ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Desenvolvimento de uma plataforma de gestão de projetos

Blended4Future

1201367 Carlos Rodrigo Marques dos Santos

Orientador: Professor Nuno Escudeiro

Porto, Setembro, 2025

Resumo

Trabalhos escritos em língua Inglesa devem incluir um resumo alargado com cerca de 1000 palavras, ou duas páginas.

Se o trabalho estiver escrito em Português, este resumo deveria ser em língua Inglesa, com cerca de 200 palavras, ou uma página.

Para alterar a língua basta ir às configurações do documento no ficheiro **`main.tex`** e alterar para a língua desejada ('english' ou 'portuguese')[1]. Isto fará com que os cabeçalhos incluídos no template sejam traduzidos para a respetiva língua.

Palavras-chave: Frontend, Backend, Software Architecture,

[1] Alterar a língua requer apagar alguns ficheiros temporários; O target **clean** do **Makefile** incluído pode ser utilizado para este propósito.

Agradecimentos

The optional Acknowledgment goes here... Below is an example of a humorous acknowledgment.

"I'd also like to thank the Van Allen belts for protecting us from the harmful solar wind, and the earth for being just the right distance from the sun for being conducive to life, and for the ability for water atoms to clump so efficiently, for pretty much the same reason. Finally, I'd like to thank every single one of my forebears for surviving long enough in this hostile world to procreate. Without any one of you, this book would not have been possible."in "The Woman Who Died a Lot"by Jasper Fforde.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Código

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 BlendedMobility[1]). 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 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

[1] https://blendedmobility.com

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 espectado, 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 em uma só plataforma. Por tal foi colocada um 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, fazer posts sobre os seus respetivos projetos.

Foi ainda requisitado uma maneira de ver todas a edições do Blended4Future e todos o 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. Para 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 subequipas:

- 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 com o objetivo de atualizar o progresso das tarefas e promover um ambiente de trabalho mais colaborativo e comunicativo. Estas reuniões não possuiam uma data definida tendo em conta os fusos horarios de cada um dos membros

1.4.2.1 Divisão de sprints

Sprint	Semanas	Data de início	Data de fim	Objetivos definidos
1	1-2	24/02	02/03	1. Documentação geral do projeto 2. Configuração dos repositórios de Backend/Frontend com pipelines de CI/CD 3. Configuração de DevOps/Azure 4. Análise SWOT do negócio
2	3-4	10/03	23/03	1. Finalização da documentação geral 2. Início do desenvolvimento
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.

