdos.
- Feedback visual claro sobre ações realizadas (erros, confirmações).

Reliability (Confiabilidade)

- Garantir que o programa se mantém ativo e disponível, mesmo na presença de erros e outras adversidades.
- Assegurar que novas versões são disponibilizadas sem prejudicar a persistência de informação na base de dados.

Performance (Desempenho)

- Resposta rápida da interface mesmo com múltiplos utilizadores.

Supportability (Suportabilidade)

- Código modular e documentado para facilitar manutenção futura.
- Facilita a adição de novas funcionalidades ou integração com outras ferramentas.

Algumas destes requisitos foram considerados como casos de uso do programa e adicionados à tabela referida no ponto 3.2.1 de forma a facilitar o processo de desenvolvimento. A tabela 3.2 subscreve esta decisão.

User Story	Título
US2	Como Representante da Empresa, quero alterar facilmente os detalhes da minha empresa (logótipo, contactos, etc.) para que represente melhor a sua imagem
US3	Como Representante da Empresa, quero submeter facilmente um novo projeto
US4	Como Empresa, quero relatórios com os resultados dos meus projetos.
US6	Como Professor, quero poder aceitar novas ideias de projeto
US7	Como Professor, quero poder selecionar um estudante e ver mais detalhes sobre ele (área de especialização, experiência profissional, LinkedIn)
US8	Como Professor, quero enviar convites a estudantes para participarem em projetos de acordo com as suas qualificações e interesses
US9	Como Estudante, gostaria de saber quais as qualificações necessárias para me inscrever num projeto
US11	Como Representante da Universidade, gostaria de poder contactar a Blended para aderir ao programa
US12	Como Utilizador, quero poder pesquisar projetos com base em critérios específicos
US13	Como Utilizador, quero receber uma confirmação por email após o registo para saber que a minha conta foi criada com sucesso
US14	Como Utilizador, quero ver uma página inicial
US15	Como Utilizador, gostaria de iniciar sessão na minha conta, para poder aceder às minhas informações pessoais
US17	Como Administrador, quero adicionar novas empresas ao sistema com o respetivo site e logótipo, para que a sua parceria seja visível na plataforma
US18	Como Administrador, quero poder suspender ou desativar contas de utilizadores
US19	Como Administrador, quero poder adicionar e gerir as universidades no sistema
US20	Como Administrador, quero poder adicionar novos utilizadores ao sistema
US21	Como Administrador, quero eliminar publicações no blogue para que informação desatualizada ou incorreta possa ser removida
US22	Como Administrador, quero receber alertas caso um projeto esteja inativo durante demasiado tempo para poder acompanhar os participantes

User Story	Título
US23	Como Administrador, quero agendar publicações no blogue com antecedência para que o conteúdo seja publicado no momento certo
US24	Como Professor, quero criar publicações no blogue para que anúncios e ideias possam ser partilhados com os visitantes
US25	Como Professor, quero editar as minhas publicações no blogue para que informação desatualizada ou incorreta possa ser atualizada
US26	Como Professor, quero carregar e gerir fotografias de grupo e testemunhos de cada projeto para que os visitantes possam ver imagens e feedback relevantes
US27	Como Blended4Future, quero avaliar como os estudantes evoluíram ao longo de todo o projeto
US28	Como Utilizador, quero poder ver uma página de apresentação para empresas (elevator pitch)
US29	Como Utilizador, quero poder ver uma página de apresentação para estudantes (elevator pitch)
US30	Como Utilizador, quero poder ver uma página de apresentação para universidades (elevator pitch)
US31	Como Empresa, quero compreender melhor a identidade visual do conteúdo que quero apresentar
US32	Como Empresa, quero ter uma boa presença de antigos e atuais alunos nas redes sociais
US33	Como Empresa, quero ter publicações automáticas nas redes sociais, para que a sua presença online seja consistente
US34	Como Empresa, quero compreender que tipo de conteúdo quero apresentar nas nossas plataformas de redes sociais
US35	Como Empresa, quero compreender melhor o impacto da Blended em todas as partes interessadas (estudantes, universidades, empresas)
US36	Como Empresa, quero seguir as orientações para encontrar um novo nome para o projeto Blended4Future
US37	Como Empresa, quero ter uma Análise SWOT para o Blended4Future, de modo a compreender o seu valor e o seu mercado

Tabela 3.1: Lista de requisitos funcionais

User Case	Title
UC1	Como Administrador, quero ter um sistema automatizado para implementar toda a solução
UC5	Como Programador, quero documentação que descreva toda a solução de forma abrangente
UC10	Como Sistema, quero que os projetos passem automaticamente de "em curso" para "concluídos" com base nas datas de início e fim, para que os estados se mantenham atualizados
UC16	Como Administrador, quero que o sistema associe automaticamente os estudantes a um projeto específico
UC38	Como Equipa de Desenvolvimento, quero ter pipelines configurados para o deployment automático no website, tanto para o backend como para o frontend
UC39	Como Sistema, quero notificar automaticamente os utilizadores sobre prazos dos projetos para que se mantenham informados

Tabela 3.2: Lista de requisitos não funcionais

3.3 Arquitetura do sistema

3.3.1 Solução

No desenvolvimento desta solução, revela-se fundamental a organização do sistema em dois módulos distintos: um módulo responsável pela gestão da lógica de negócios, encarregada da execução das regras, validações e persistência de dados (*Backend*); e um módulo de apresentação, dedicada à visualização e interação com a informação (*Frontend*).

3.3.2 Nível 1 - Contexto

3.3.2.1 Diagrama Lógico

Na figura 3.2 encontra-se o diagrama lógico de nível. Este mostra os componentes desta arquitetura com um alto nível de abstração. Apenas é possível ver a exposição de 3 tipos de página web para vários tipos de utilizadores: não registado, registado e administradores.

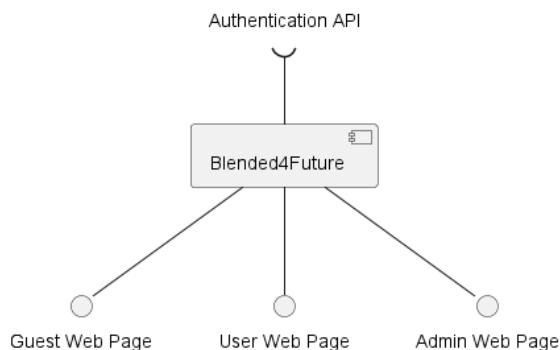


Figura 3.2: Diagrama Lógico de Nível 1

3.3.2.2 Diagrama Físico

A figura 3.3 demonstra o diagrama de físico de nível 1. Utiliza-se o *port 80* pois este é por norma utilizado em chamadas HTTP por este um protocolo baseado em TCP

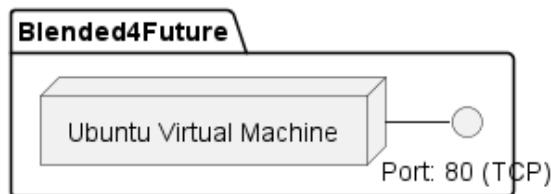


Figura 3.3: Diagrama Físico de Nível 1

3.3.3 Nível 2

3.3.3.1 Diagrama Lógico

Na figura 3.4 é possível ver o anteriormente desmosntrado na figura 3.2 com um menor nível de abstração. A este somam-se os conceitos de *Frontend* e *Backend*.

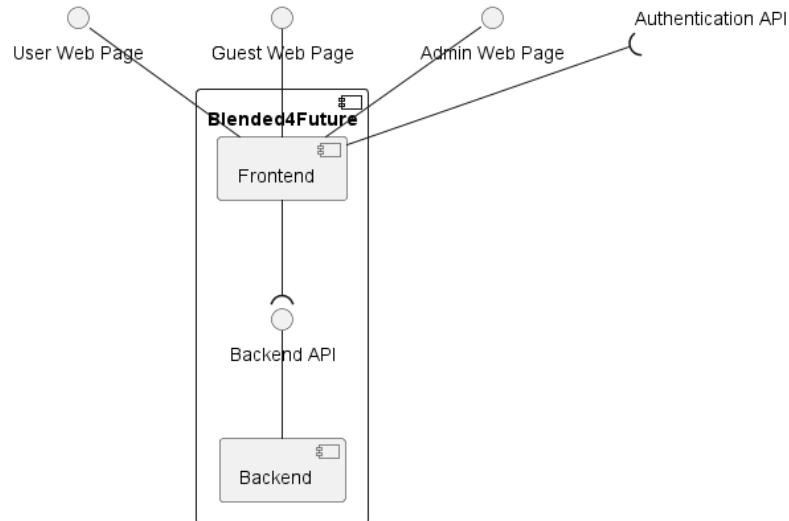


Figura 3.4: Diagrama Lógico de Nível 2

3.3.3.2 Diagrama Físico

A figura 3.5 demonstra o diagrama de físico de nível 2. Neste entende-se a função do servidor *NGINX* como a de criação de duas rotas para *Backend* e *Frontend*, respectivamente '/api' e '/'.

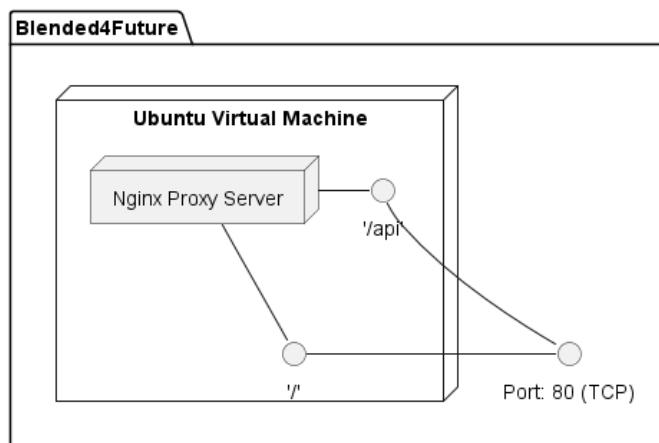


Figura 3.5: Diagrama Físico de Nível 2

3.3.4 Nível 3

3.3.4.1 Diagrama Lógico

Com um nível menor de abstração, a figura 3.6 apresenta aprofunda os modulos de *Backend* e *Frontend*. O primeiro é responsável pela logica de negócio e persistencia e, por tal, é detentor de uma aplicação Spring e de uma base de dados relacional MySQL.

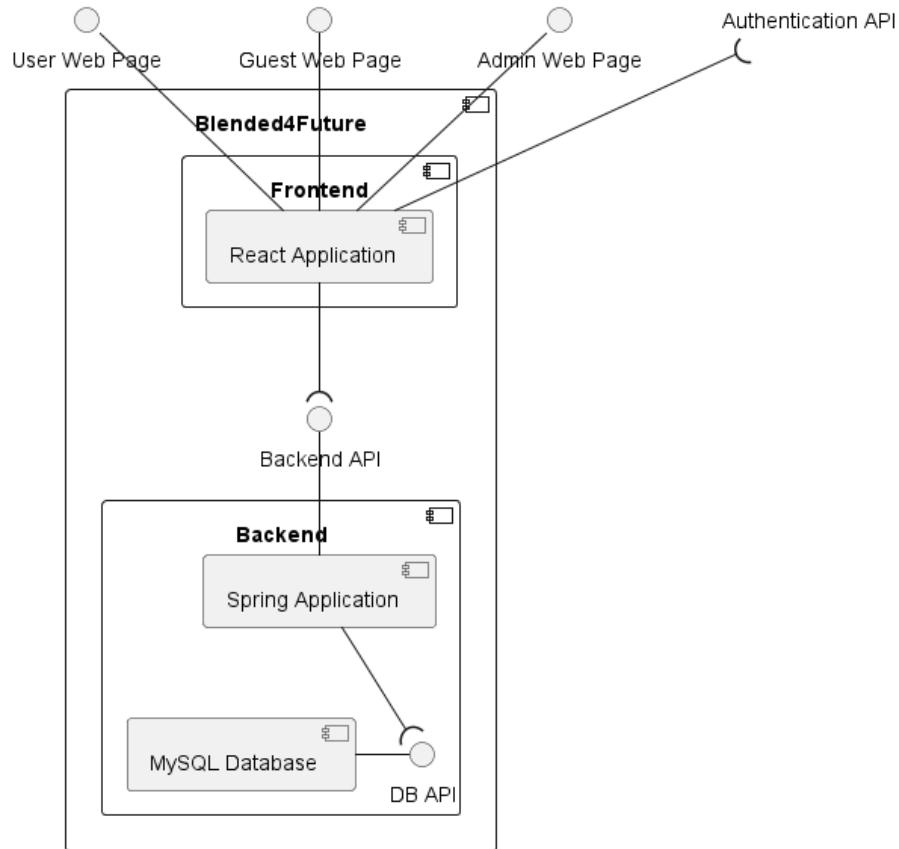


Figura 3.6: Diagrama Lógico de Nível 3

3.3.4.2 Diagrama Físico

A figura 3.7 demonstra o diagrama de físico de nível 3. Com este é possível verificar o mapeamento das duas rotas criadas pelo servidor proxy *NGINX* para os PORTS 3000 e 4000, escolhidos, pois são memoráveis. Foi feita ainda a escolha de encapsular as aplicações *Backend* e *Frontend* em imagens *docker* garantindo assim o seu funcionamento independentemente do *hardware* escolhido.

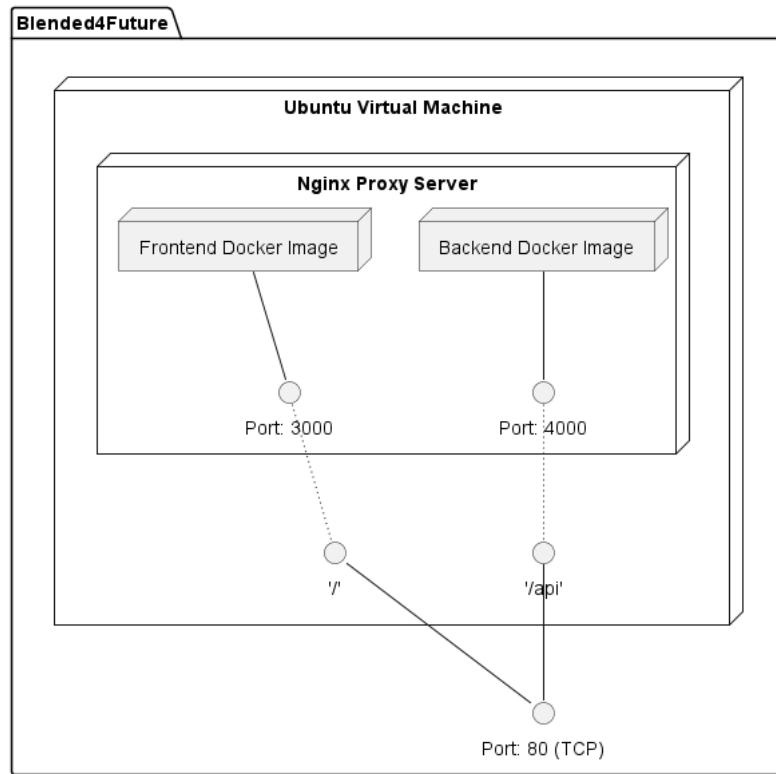


Figura 3.7: Diagrama Físico de Nível 3

3.3.5 Nível 4

3.3.5.1 Diagrama Físico

A figura 3.8 demonstra o diagrama de físico de nível 4. Dentro das respetivas imagens *docker* é possível as aplicações *Backend* e *Frontend* encapsuladas. No contexto do *docker* é necessário fazer um mapeamento dos *ports* utilizados do ambiente interno (Imagem) para o externo (Máquina Virtual), por isso é possível ver o mapeamento dos ports, respetivamente, internos e externos, 8080 e 4000, no *Backend* e 3000 e 3000 no *Frontend*.

Em nota, é possível incluir a base de dados na imagem graças à funcionalidade *Volumes* do Docker e esta pode persistir diretamente no *hardware* da máquina virtual.

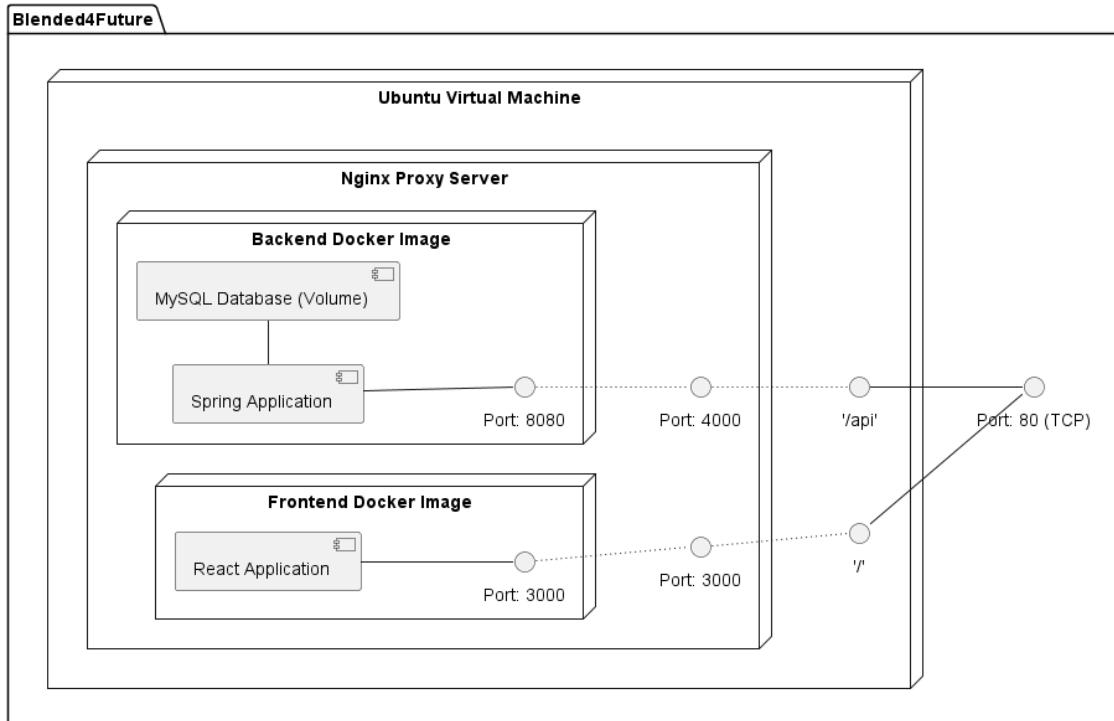


Figura 3.8: Diagrama Físico de Nível 4

Capítulo 4

Implementação de uma solução

4.1 A Implementação

O presente capítulo tem como objetivo detalhar a implementação de funcionalidades específicas, sustentadas na arquitetura previamente apresentada. Serão abordados os diferentes módulos de *frontend* e *backend*, bem como as *pipelines* de *deployment* necessárias para assegurar a integração contínua e a entrega eficiente do sistema.

4.1.1 Backend

O desenvolvimento do *backend* constituiu uma das partes fundamentais do projeto, assegurando a implementação da lógica de negócio, a gestão dos dados e a comunicação entre o sistema e a interface de utilizador. Este componente atua como a camada central responsável por garantir que os processos internos sejam executados de forma consistente, segura e eficiente, proporcionando suporte às funcionalidades disponibilizadas no *frontend*.

Nesta secção irá se apresentar funcionalidades do *backend* no contexto da US3 (definida na tabela 3.1).

4.1.1.1 Spring

A framework Spring apresentou à equipa uma alta curva de aprendizagem. Os vários conceitos e ferramentas nesta são bastante alienígenas a qualquer outra ferramenta antes utilizada pelos membros. Por tal foi necessário mais tempo para a aprender. Alguns dos conceitos-chave considerados incluem:

- **Spring Data** é um agregado de módulos Spring que têm como função principal facilitar a programação de entidades e o seu respetivos acesso quando conectado a uma fonte de dados.
- **Spring Data JPA** (ou só *JPA*) é um dos módulos pertencente à coletânea Spring Data. O seu objetivo é facilitar a implementação de repositórios, reduzindo estes a interfaces Java, nas quais o Spring analisa o nome do método e implementa em *runtime* o mesmo.

- **Hibernate** [2] é uma framework Java e uma solução ORM que serve como implementação do *JPA* para, logicamente, persistir a informação na respetiva base de dados.
- **IOC** (Inversion of Control, em português *Inversão de Controlo*) é um princípio de engenharia de *software* que transfere a responsabilidade pela criação e gestão dos objetos para uma *framework* específica. No *Spring*, esta funcionalidade é desempenhada pelo denominado *IOC Container*.
- **Bean** corresponde a uma instância de objeto gerida pelo *IOC Container*. Cada *Bean* é criado, configurado e mantido pelo próprio contendor, segundo a configuração definida.
- **DI** (Dependency Injection, em português *Injecção de Dependências*) é um princípio amplamente utilizado no desenvolvimento de *software* que visa reduzir o acoplamento entre classes. No contexto do *Spring*, esta prática é suportada através do *IoC Container*, que fornece e gere as instâncias necessárias sob a forma de *Beans*.