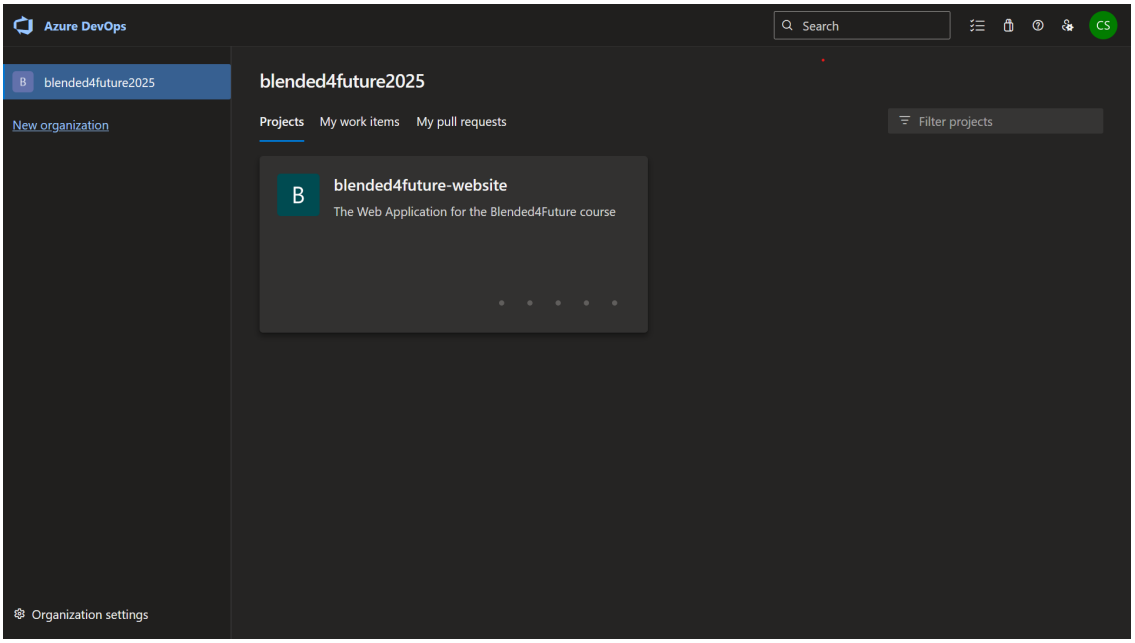

Figura 1.1: Homepage do projeto

1.4.3 Tecnologias Definidas

Na referida semana de kickoff foi necessária uma deliberação sobre as tecnologias e ferramentas a utilizar. Durante o desenvolvimento foram adicionadas ainda outras com o intuito de agilizar ainda mais este.

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 aconcelhamento 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.

Outras possiveis 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

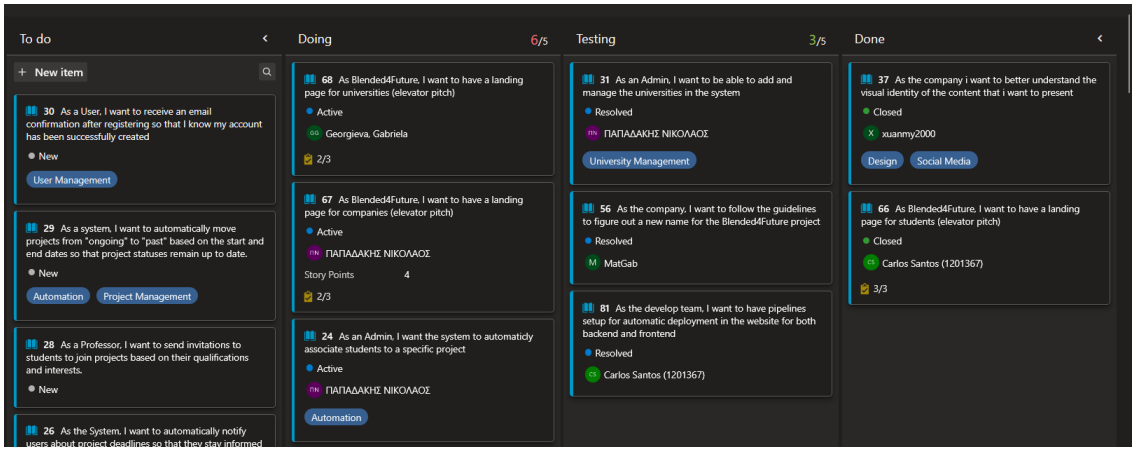

Figura 1.2: Exemplificação do uso das *boards* no DevOps

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

1.4.3.2 Tecnologias para desenvolvimento

Atendendo à heterogeneidade da equipa, composta por indivíduos com diferentes experiências e origens académicas, 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.

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

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 destaca-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 Aspectos, permitindo isolar funcionalidades transversais como segurança e logging. Entras a suas varias funcionalidades denota-se o Spring Data, que abstrai o acesso a repositórios e bases de dados, facilitando a interação com diversos sistemas de armazenamento através 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 repetível do comportamento de unidades individuais dentro de uma 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 destacam-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 retrospetiva, 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

2.1 ESN - European Student Network

2.2 Plataforma de estágios PRAXIS

2.3 Spain Internship

Capítulo 3

Analise do problema e desenho de uma solução

Adicionar texto introdutorio

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

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

A tabela 3.1 reflete esta decisão.