4.1.1.2 Estrutura de pastas

A estrutura de pastas do backend foi definida para enaltecer a função de cada ficheiro e/ou classe em cada uma, como pode ser observado na figura 4.1.

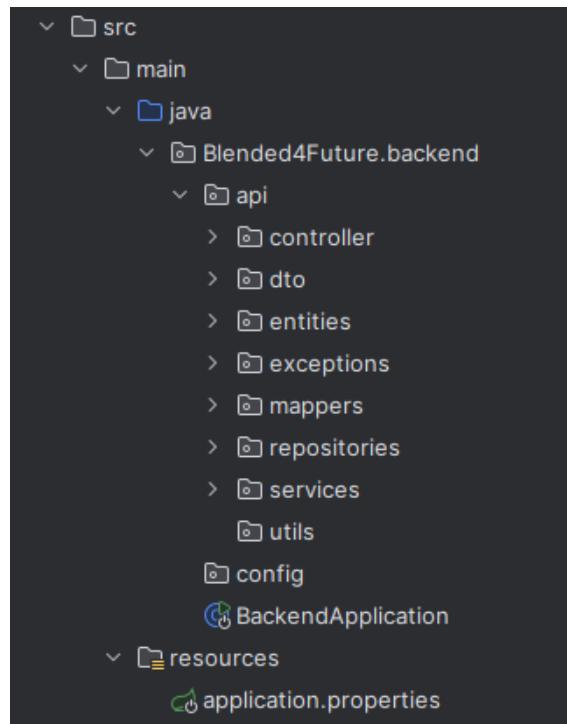


Figura 4.1: Estrutura de pastas do backend

4.1.1.3 Entidades e relações

Para o caso de uso a ser considerado, a entidade de maior importância será *Project*. Esta encontra-se representada na figura 4.1.

Em nota, interessante enaltecer o uso das respectivas anotações:

- **@Entity** informa o JPA/Hibernate que esta é uma entidade que deve ser persistida. Para tal deve conter um argumento anotado **@Id**.
 - **@Data**[1] serve como substituto para as anotações **@ToString**, **@EqualsAndHashCode**, **@Getter**, **@Setter** e **@RequiredArgsConstructor** que, respetivamente:
 - implementa o método `toString()` incluindo todos os parâmetros não estáticos da classe;
 - implementa o método `equals(Object object)` e `hashCode()`. Uma definição (`callSuper`) teve de ser subscrita, pois era desejado inclusão dos atributos da superclasse `BaseEntity` neste método (ver secção 4.1.1.4);
 - implementa métodos *getter* para todos os atributos privados
 - implementa métodos *setter* para todos os atributos privados
 - implementa um construtor com um parâmetro por atributo não final da classe.

```
1  @Entity
2  @Data
3  @EqualsAndHashCode(callSuper = true)
4  public class Project extends BaseEntity {
5
6      @Column(nullable = false)
7      @Convert(converter = ProjectName.NameConverter.class)
8      private ProjectName name = new ProjectName();
9
10     @Column(nullable = false)
11     @Convert(converter =
12         ProjectDescription.DescriptionConverter.class)
13     private ProjectDescription description = new ProjectDescription();
14
15     @ManyToOne
16     private Company company;
17
18     @ManyToOne
19     private CompanyRepresentative companyRepresentative;
20
21     @ManyToMany
22     private Set<Student> students = Set.of();
23
24     @ManyToMany
25     private Set<Qualification> qualifications = Set.of();
26
27     @OneToMany(mappedBy = "project", cascade = CascadeType.REMOVE)
```

```

27     private Set<Report> reports = Set.of();
28
29     @ManyToOne
30     private CourseEdition courseEdition;
31 }
```

Listing 4.1: Classe *Project*

4.1.1.4 BaseEntity

Como medida de segurança e de padronização, foi desenvolvida a classe *BaseEntity* (ver listagem 4.2). Esta classe define dois identificadores: um interno e um externo. O identificador externo é utilizado na comunicação com o cliente, sendo retornado nos pedidos sob a forma de *DTO*, enquanto o identificador interno é reservado para uso exclusivo do sistema.

Tal como o próprio nome indica, a *BaseEntity* foi concebida para servir como *superclasse* de todas as entidades persistentes.

A implementação da *BaseEntity* envolve várias decisões de *design* que merecem atenção:

- No *Hibernate*, os atributos definidos em superclasses não são, por defeito, persistidos. A utilização da anotação `@MappedSuperclass` assegura que as classes filhas possam herdar esses parâmetros, permitindo a sua correta persistência na base de dados.
- A anotação `@Id` define o atributo `iid` como identificador principal da entidade no contexto da persistência. Complementarmente, `@GeneratedValue(strategy=GenerationType.UUID)` especifica a forma como o identificador deve ser gerado, recorrendo ao padrão *UUID*. Esta opção é particularmente relevante dado que, por omissão, o *Hibernate* apenas consegue popular atributos de tipos primitivos.
- Verificou-se a necessidade de avaliar, de forma global, todos os atributos relacionados com a lógica de negócio, de modo a identificar e retornar eventuais erros existentes. Para esse fim, a função `isBusinessDataValid()` analisa todos os *Value Objects* (Ver secção 4.1.1.5) da respetiva classe e devolve um `HashMap` contendo a listagem dos erros detetados.

```

1  @MappedSuperclass
2  @Getter
3  public abstract class BaseEntity {
4
5      @Id
6      @GeneratedValue(strategy = GenerationType.UUID)
7      private UUID iid;
8
9      @Column(name="eid", unique = true, nullable = false)
10     private UUID externalId;
11 }
```

```

12     public BaseEntity() {
13         this.externalId = generateCode();
14     }
15
16     private UUID generateCode() {
17         return UUID.randomUUID();
18     }
19
20     public HashMap<String, Object> isBusinessDataValid() {
21
22         HashMap<String, Object> errors = new HashMap<>();
23
24         Class<?> clazz = this.getClass();
25         List<IValueObject<?>> valueObjects = new ArrayList<>();
26
27         while (clazz != null && clazz != Object.class) {
28             for (Field field : clazz.getDeclaredFields()) {
29                 field.setAccessible(true);
30                 try {
31                     Object value = field.get(this);
32
33                     if (value instanceof IValueObject<?> vo) {
34                         valueObjects.add(vo);
35                     }
36                 } catch (IllegalAccessException ignored) {}
37             }
38             clazz = clazz.getSuperclass();
39         }
40
41         return isBusinessDataValid(valueObjects);
42     }
43
44     public static HashMap<String, Object>
45         isBusinessDataValid(List<IValueObject<?>> fields) {
46         HashMap<String, Object> errors = new HashMap<>();
47         for (IValueObject<?> field : fields) {
48             if (field == null) {
49                 continue;
50             }
51             errors.putAll(field.isValid());
52         }
53         return errors;
54     }

```

Listing 4.2: Classe *BaseEntity*

4.1.1.5 Value Objects

A utilização de *Value Objects* permite um maior controlo sobre a lógica de negócio inherente ao projeto. Por tal a sua implementação é necessária aquando a construção de um sistema de superior complexidade.

A listagem 4.3 mostra a interface ValueObject.

```

1  public interface IValueObject<T> {
2      T getValue();
3      public HashMap<String, Object> isValid();
4 }
```

Listing 4.3: Interface Value Object

A notar, a função `isValid()` em conjunto com a superclasse `BaseEntity` permite verificar a lógica de negócio de cada respetivo `ValueObject`. Para tal basta implementação fazer uma verificação do objecto em questão e retornar um `HashMap` com toda a informação de erros.

```

1 @Value
2 public class ProjectDescription implements IValueObject<String> {
3
4     public static final int MIN_LENGTH = 10;
5     public static final int MAX_LENGTH = 500;
6     public static final String DEFAULT_DESCRIPTION = "Default Project
7 Description";
8
9     private String description;
10
11    @Override
12    public String getValue() {
13        return this.description;
14    }
15
16    public ProjectDescription(String value) {
17        this.description = value;
18    }
19
20    public ProjectDescription() {
21        this(DEFAULT_DESCRIPTION);
22    }
23
24    @Override
25    public HashMap<String, Object> isValid() {
26        HashMap<String, Object> errors = new HashMap<>();
27        if (this.getValue() == null || this.getValue().isEmpty()) {
28            errors.put("description", "Description cannot be null or
empty");
29        } else if (this.getValue().length() < MIN_LENGTH ||
30        this.getValue().length() > MAX_LENGTH) {
31            errors.put("description", "Description must be between 10
and 500 characters long");
32        }
33        return errors;
34    }
35
36    @Converter
```

```

35     public static class DescriptionConverter implements
36         jakarta.persistence.AttributeConverter<ProjectDescription, String>
37     {
38         @Override
39         public String convertToDatabaseColumn(ProjectDescription
40             description) {
41             return description.getValue();
42         }
43         @Override
44         public ProjectDescription convertToEntityAttribute(String
45             dbData) {
46             return new ProjectDescription(dbData);
47         }
48     }

```

Listing 4.4: Classe *ProjectDescription*

A listagem 4.4 demonstra uma implementação de *IValueObject*. Como referido a função *isValid()* avalia se o objeto é valido ou não retornando quaisquer erros.

Importante também seria referir a necessidade de um conversor, uma classe que informa o *Hibernate* como deve mapear o objeto do programa para a base de dados e vice-versa.

4.1.1.6 Repositórios

Como referido na secção 4.1.1.1, o modulo Spring Data JPA, permite a implementação de repositórios em *runtime*.

Tendo em conta a implementação de *BaseEntity* e a necessidade de diferenciação entre identificadores internos e externos, foi implementada a interface *BaseRepository* (ver listagem 4.5).

Na referida interface o uso de *@NoRepositoryBean* informa o *IOC Container* para não guardar uma instância deste repositório, pois por norma todos os *JpaRepository* são guardados automaticamente. É ainda adicionada a função *findByExternalId*, esta procura automaticamente por todas a instâncias que o correspondem ao nome do atributo *externalId*. Esta segue uma nomenclatura específica do Spring [5].

```

1  @NoRepositoryBean
2  public interface BaseRepository<T extends BaseEntity> extends
3      JpaRepository<T, UUID> {
4      Optional<T> findByExternalId(UUID internalId);
5  }

```

Listing 4.5: Inteface *BaseRepository*

Graças ao *Spring* é possível então fazer implementações de repositórios muito facilmente, como exemplificado na listagem 4.6

```

1  public interface ProjectRepository extends
2   BaseRepository<Project> {}

```

Listing 4.6: interface *ProjectRepository*

4.1.1.7 Controladores e prevenção de erros

Na eventualidade da ocorrência de erros, que estes sejam de negócio ou de sistema torna-se necessária implementar um sistema que os consiga detetar e retornar num formato predefinido, algo que o *Spring* não faz por base.

O *Spring* disponibiliza `@RestControllerAdvice` que permite a implementação de rotas ou ferramentas de deteção de erros em todas as classes anotadas com `@RestController`.

Na listagem 4.10 entende-se o elemento base que permite capturar erros internos ou de negócio e retorná-los num formato predefinido (ver listagem 4.8). Em prática a anotação `@ExceptionHandler(AppException.class)` faz com que a qualquer ponto de execução do programa, no caso de ser lançada uma exceção do tipo `AppException` a esta se redirecione para a função `handleAppException`. Na listagem 4.9 e 4.10 mostrase, respetivamente, exemplo do uso desta exceções e propria exceção `AppException`.

```

1  @Slf4j
2  @RestControllerAdvice
3  @RequiredArgsConstructor
4  public class AppExceptionController {
5
6      // Everytime an app exception is thrown, this method will be
7      // called
8      // It will log the error and return a ResponseEntity with the
9      // error details
10     @ResponseBody
11     @ExceptionHandler(AppException.class)
12     public ResponseEntity<?> handleAppException(AppException e,
13         HttpServletRequest request) {
14         log.error("App Exception occurred: {}", e.getMessage(), e);
15         return createResponseError(e, request.getRequestURI());
16     }
17
18     private ResponseEntity<ErrorResponse>
19     createResponseError(AppException e, String path) {
20         Error err = new Error(
21             e.getMessage(),
22             e.getStatus().value(),
23             path,
24             Instant.now(),
25             e.getData()
26         );
27
28         ErrorResponse errorResponse = ErrorResponse.fromError(err);
29
30     }

```

```

26         return new ResponseEntity<>(errorResponse, new HttpHeaders(),
27             e.getStatus());
28     }
29 }
```

Listing 4.7: Class *AppExceptionController*

```

1 @Value
2 @AllArgsConstructor(access = PUBLIC)
3 public class Error {
4     String message;
5     int status;
6     String path;
7     Instant timestamp;
8     Map<String, Object> data;
9 }
```

Listing 4.8: Class *Error*

```

1 @RestController
2 @RequestMapping("/api/project")
3 public class ProjectController {
4
5
6     private final IProjectService service;
7
8     public ProjectController(IProjectService IProjectService) {
9         this.service = IProjectService;
10    }
11
12    @GetMapping("")
13    public List<ProjectDTO> getAllProjects() {
14        return service.getProjects();
15    }
16
17    @GetMapping("/{id}")
18    public ProjectDTO getProjectById(UUID id) {
19        try {
20            return service.getProject(id);
21        } catch (NoSuchElementException e) {
22            throw new AppException(e, HttpStatus.NOT_FOUND);
23        }
24    }
25
26    @PostMapping("")
27    public ProjectDTO addProject(@RequestBody AddProjectDTO project) {
28        try {
29            return service.addProject(project);
30        } catch (NoSuchElementException e) {
31            throw new AppException(e, HttpStatus.NOT_FOUND);
32        } catch (FormDataException e) {
```

```

33             throw new AppException(e, HttpStatus.BAD_REQUEST,
34     e.getErrors());
35 }
36
37 @PutMapping("/{id}")
38 public ProjectDTO updateProject(@PathVariable UUID id,
39 @RequestBody @Valid AddProjectDTO project) {
40     try {
41         return service.updateProject(id, project);
42     } catch (NoSuchElementException e) {
43         throw new AppException(e, HttpStatus.NOT_FOUND);
44     } catch (FormDataException e) {
45         throw new AppException(e, HttpStatus.BAD_REQUEST,
46     e.getErrors());
47     }
48 }
49
50 @DeleteMapping("/{id}")
51 public ProjectDTO deleteProject(@PathVariable UUID id) {
52     try {
53         return service.deleteProject(id);
54     } catch (NoSuchElementException e) {
55         throw new AppException(e, HttpStatus.NOT_FOUND);
56     }
57 }
```

Listing 4.9: Class *ProjectController*

```

1  @Getter
2  public class AppException extends RuntimeException{
3
4      private final HttpStatus status;
5      private Map<String, Object> data;
6
7      public AppException(String message, HttpStatus status,
8          Map<String, Object> data) {
9          super(message);
10         this.status = status;
11         this.data = data;
12     }
13
14     public AppException(Exception e, HttpStatus status, Map<String,
15         Object> data) {
16         super(e.getMessage());
17         this.status = status;
18         this.data = data;
19     }
20
21     public AppException(Exception e, HttpStatus status) {
22         super(e.getMessage());
23         this.status = status;
```

```

22         this.data = new HashMap<>();
23     }
24
25     public AppException(String message, HttpStatus status) {
26         super(message);
27         this.status = status;
28         this.data = new HashMap<>();
29     }
30 }
```

Listing 4.10: Exceção *AppException*

4.1.1.8 Serviços

O padrão de *software Service* entende a concentração de toda a lógica de negócio numa só camada do sistema. A listagem 4.11 apresenta a implementação do método `addProject` nesta mesma camada.

A notação `@Service` permite ao IOC *container* identificar esta classe como serviço e guardá-la como *Bean*. Já a notação `@Slf4j` cria uma ferramenta utilizada para manter registo de qualquer ação no programa.

```

1 @Service
2 @Slf4j
3 public class ProjectService implements IProjectService{
4
5     ...
6
7     @Override
8     public ProjectDTO addProject(AddProjectDTO project) throws
9     FormDataException {
10
11         HashMap<String, Object> errors =
12         checkIdAttributeAddDTO(project);
13
14         ProjectName name = new ProjectName(project.name());
15         ProjectDescription description = new
16         ProjectDescription(project.description());
17
18         errors.putAll(
19             Project.isBusinessDataValid(
20                 new ArrayList<>(List.of(name, description))
21             )
22         );
23         if (!errors.isEmpty()) {
24             throw new FormDataException("Project data is not valid",
25             errors);
26         }
27
28         Company company = project.companyId() == null ?
29             null : Company.findById(project.companyId());
30
31         ProjectDTO result = new ProjectDTO();
32         result.setName(name.value());
33         result.setDescription(description.value());
34         result.setCompanyId(company.id());
35
36         return result;
37     }
38 }
```

```

23     companyRepository.findByExternalId(project.companyId()).get() : null;
24
25     CompanyRepresentative companyRepresentative =
26         project.companyRepresentativeId() == null ?
27             companyRepresentativeRepository.findByExternalId(
28                 project.companyRepresentativeId()
29             ).get() : null;
30
31     List<Student> students = project.studentIds() == null ?
32         List.of() :
33             project.studentIds().stream()
34                 .map(studentRepository::findByExternalId)
35                 .filter(Optional::isPresent)
36                 .map(Optional::get)
37                 .toList();
38
39     CourseEdition courseEdition =
40         project.courseEditionId() == null ?
41             courseEditionRepository.findByExternalId(
42                 project.courseEditionId()
43             ).get() : null;
44
45     Project newProject = new Project();
46
47     newProject.setName(name);
48     newProject.setDescription(description);
49     newProject.setCompany(company);
50     newProject.setCompanyRepresentative(companyRepresentative);
51     newProject.setStudents(new HashSet<>(students));
52     newProject.setCourseEdition(courseEdition);
53
54     Project savedProject = projectRepository.save(newProject);
55     log.info(
56         "Project with eid {} created successfully",
57         savedProject.getExternalId()
58     );
59     return new ProjectMapper().toDTO(savedProject);
60 }
61 ...
62 }
```

Listing 4.11: Implementação do método para criação de um novo projeto

4.1.1.9 Documentação

De forma a assegurar que os atuais e potenciais futuros desenvolvedores pudessem acompanhar e compreender o trabalho previamente realizado, optou-se pela integração do *Swagger*. Esta ferramenta permite a geração automática de documentação da API, disponibilizando-a por uma interface Web centralizada. Para além de facilitar a consulta das funcionalidades já implementadas, o *Swagger* contribui para a padronização da comunicação entre a equipa de desenvolvimento e outros intervenientes, garantindo maior clareza, acessibilidade e manutenção contínua da solução.

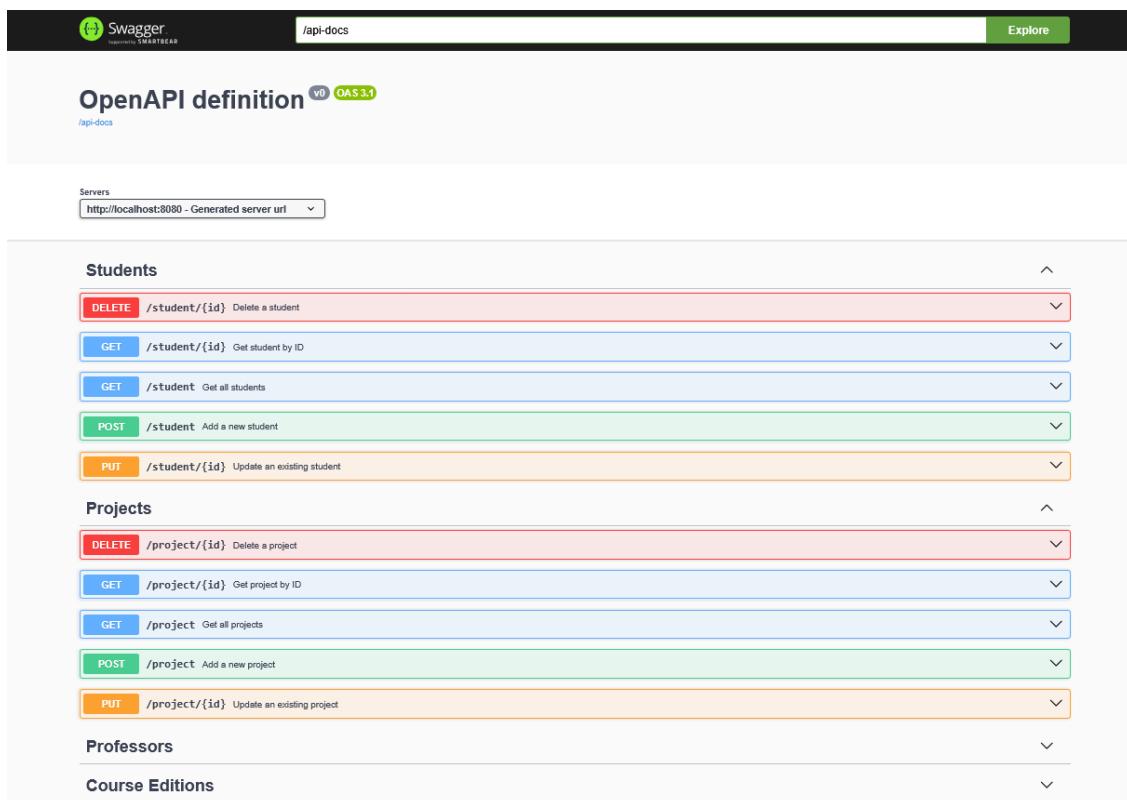


Figura 4.2: *Webpage do Swagger*

4.1.2 Frontend

Como referido anteriormente, o módulo de *Frontend* encontra-se exclusivamente dedicado à visualização e interação com os elementos de negócio, tendo como principal objetivo a implementação de uma interface de utilização simples e intuitiva. Para alcançar este propósito, recorreu-se a ferramentas de design, nomeadamente o *Figma*, que possibilitaram a prototipagem e a definição estruturada da interface antes da sua implementação final.