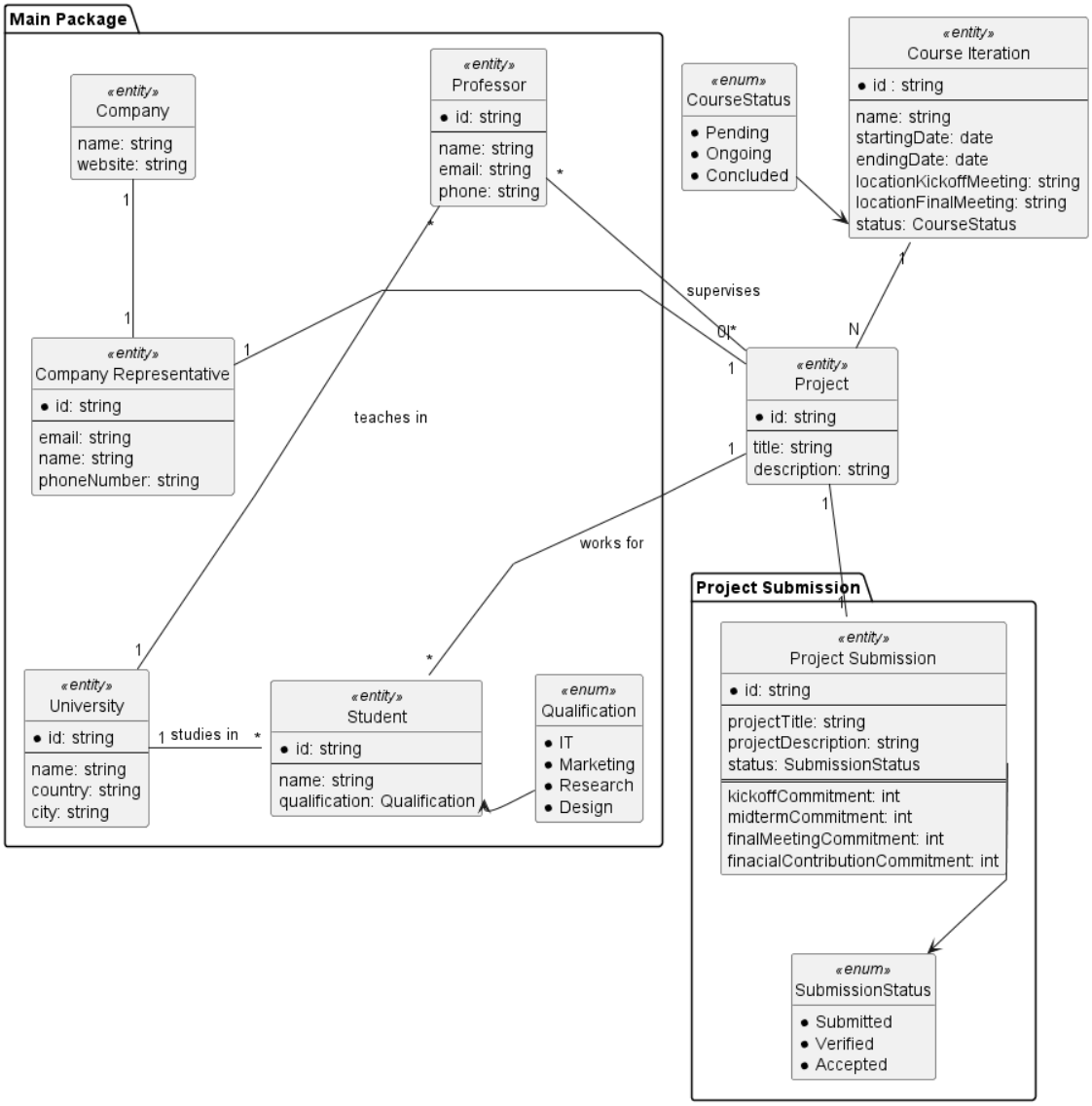

Figura 3.1: Diagrama de classes da lógica de negócio do Blended4Future 1

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 disponivel, 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
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

User Story	Título
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 Desenho do sistema

3.3.2 Nivel 1

3.3.2.1 Diagrama Logico

3.3.2.2 Diagrama de Implementação

3.3.2.3 Diagrama Físico

A figura 3.2 demonstra o diagrama de físico de nível 1.