4.1.2.1 React

O *React* consiste numa biblioteca de *JavaScript* concebida para a criação de *Single Page Applications* (aplicações de página única). A sua arquitetura baseia-se no

conceito de componentes, que correspondem a unidades de código modulares, reutilizáveis e personalizáveis, responsáveis por encapsular a interface, o comportamento e o estilo de forma estruturada.

Um dos aspectos centrais do React é a utilização do *Virtual DOM*, que permite atualizar de forma eficiente a interface em função das alterações de estado, evitando manipulações diretas do DOM real, que são mais custosas em termos de desempenho. Esta abordagem declarativa contribui para a criação de interfaces interativas, escaláveis e de fácil manutenção, adequadas a projetos que exigem dinamismo e modularidade.

4.1.2.2 Páginas Estáticas

Por especificação do cliente existia um importante foco nas páginas estáticas do website. Era exigido a criação de páginas que funcionassem com um *elevator pitch* para cada um dos atores do negócio.

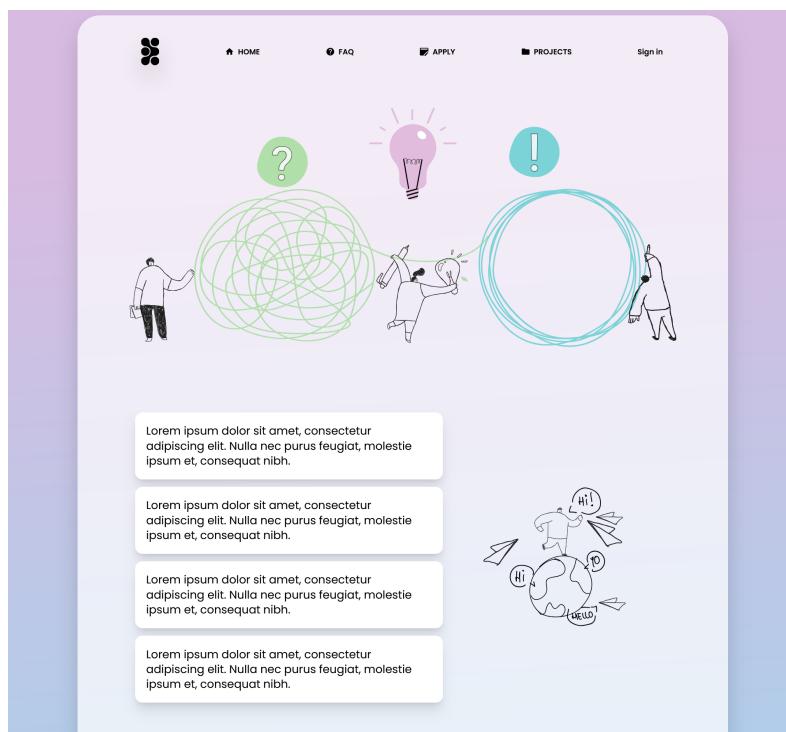


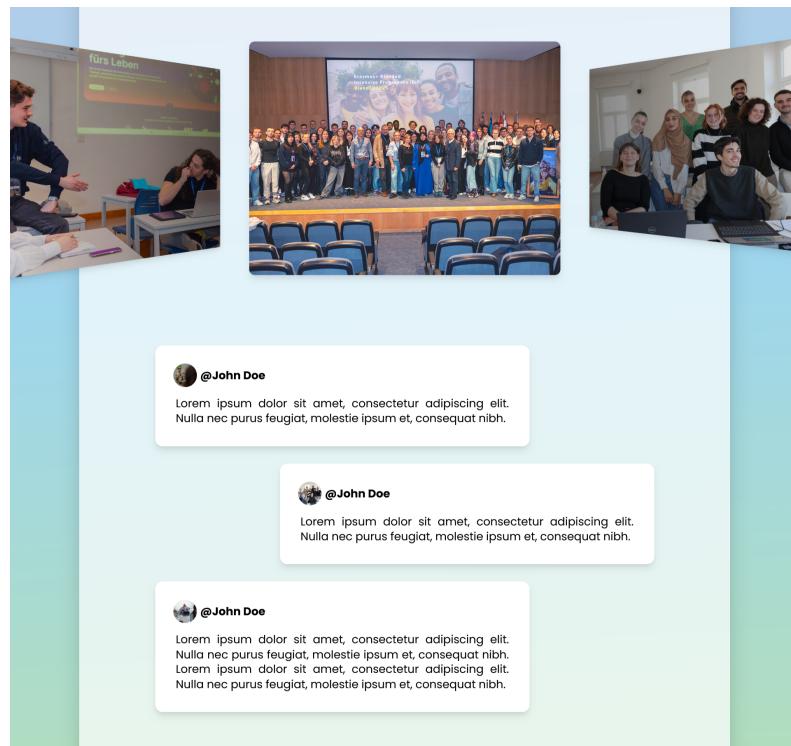
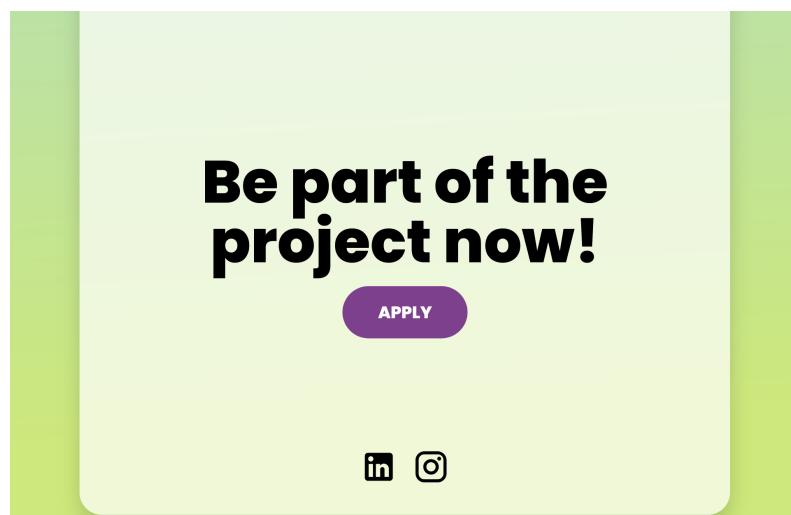
Figura 4.3: Página *Elevator Pitch* de estudante - parte 1

Para manter o efeito vidro + background com gradiente de cores foi implementado um componente *wrapper* para todas as páginas do website. Na listagem 4.12 é possível ver essa implementação.

```

1  interface LandingPageContainerProps extends
2    React.HTMLAttributes<HTMLDivElement> {
3    className?: string;
4    children?: React.ReactNode;
5  }

```

Figura 4.4: Pagina *Elevator Pitch* de estudante - parte 2Figura 4.5: Pagina *Elevator Pitch* de estudante - parte 3

```

5  const LandingPageContainer = (props: LandingPageContainerProps) => {
6    return (
7      <div style={{paddingBlock: "12px", paddingInline: "10px"}}
8        className="w-screen min-h-screen
9          bg-gradient-to-b
10         [background: linear-gradient(177deg ,#D7BBE1_7%,#9ED6ED_45.52%,#CDE87B_95.32%]
11         flex justify-center box-border
12         px-20 py-2.5 overflow-x-hidden"
13       >
14         <div
15           style={{ padding: "15px", borderRadius: "48px", gap:
16             "12px", scrollSnapType: "y mandatory" }}
17           className="w-full sm:w-full lg:w-4xl xl:w-10/12
18             p-10 rounded-28 shadow-2xl
19             bg-white/70 backdrop-blur-3xl box-border
20             snap-y scroll"
21         >
22           {props.children}
23         </div>
24       </div>
25     );
26   };

```

Listing 4.12: Função LandingPageContainer

4.1.2.3 Pagina de Visualização de Projetos

A pagina de visualização de projetos era um dos elementos de maior importancia. Esta permite qualquer visitante do website visualizar os projetos que estão a ser realizados. A listagem 4.13 permite entender a implementação desta pagina. Esta utiliza elementos essenciais do *React*, como useState e useEffect que, respetivamente, permite

```

1  const ProjectLibraryPage = () => {
2    const projectService = new ProjectService();
3    const [projects, setProjects] = useState<Project[]>([]);
4
5    const fetchProjects = async () => {
6      let req = await projectService.getAll();
7      if (req.status > 300) {
8        console.error(req.statusText);
9        return;
10    }
11    setProjects(req.data.projects);
12  };
13
14  useEffect(() => {
15    fetchProjects();
16  }, []);
17
18  return (