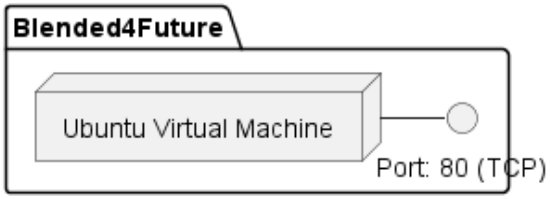

Figura 3.2: Diagrama Físico de Nível 1

3.3.3 Nivel 2

3.3.3.1 Diagrama Logico

3.3.3.2 Diagrama de Implementação

A figura 3.3 demonstra o diagrama de físico de nível 2.

3.3.3.3 Diagrama Físico

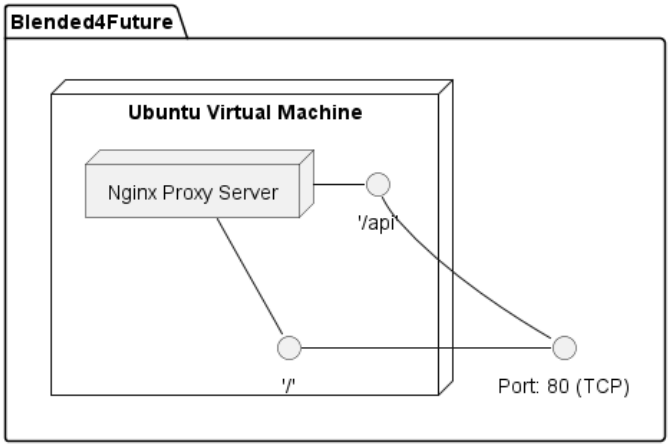

Figura 3.3: Diagrama Físico de Nível 2

3.3.4 Nivel 3

3.3.4.1 Diagrama Logico

3.3.4.2 Diagrama de Implementação

3.3.4.3 Diagrama Físico

A figura 3.4 demonstra o diagrama de físico de nível 3

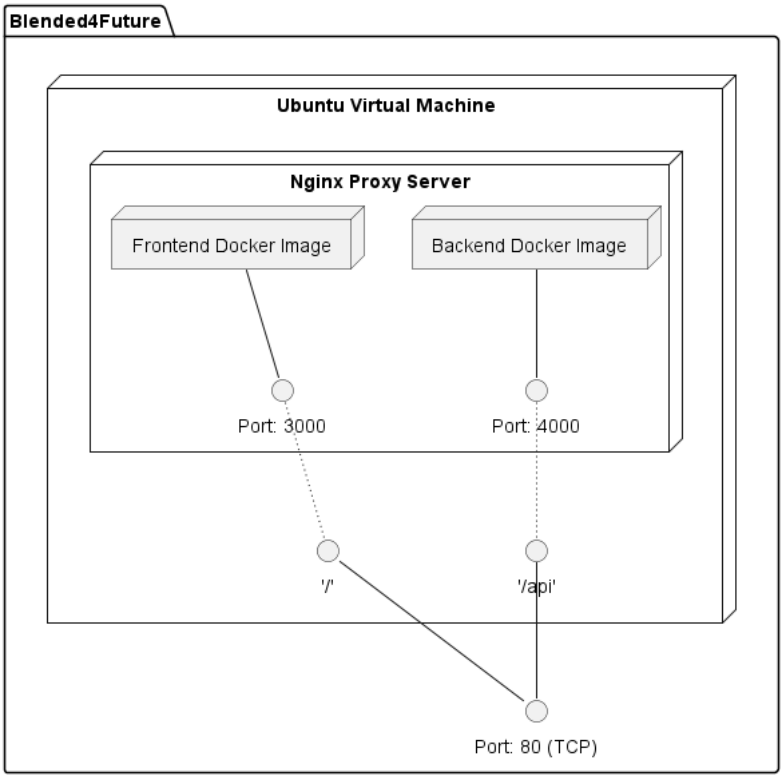

Figura 3.4: Diagrama Físico de Nível 3

3.3.5 Nivel 4 - Código

3.3.5.1 Diagrama Logico

3.3.5.2 Diagrama de Implementação

3.3.5.3 Diagrama Físico

A figura 3.5 demonstra o diagrama de físico de nível 4.