```

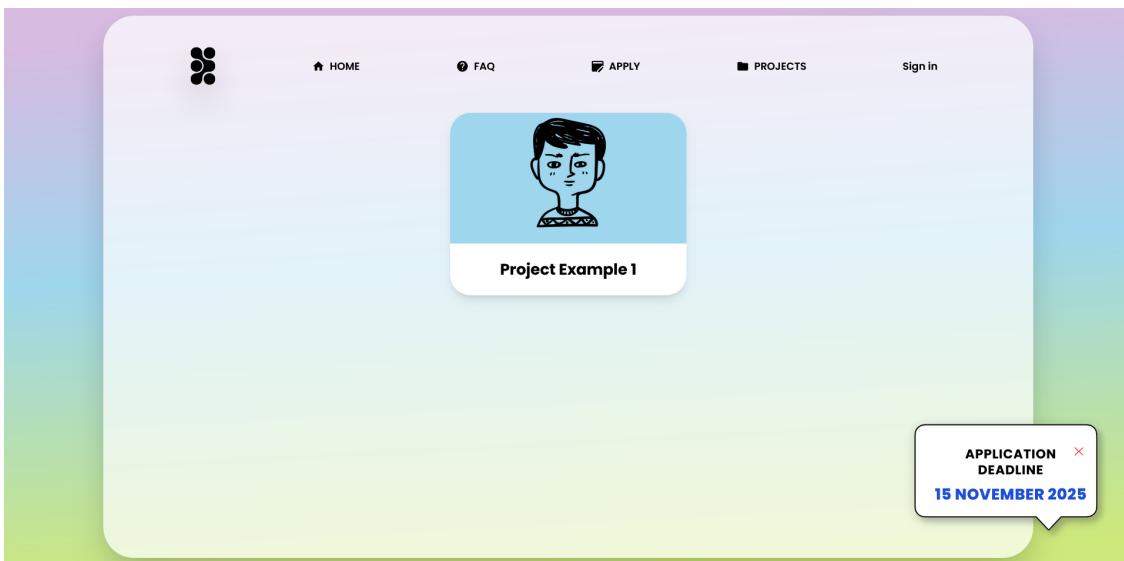


Figura 4.6: Pagina de visualização de projetos

```

19   <LandingPageContainer>
20     <NavbarComponent />
21     <div className="px-1">
22       <div className="flex flex-wrap justify-center w-full gap-8
23         gap-y-8">
24         {projects.map((project) => (
25           <ProjectContainer project={project} key={project.id} />
26         )));
27       </div>
28     </div>
29   </LandingPageContainer>
30 );
  
```

Listing 4.13: Função ProjectLibraryPage

4.1.3 Controlo de Versões

Em todos os repositórios em uso foi definido o uso de múltiplos *branches*, tendo como principais o *main/master* e o *dev*.

Para o desenvolvimento de qualquer nova funcionalidade era necessário a criação de um novo branch baseado na versão mais recente do *dev*. Quando este era finalizado era feito um *merge* devolta no mesmo. Aquando da necessidade de lançar uma nova versão bastava dar merge da versão desejada do *dev* no branch *main*.

4.1.4 Deployments

Como foi explicado anteriormente, um dos requisitos definidos era a criação de um ambiente de *CI/CD* que permitisse simplificar o processo de desenvolvimento,

facilitando a disponibilização de novas versões.

A plataforma Azure disponibiliza a funcionalidade *Pipelines*. Com o recurso à linguagem YAML, esta permite definir os trabalhos que os "agentes" definidos pelo utilizador

Finalmente, utilizando a ferramenta Docker é possível "empacotar" o código numa só unidade que funciona de forma igual em diferentes máquinas. Esta funcionalidade será feita numa máquina Linux Ubuntu 24.04.2.

4.1.4.1 Definição dos agentes

No contexto do Azure Pipelines um agente (em inglês, *Agent*) é a máquina indicada ao Azure Pipelines que executará os trabalhos pedidos, como, por exemplo, compilar o projeto.

Neste caso iremos indicar ao Azure Pipelines para que utilize a nossa máquina como um agente. O agente irá executar dentro de um *container Docker*.

A Microsoft oferece ferramentas para ajudar neste processo [3]. O script que estabelece a relação entre os agentes e os serviços Azure, tal como o Dockerfile necessário para a compilação do Docker poderá ser encontrado nos anexos B e C respetivamente.

Após a criação do agente basta iniciar o *container Docker*, tendo em atenção à indicação das variáveis de ambiente necessárias. A figura 4.7 demonstra isso mesmo, com censura a qualquer informação sensível à organização.

```

1  sudo docker run -d
2    -e AZP_URL="https://dev.azure.com/blended4future2025"
3    -e AZP_TOKEN="*****"
4    -e AZP_POOL="*****"
5    -e AZP_AGENT_NAME="*****"
6    --name "azp-agent-linux"
7    devops-agent:linux

```

Figura 4.7: Script de inicialização do Container Docker para o agente do Azure Pipelines

Logo após, é possível ver o agente disponível no Azure DevOps (Figura 4.8).

4.1.4.2 Trabalhos da pipeline

O conceito de trabalho (em inglês, *Job*) é de extrema importância no Azure Pipelines. Este permite definir ações e os seus respetivos passos para atingir um objetivo.

Nesta secção demonstra-se a *pipeline* criada e os respetivos passos por ela tomados, utilizando como exemplo a definida para o módulo *frontend*. A versão completa deste ficheiro poderá ser encontrada no apêndice D.

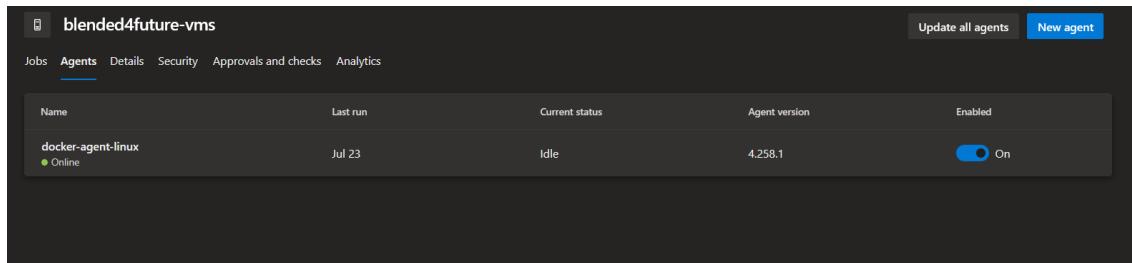


Figura 4.8: Agente disponibilizado no Azure DevOps

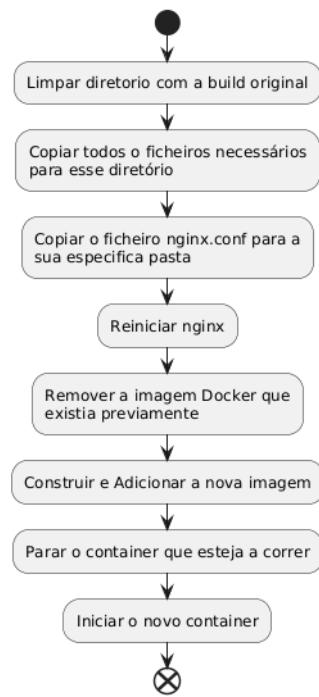


Figura 4.9: Diagrama de atividade representante da Pipeline de construção de uma nova versão

Na figura 4.9, é possível ver o um diagrama de atividade que descreve cada passo feito por este trabalho.

```

1 trigger:
2   branches:
3     include:
4       - main

```

Listing 4.14: Trigger - Pipeline

A listagem 4.14, descreve o gatilho (em inglês, *trigger*) de ativação da pipeline. Neste caso qualquer *commit* no branch "main"irá ativar os trabalhos definidos.

É importante enaltecer que existem variáveis que foram definidas externamente através do Azure DevOps e que são utilizadas ao longo da pipeline. Estas são:

- **CONTAINERNAME**: Nome que o *container* terá quando for criado.
- **FRONTENDBUILDPATH**: Caminho para o qual o projeto será copiado para.
- **VMBUILDPATH**: Caminho para o qual a *build* do projeto irá ser colada em.

Finalmente, cada passo segue o conceito apresentado na figura 4.9. Especial atenção para o uso de `SSH@0[4]`, esta é uma *task* pre disponibilizada para estabelecer uma conexão SSH a uma máquina alvo. A chave está definida como segredo e é importado no início do ficheiro quando ocorre uma referência ao grupo de variáveis "ssh".

4.2 Testes

A prática de testes é fundamental para assegurar o correto funcionamento de qualquer programa, permitindo verificar se o código implementado cumpre as especificações definidas. Esta secção pretende apresentar e detalhar os diferentes tipos de testes realizados ao longo do projeto e as tecnologias utilizadas para este.

4.2.1 Testes Unitários

Os testes unitários constituem uma prática essencial no desenvolvimento de *software*, cujo principal objetivo é garantir que cada unidade de código se comporta conforme esperado. Esta abordagem permite a deteção precoce de erros, aumenta a fiabilidade do *software* e facilita a manutenção e evolução do sistema. Para a execução destes testes, deu-se prioridade à utilização do *JUnit*, em conjunto com as ferramentas disponibilizadas pelo *Spring*.

```

1 @DataJpaTest()
2 class CompanyTest {
3
4   @Autowired
5   private TestEntityManager entityManager;

```

```

6
7     @Autowired
8     private CompanyRepository companyRepository;
9
10    @Test
11    void shouldPersistAndLoadCompanyWithValueObjects() {
12        Company company = new Company();
13        company.setName(new CompanyName("OpenAI"));
14        company.setDescription(new CompanyDescription("Artificial
Intelligence Research"));
15
16        entityManager.persist(company);
17        entityManager.flush();
18        entityManager.clear();
19
20        Optional<Company> found =
21        companyRepository.findById(company.getId());
22
23        assertThat(found.isEmpty()).isFalse();
24        assertThat(found.get().equals(company));
25    }
26
...
26 }
```

Listing 4.15: Class *CompanyTest* - Exemplificação de testes Unitários

4.2.2 Testes de Implementação

Para os testes de implementação, deu-se especial ênfase à utilização da ferramenta Postman, a qual proporciona um ambiente colaborativo e estruturado para a testagem de API.

A figura 4.16 apresenta o script pós-teste executado pelo postman para verificar a resposta ao pedido do tipo POST para "/company".

```

1 pm.test("Status code is 201", function () {
2     pm.response.to.have.status(201);
3 });
4
5 pm.test("Response has correct company data", function () {
6     var jsonData = pm.response.json();
7     pm.expect(jsonData.name.value).to.eql("OpenAI");
8     pm.expect(jsonData.description.value).to.eql("Artificial
Intelligence Research");
9     pm.expect(jsonData.id).to.exist;
10 })
11
12 const companyId = pm.response.json().id;
```

```
13  await pm.sendRequest({
14    url:
15      `${pm.environment.get("localhost")}/company/${companyId}` ,
16    method: 'DELETE',
17    header: {
18      'Content-Type': 'application/json'
19    }
});
```

Listing 4.16: Script de test da rota POST /company

Capítulo 5

Conclusão

O presente capítulo visa sintetizar os principais resultados alcançados ao longo do desenvolvimento do projeto, refletindo sobre os objetivos inicialmente definidos, os desafios enfrentados e as soluções implementadas. Adicionalmente, pretende-se apresentar uma apreciação final da experiência adquirida, destacando aprendizagens relevantes, limitações identificadas e oportunidades de melhoria para projetos futuros. Esta secção constitui, assim, um ponto de reflexão crítica sobre todo o processo de desenvolvimento, integrando tanto a perspetiva técnica como a experiência de trabalho em equipa.