3.3.6 Padrões utilizados

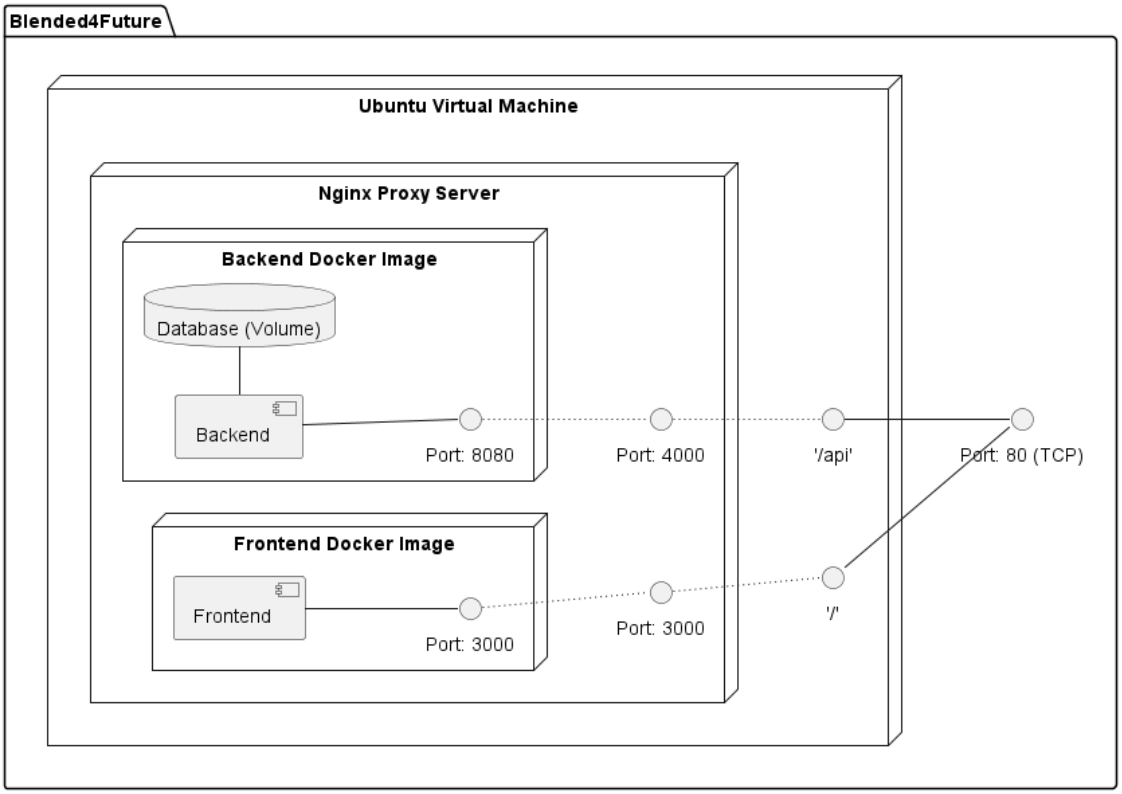

Figura 3.5: Diagrama Físico de Nível 4

Capítulo 4

Implementação de uma solução

4.1 A Implementação

Este capitulo demonstrará a implementação de certas funcionalidades baseada na arquitetura apresentada anteriormente. Estas irão modulos de frontend, backend e as pipelines de deployment necessárias.

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 programção 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*.

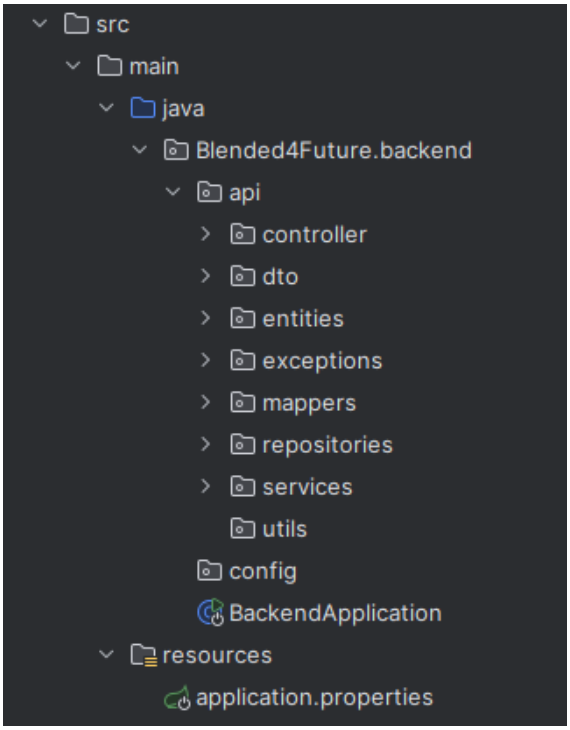

Figura 4.1: Estrutura de pastas do backend

- **Bean** corresponde a uma instância de objeto gerida pelo *IoC Container*. Cada *Bean* é criado, configurado e mantido pelo próprio contentor, 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.