5.1 Desafios

Como referido anteriormente, o desenvolvimento deste projeto enfrentou diversas dificuldades relacionadas com a presença e o cumprimento de tarefas por parte dos restantes elementos da equipa. O trabalho, originalmente planeado para cinco desenvolvedores, foi, na prática, conduzido de forma consistente apenas pelo autor deste relatório. Esta situação revelou-se um obstáculo significativo, dado que muitas tarefas eram interdependentes, resultando em atrasos consideráveis no progresso global do projeto.

5.2 Objetivos Concluídos

No âmbito do trabalho desenvolvido, foi possível implementar um *backend* funcional, com toda a lógica de negócio necessária e uma testagem robusta. Relativamente ao *frontend*, foram desenvolvidas as páginas de visualização de projetos e de *elevator pitch*, direcionadas para os diferentes atores do negócio, assegurando uma experiência consistente e intuitiva.

5.3 Apreciação Final

Apesar dos desafios enfrentados, este projeto proporcionou uma aprendizagem significativa, permitindo ao autor consolidar conhecimentos em ferramentas de desenvolvimento essenciais, como *docker* e *Spring*. Adicionalmente, contribuiu para uma

reflexão aprofundada sobre gestão de equipa, destacando práticas a adotar e a evitar em contextos colaborativos.

Em suma, o autor considera o trabalho desenvolvido gratificante, tendo adquirido competências técnicas e interpessoais que serão valiosas no futuro profissional.

Bibliografia

- [1] *@Data All together now: A shortcut for @ToString, @EqualsAndHashCode, @Getter on all fields, @Setter on all non-final fields, and @RequiredArgsConstructor!* Project Lombok. url: <https://projectlombok.org/features/Data>.
- [2] *Hibernate Documentation.* Commonhaus Foundation. url: <https://hibernate.org/orm/>.
- [3] Microsoft. *Run a self-hosted agent in Docker.* Microsoft. Jan. de 2025. url: <https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/docker?view=azure-devops>.
- [4] Microsoft. *SSH@0 - SSH v0 task.* Microsoft. Jul. de 2025. url: <https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/reference/ssh-v0?view=azure-pipelines>.
- [5] *Working with Spring Data Repositories.* url: <https://docs.spring.io/spring-data/commons/docs/1.6.1.RELEASE/reference/html/repositories.html>.