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 4.2.

Em nota, interessante enaltecer o uso das respetivas 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 **??**);
- implementa métodos *getter* para todos os atributos privados
- implementa metodos *setter* para todos os atributos privados
- implementa um construtor com um parâmetro por atributo não final da classe.

4.1.1.4 BaseEntity

Como medida de segurança e de padronização, foi desenvolvida a classe *BaseEntity* (ver listagem **??**). 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();
```

```
1
2  @Entity
3  @Data
4  @EqualsAndHashCode(callSuper = true)
5  public class Project extends BaseEntity {
6
7      @Column(nullable = false)
8      @Convert(converter = ProjectName.NameConverter.class)
9      private ProjectName name = new ProjectName();
10
11     @Column(nullable = false)
12     @Convert(converter =
      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)
28     private Set<Report> reports = Set.of();
29
30     @ManyToOne
31     private CourseEdition courseEdition;
32 }
33
34
```

Figura 4.2: Classe *Project*

```
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>
        isBusinessDataValid(List<IValueObject<?>> fields) {
45          HashMap<String, Object> errors = new HashMap<>();
46          for (IValueObject<?> field : fields) {
47              if (field == null) {
48                  continue;
49              }
50              errors.putAll(field.isValid());
51          }
52          return errors;
53      }
54  }
```

Listing 4.1: Classe *BaseEntity*

4.1.1.5 Value Objects

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

A listagem 4.2 mostra a interface `ValueObject`.

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

Listing 4.2: 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.

```
@Value
public class ProjectDescription implements IValueObject<String> {

    public static final int MIN_LENGTH = 10;
    public static final int MAX_LENGTH = 500;
    public static final String DEFAULT_DESCRIPTION = "Default Project
    Description";

    private String description;

    @Override
    public String getValue() {
        return this.description;
    }

    public ProjectDescription(String value) {
        this.description = value;
    }

    public ProjectDescription() {
        this(DEFAULT_DESCRIPTION);
    }

    @Override
    public HashMap<String, Object> isValid() {
        HashMap<String, Object> errors = new HashMap<>();
        if (this.getValue() == null || this.getValue().isEmpty()) {
            errors.put("description", "Description cannot be null or
    empty");
        } else if (this.getValue().length() < MIN_LENGTH ||
    this.getValue().length() > MAX_LENGTH) {
            errors.put("description", "Description must be between 10
    and 500 characters long");
        }
        return errors;
    }

    @Converter
    public static class DescriptionConverter implements
    jakarta.persistence.AttributeConverter<ProjectDescription, String>
    {
        @Override
        public String convertToDatabaseColumn(ProjectDescription
    description) {
            return description.getValue();
        }
```

```
40
41          @Override
42          public ProjectDescription convertToEntityAttribute(String
        dbData) {
43              return new ProjectDescription(dbData);
44          }
45      }
46  }
```

Listing 4.3: Classe *ProjectDescription*

A listagem 4.3 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 4.4).

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 especifica do Spring [5].

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

Listing 4.4: Inteface BaseRepository

Graças ao *Spring* é possível então fazer implementações de repositórios muito facilmente, como exemplificado na listagem 4.5

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

Listing 4.5: 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 formata 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 4.6 entende-se o elemento base que permite capturar erros internos ou de negócio e retorná-los num formato predefinido (ver listagem 4.7). 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 4.8 e **??** mostra-se, respetivamente, exemplo do uso desta exceções e propria exceção `AppException`.

```
@Slf4j
@RestControllerAdvice
@RequiredArgsConstructor
public class AppExceptionController {

    // Everytime an app exception is thrown, this method will be called
    // It will log the error and return a ResponseEntity with the error details
    @ResponseBody
    @ExceptionHandler(AppException.class)
    public ResponseEntity<?> handleAppException(AppException e, HttpServletRequest request) {
        log.error("App Exception occurred: {}", e.getMessage(), e);
        return createResponseError(e, request.getRequestURI());
    }

    private ResponseEntity<ErrorResponse> createResponseError(AppException e, String path) {
        Error err = new Error(
                e.getMessage(),
                e.getStatus().value(),
                path,
                Instant.now(),
                e.getData()
        );

        ErrorResponse errorResponse = ErrorResponse.fromError(err);

        return new ResponseEntity<>(errorResponse,new HttpHeaders(), e.getStatus());
    }

}
```

Listing 4.6: Class *AppExceptionController*

```
@Value
@AllArgsConstructor(access = PUBLIC)
public class Error {
    String message;
    int status;
    String path;
```

```
7       Instant timestamp;
8       Map<String, Object> data;
9   }
```

Listing 4.7: 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,
    e.getErrors());
34          }
35      }
36
37      @PutMapping("/{id}")
38      public ProjectDTO updateProject(@PathVariable UUID id,
    @RequestBody @Valid AddProjectDTO project) {
39          try {
40              return service.updateProject(id, project);
41          } catch (NoSuchElementException e) {
42              throw new AppException(e, HttpStatus.NOT_FOUND);
43          } catch (FormDataException e) {
44              throw new AppException(e, HttpStatus.BAD_REQUEST,
    e.getErrors());
```

```
45          }
46      }
47
48      @DeleteMapping("/{id}")
49      public ProjectDTO deleteProject(@PathVariable UUID id) {
50          try {
51              return service.deleteProject(id);
52          } catch (NoSuchElementException e) {
53              throw new AppException(e, HttpStatus.NOT_FOUND);
54          }
55      }
56  }
```

Listing 4.8: 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,
        Map<String, Object> data) {
8           super(message);
9           this.status = status;
10          this.data = data;
11      }
12
13      public AppException(Exception e, HttpStatus status, Map<String,
        Object> data) {
14          super(e.getMessage());
15          this.status = status;
16          this.data = data;
17      }
18
19      public AppException(Exception e, HttpStatus status) {
20          super(e.getMessage());
21          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 4.9: Exceção *AppException*

```
sudo docker run -d
    -e AZP_URL="https://dev.azure.com/blended4future2025"
    -e AZP_TOKEN="******"
    -e AZP_POOL="*******"
    -e AZP_AGENT_NAME="*****"
    --name "azp-agent-linux"
    devops-agent:linux
```

Figura 4.3: Script de inicialização do Container Docker para o agente do Azure Pipelines

4.1.2 Frontend

4.1.3 Controlo de Versões

Em todos os repositórios em uso foi definido o uso de multiplos *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 simplicar o processo de desenvolvimento, facilitando a disponibilização de novas versões.

A plataforma Azure disponbiliza 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 màquina Linux Ubuntu 24.04.2.

4.1.4.1 Definição dos agentes

No contexto do Azure Pipelines um agente (em inglês, *Agent*) é a maquiná indica 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 compilção 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 variaveis de ambiente necessárias. A figura 4.3 demonstra isso mesmo, com censura a qualquer informação sensivel à organização.

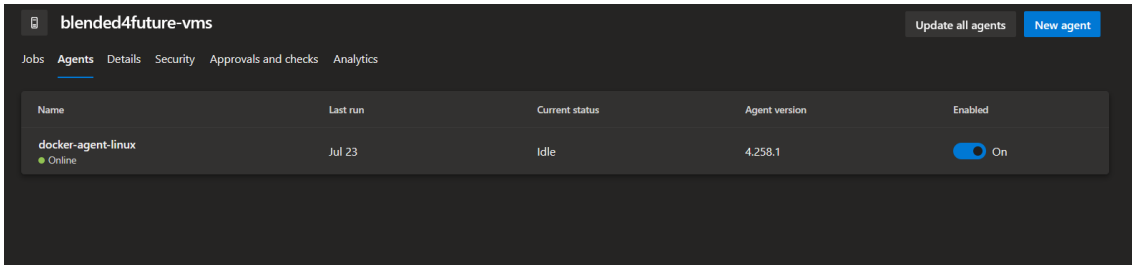

Figura 4.4: Agente disponbilizado no Azure DevOps

Figura 4.5: Diagrama de atividade representante da Pipeline de construção de uma nova versão

Logo após, é possivél ver o agente disponível no Azure DevOps (Figura 4.4).