Apêndice A

Diagrama de User Flow

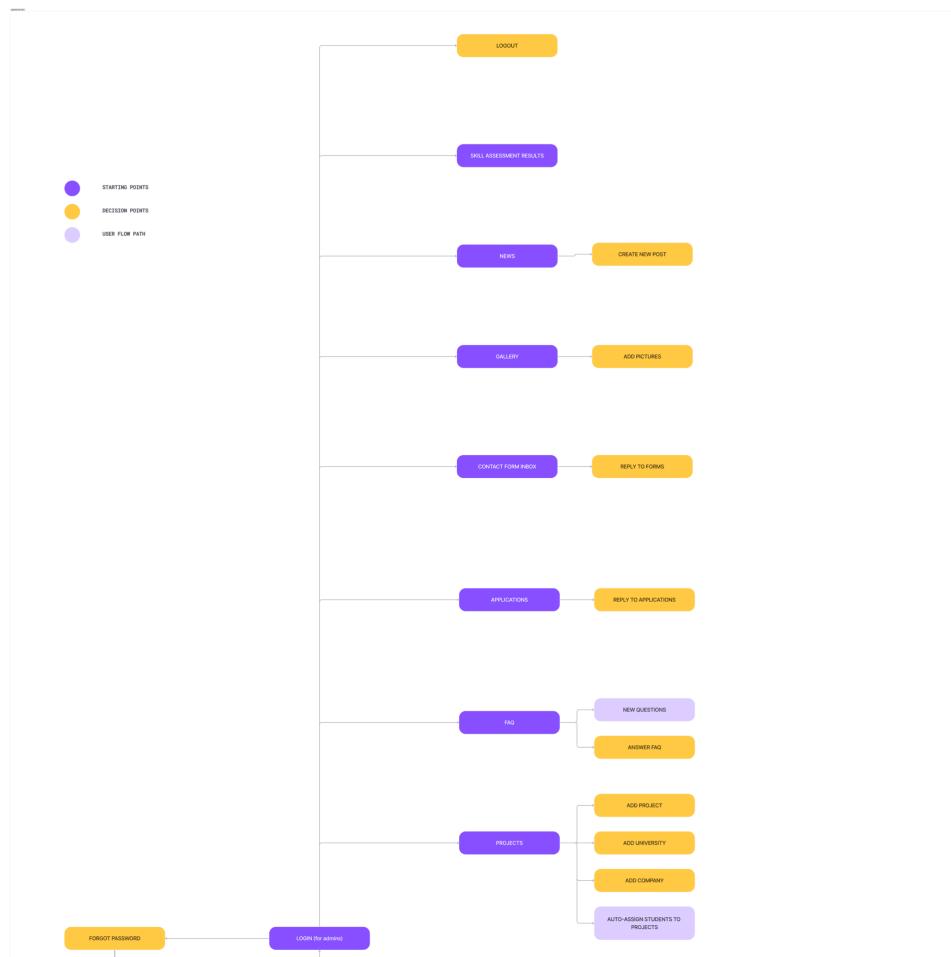


Figura A.1: Diagrama de User Flow da Aplicação 1

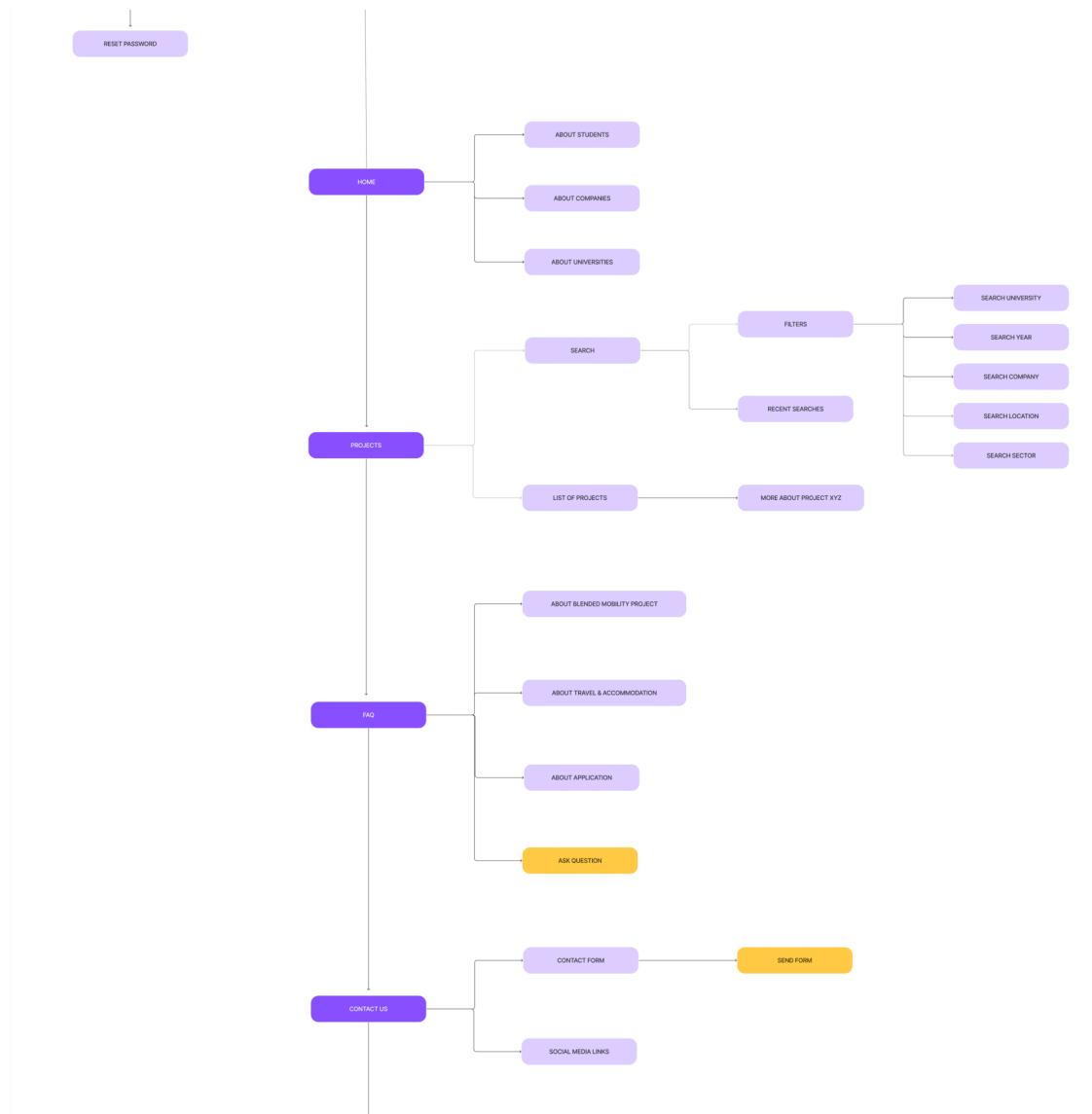


Figura A.2: Diagrama de User Flow da Aplicação 2

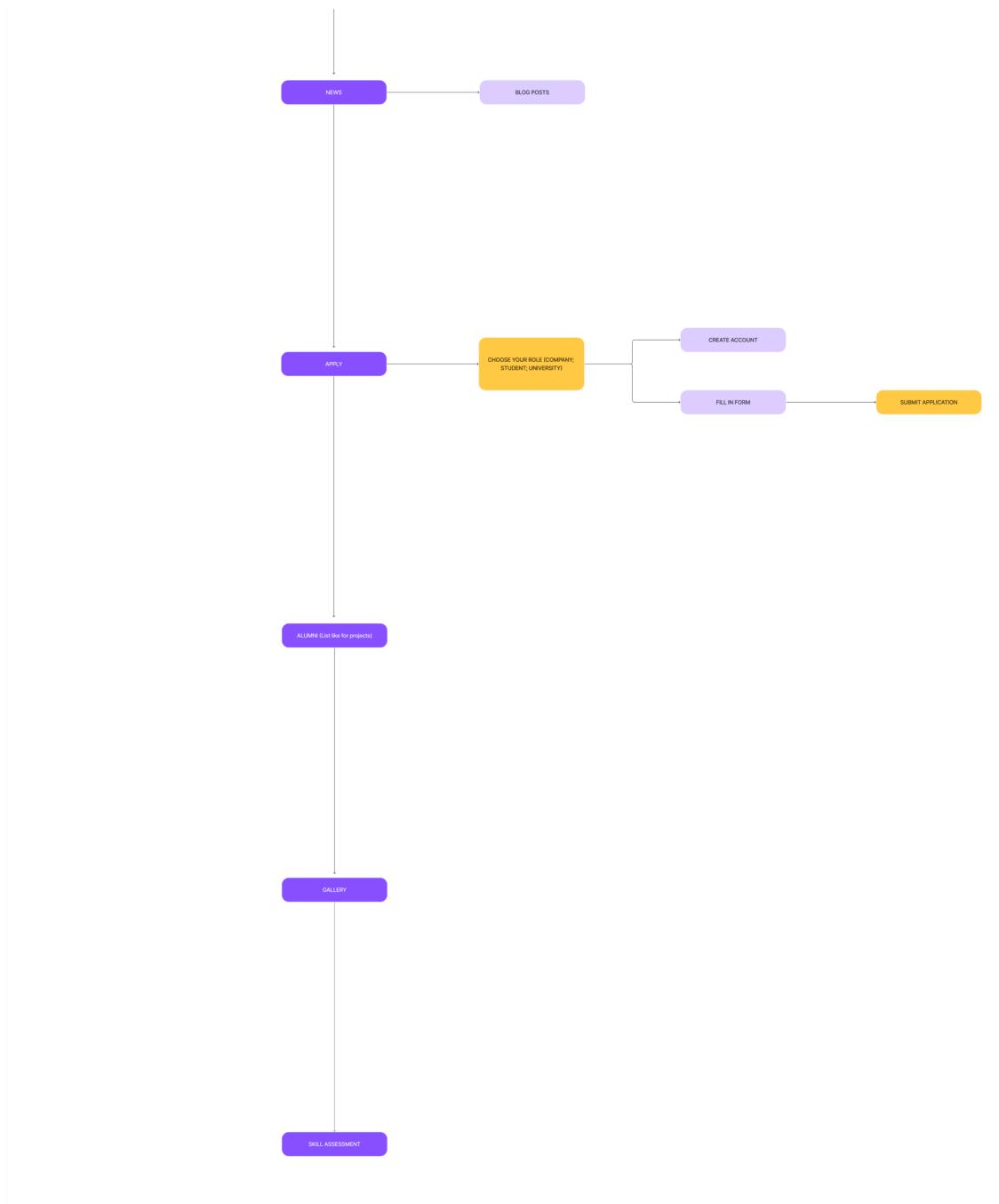


Figura A.3: Diagrama de User Flow da Aplicação 3

Apêndice B

Script para conexão de uma máquina à agent pool do Azure Pipelines

```

set -e

if [ -z "${AZP_URL}" ]; then
    echo 1>&2 "error: missing AZP_URL environment variable"
    exit 1
fi

if [ -n "$AZP_CLIENTID" ]; then
    echo "Using service principal credentials to get token"
    az login --allow-no-subscriptions --service-principal --username
    ↪ "$AZP_CLIENTID" --password "$AZP_CLIENTSECRET" --tenant "$AZP_TENANTID"
    # adapted from
    ↪ https://learn.microsoft.com/en-us/azure/databricks/dev-tools/user-aad-token
    AZP_TOKEN=$(az account get-access-token --query accessToken --output tsv)
    echo "Token retrieved"
fi

if [ -z "${AZP_TOKEN_FILE}" ]; then
    if [ -z "${AZP_TOKEN}" ]; then
        echo 1>&2 "error: missing AZP_TOKEN environment variable"
        exit 1
    fi

    AZP_TOKEN_FILE="/azp/.token"
    echo -n "${AZP_TOKEN}" > "${AZP_TOKEN_FILE}"
fi

unset AZP_CLIENTSECRET
unset AZP_TOKEN

if [ -n "${AZP_WORK}" ]; then
    mkdir -p "${AZP_WORK}"
fi

cleanup() {

```

Apêndice B. Script para conexão de uma máquina à agent pool do Azure Pipelines

```
trap "" EXIT

if [ -e ./config.sh ]; then
    print_header "Cleanup. Removing Azure Pipelines agent..."

    while true; do
        ./config.sh remove --unattended --auth "PAT" --token $(cat
        ↵ "${AZP_TOKEN_FILE}") && break
        echo "Retrying in 30 seconds..."
        sleep 30
    done
fi
}

print_header() {
    lightcyan="\033[1;36m"
    nocolor="\033[0m"
    echo -e "\n${lightcyan}$1${nocolor}\n"
}

export VSO_AGENT_IGNORE="AZP_TOKEN,AZP_TOKEN_FILE"

print_header "1. Determining matching Azure Pipelines agent..."

AZP_AGENT_PACKAGES=$(curl -LsS \
    -u user:$(cat "${AZP_TOKEN_FILE}") \
    -H "Accept:application/json" \
    ↵ "${AZP_URL}/_apis/distributedtask/packages/agent?platform=${TARGETARCH}&top=1")

AZP_AGENT_PACKAGE_LATEST_URL=$(echo "${AZP_AGENT_PACKAGES}" | jq -r
    ↵ ".value[0].downloadUrl")

if [ -z "${AZP_AGENT_PACKAGE_LATEST_URL}" -o
    ↵ "${AZP_AGENT_PACKAGE_LATEST_URL}" == "null" ]; then
    echo 1>&2 "error: could not determine a matching Azure Pipelines agent"
    echo 1>&2 "check that account ${AZP_URL} is correct and the token is valid
    ↵ for that account"
    exit 1
fi

print_header "2. Downloading and extracting Azure Pipelines agent..."

curl -LsS "${AZP_AGENT_PACKAGE_LATEST_URL}" | tar -xz & wait $!

source ./env.sh

trap "cleanup; exit 0" EXIT
trap "cleanup; exit 130" INT
trap "cleanup; exit 143" TERM
```

```
print_header "3. Configuring Azure Pipelines agent..."  
  
./config.sh --unattended \  
--agent "${AZP_AGENT_NAME:-$(hostname)}" \  
--url "${AZP_URL}" \  
--auth "PAT" \  
--token $(cat "${AZP_TOKEN_FILE}") \  
--pool "${AZP_POOL:-Default}" \  
--work "${AZP_WORK:_work}" \  
--replace \  
--acceptTeeEula & wait $!  
  
print_header "4. Running Azure Pipelines agent..."  
  
chmod +x ./run.sh  
  
./run.sh "$@" & wait $!
```

Apêndice C

Dockerfile do Agente Azure Pipelines

```
1 FROM python:3-alpine
2 ENV TARGETARCH="linux-musl-x64"
3
4 RUN apk update && \
5     apk upgrade && \
6     apk add bash curl gcc icu-libs jq musl-dev python3-dev
7     libffi-dev openssl-dev cargo make
8
9 RUN pip install --upgrade pip
10 RUN pip install azure-cli
11 WORKDIR /azp/
12
13 COPY ./start.sh .
14 RUN chmod +x ./start.sh
15
16 RUN adduser -D agent
17 RUN chown agent ./
18 USER agent
19
20 ENTRYPOINT [ "./start.sh" ]
```

appendices/c-pipeline-agent-dockerfile/dockerfile

Apêndice D

Pipeline criada em formato YAML

```

trigger:
  branches:
    include:
      - main

pool: 'blended4future-vms'

variables:
  - group: ssh
  - name: BUILDTAG
    value: blended4future-frontend:latest
  - name: CONTAINERNAME
    value: blended4future-frontend

steps:
  - task: SSH@0
    displayName: "Clean build directory on VM"
    inputs:
      sshEndpoint: 'blended4future-vm'
      runOptions: 'commands'
      commands: |
        sudo rm -rf $(FRONTENDBUILDPATH)
        mkdir -p $(FRONTENDBUILDPATH)
    readyTimeout: '20000'

  - task: CopyFilesOverSSH@0
    displayName: "Copy all necessary files to VM"
    inputs:
      sshEndpoint: 'blended4future-vm'
      sourceFolder: '$(Build.SourcesDirectory)'
      contents: |
        **/*
        !node_modules/**
        ! .git/**

```

```
!cypress/**  
targetFolder: $(FRONTENDBUILDPATH)  
readyTimeout: '20000'  
  
- task: SSH@0  
  displayName: "Move nginx.conf & Restart Nginx"  
  inputs:  
    sshEndpoint: 'blended4future-vm'  
    runOptions: 'commands'  
    failOnStdErr: false  
  commands: |  
    echo "Deploying nginx.conf and restarting Nginx..."  
    sudo mv $(FRONTENDBUILDPATH)/nginx.conf /etc/nginx/nginx.conf  
    sudo nginx -t && sudo systemctl restart nginx  
  readyTimeout: '20000'  
  
- task: SSH@0  
  displayName: "Remove Old Docker Image (optional clean)"  
  continueOnError: true  
  inputs:  
    sshEndpoint: 'blended4future-vm'  
    runOptions: 'commands'  
  commands: |  
    echo "Cleaning up old Docker image..."  
    sudo docker rmi -f $(BUILDTAG) || true  
  readyTimeout: '20000'  
  
- task: SSH@0  
  displayName: "Build Frontend Docker image on VM"  
  continueOnError: true  
  inputs:  
    sshEndpoint: 'blended4future-vm'  
    runOptions: 'commands'  
    failOnStdErr: false  
  commands: |  
    echo "Building Docker image..."  
    sudo docker build -t $(BUILDTAG) $(FRONTENDBUILDPATH)  
  readyTimeout: '20000'  
  
- task: SSH@0  
  displayName: "Remove Previous Frontend Container on VM"  
  continueOnError: true  
  inputs:  
    sshEndpoint: 'blended4future-vm'  
    runOptions: 'commands'
```

```
commands: |
  echo "Removing old Docker container..."
  sudo docker rm -f $(CONTAINERNAME) || true
readyTimeout: '20000'

- task: SSH@0
  displayName: "Run Docker Container"
  inputs:
    sshEndpoint: 'blended4future-vm'
    runOptions: 'commands'
  commands: |
    echo "Starting new Docker container..."
    sudo docker run -d --name $(CONTAINERNAME) -p 3000:3000 $(BUILDTAG)
  readyTimeout: '20000'
```