4.1.4.2 Trabalhos da pipeline

O conceito de trabalho (em inglês, *Job*) é de extrema importância no Azure Pipelines. Este premite definir ações e os seus resptivos 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 appêndice D.

Na figura 4.5, 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 4.10: Trigger - Pipeline

A listagem 4.10, 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 4.5. Especial atenção para o uso de `SSH@0`[4], esta é uma task pre disponbilizada para estabelecer uma conexção SSH a uma màquina alvo. A chave está definida como segredo e é importado no início do ficheiro quando ocorre uma referência ao grupo de variaveis "ssh".

4.2 Testes

4.2.1 Testes Unitários

4.2.2 Testes de Implementação

4.3 Avaliação da solução

Capítulo 5

Conclusão

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/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html.

Apêndice A

Diagrama de User Flow

Figura A.1: Diagram de User Flow da Aplicação

Apêndice B

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

Apêndice C

Dockerfile do Agente Azure Pipelines

```
FROM python:3-alpine
ENV TARGETARCH="linux-musl-x64"

RUN apk update && \
  apk upgrade && \
  apk add bash curl gcc git icu-libs jq musl-dev python3-dev
   libffi-dev openssl-dev cargo make

RUN pip install --upgrade pip
RUN pip install azure-cli

WORKDIR /azp/

COPY ./start.sh ./
RUN chmod +x ./start.sh

RUN adduser -D agent
RUN chown agent ./
USER agent

ENTRYPOINT [ "./start.sh" ]
```

appendices/c-pipeline-agent-dockerfile/dockerfile

Apêndice D

Pipeline criada em formato YAML

```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:
```

```
42        sshEndpoint: 'blended4future-vm'
43        runOptions: 'commands'
44        failOnStdErr: false
45        commands: |
46          echo "Deploying nginx.conf and restarting Nginx..."
47          sudo mv $(FRONTENDBUILDPATH)/nginx.conf /etc/nginx/nginx.conf
48          sudo nginx -t && sudo systemctl restart nginx
49        readyTimeout: '20000'
50
51    - task: SSH@0
52      displayName: "Remove Old Docker Image (optional clean)"
53      continueOnError: true
54      inputs:
55        sshEndpoint: 'blended4future-vm'
56        runOptions: 'commands'
57        commands: |
58          echo "Cleaning up old Docker image..."
59          sudo docker rmi -f $(BUILDTAG) || true
60        readyTimeout: '20000'
61
62    - task: SSH@0
63      displayName: "Build Frontend Docker image on VM"
64      continueOnError: true
65      inputs:
66        sshEndpoint: 'blended4future-vm'
67        runOptions: 'commands'
68        failOnStdErr: false
69        commands: |
70          echo "Building Docker image..."
71          sudo docker build -t $(BUILDTAG) $(FRONTENDBUILDPATH)
72        readyTimeout: '20000'
73
74    - task: SSH@0
75      displayName: "Remove Previous Frontend Container on VM"
76      continueOnError: true
77      inputs:
78        sshEndpoint: 'blended4future-vm'
79        runOptions: 'commands'
80        commands: |
81          echo "Removing old Docker container..."
82          sudo docker rm -f $(CONTAINERNAME) || true
83        readyTimeout: '20000'
84
85    - task: SSH@0
86      displayName: "Run Docker Container"
87      inputs:
88        sshEndpoint: 'blended4future-vm'
89        runOptions: 'commands'
90        commands: |
91          echo "Starting new Docker container..."
92          sudo docker run -d --name $(CONTAINERNAME) -p 3000:3000
      $(BUILDTAG)
93        readyTimeout: '20000'
```

appendices/d–pipeline–yaml/pipline.yaml