UNIVERSIDADE
BEIRA INTERIOR

Estágio na Empresa Latittude - Digital Enablers, LDA: Desenvolvimento de uma Web App e de uma Server App em Contentor Docker

Versão Definitiva Após Defesa Pública

Inês Batista dos Santos

Relatório de Estágio para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Mário Marques Freire
Co-orientador: João Gouveia

Setembro de 2023

Declaração de Integridade

Eu, Inês Batista dos Santos, que abaixo assino, estudante com número de inscrição M11660 do curso de 2ºciclo de Engenharia Informática da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o Código de Integridade da Universidade da Beira Interior.

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

Universidade da Beira Interior, Covilhã 26/09/2023.

Assinado por: **Inês Batista dos Santos**
Num. de Identificação: 15367954

Inês Batista dos Santos

Agradecimentos

Gostaria de expressar a minha profunda gratidão a todos aqueles que tornaram possível este estágio e me ajudaram a alcançar os meus objetivos. Antes de mais, gostaria de agradecer aos meus **orientadores** Mário Marques Freire e João Gouveia, que me deram um apoio inabalável e me ensinaram tanto sobre o trabalho e a indústria.

Gostaria de agradecer aos meus **colegas**, que me acolheram de braços abertos e me ajudaram a sentir-me parte da equipa. Aprendi muito com cada um de vós e espero colaborar em projetos futuros.

À minha **família**, gostaria de agradecer por terem aceite as minhas decisões ao longo da minha jornada académica e pelo apoio constante durante estes anos.

Finalmente, também quero agradecer aos meus **amigos** pelo apoio constante quer nos bons, quer nos maus momentos, por me terem ajudado tanto ao longo dos anos e espero que a nossa amizade se mantenha nos anos futuros.

Agradeço a todos do fundo do meu coração e espero retribuir a todos os que me ajudaram nesta viagem de alguma forma.

Resumo

O projeto de estágio foi realizado na empresa Latitudde como parte da conclusão do mestrado em Engenharia Informática na Universidade da Beira Interior (UBI) . A empresa Latitude é reconhecida pelo seu foco em soluções tecnológicas inovadoras e especialização em arquiteturas de microserviços. Durante o estágio, o objetivo foi construir uma aplicação web utilizando *Angular* para o *frontend* e *.NET* para o *backend*.

O foco principal foi a implementação de uma arquitetura de *REST API*, onde o *Ocelot* foi utilizado como *API Gateway* para simplificar a comunicação entre os microserviços. Além disso, o *Keycloak* foi incorporado para fornecer autenticação e autorização seguras. A utilização do *Docker* permitiu a contentorização da aplicação, facilitando a implantação e a gestão dos serviços.

Palavras-chave

Angular; .NET; Frontend; Backend; REST; Ocelot; Keycloak; Docker

Abstract

The internship project was carried out at the company Latitudde as part of the conclusion of the master's degree in Computer Engineering at UBI . The company Latitude is recognized for its focus on innovative technological solutions and specialization in microservices architectures. During the internship, the goal was to build a web application using Angular for the *frontend* and .NET for the backend.

The main focus was on implementing a REST API architecture, where Ocelot was used as API Gateway to simplify communication between microservices. In addition, Keycloak was incorporated to provide secure authentication and authorization. The use of Docker enabled containerization of the application, making it easier to deploy and manage the services.

Keywords

Angular; .NET; Frontend; Backend; REST; Ocelot; Keycloak; Docker

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

API *Application Programing Interface*

CRUD *Create, Read, Update, Delete*

CSRF *Cross-Site Request Forgery*

CSS *Cascading Style Sheets*

HTML *Hypertext Markup Language*

IAM *Identity and Access Management*

IDE *Integrated Development Environment*

REST API *Representational State Transfer Application Programming Interface*

SPA *Single-Page Application*

SSO *Single Sign-on*

VSCode *Visual Studio Code*

UBI Universidade da Beira Interior

UI *User Interface*

WWW *World Wide Web*

XSS *Cross-Site Scripting*

Capítulo 1

Introdução

1.1 Enquadramento

Este documento pretende descrever as atividades mais relevantes do estágio realizado na empresa Latitudde, no âmbito da Unidade Curricular de Dissertação ou de Estágio em Engenharia Informática na UBI.

Durante o estágio na empresa Latitude, foi desenvolvida uma aplicação *web* com *Angular* no *frontend* e *.NET* no *backend*. O projeto focou na criação de uma arquitetura de *REST API* com o *Ocelot* como *API Gateway* e a integração do *Keycloak* para autenticação e autorização, tendo a aplicação sido contentorizada com o *Docker*.

Ao longo do documento irão ser descritas as atividades que irão ser realizadas durante o estágio.

1.2 Objetivos

Este relatório descreve todas as atividades realizadas durante o estágio, incluindo a noção de ética de trabalho, a evolução do aluno ao longo do período e a aquisição de habilidades fundamentais para o mundo profissional.

Os principais objetivos do estágio consistem em:

1. Ser assíduo e pontual;
2. Cumprir os prazos de trabalho;
3. Conseguir aplicar os conceitos aprendidos durante a formação;
4. Ter uma noção das éticas de trabalho;
5. Adquirir experiência na colaboração com uma equipa de desenvolvimento e seguir os processos de desenvolvimento de software da empresa;
6. Adquirir conhecimento nas ferramentas, linguagens e *frameworks* durante o tempo de estágio;
7. Construir uma aplicação com base nos conceitos aprendidos durante a formação.

1.3 Organização do Documento

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

1. O primeiro capítulo – **Introdução** – apresenta o enquadramento do estágio, os objetivos e a organização do documento.
2. O segundo capítulo – **Descrição da Empresa e do Plano de Estágio** – apresenta uma breve descrição da Empresa referida anteriormente, descrevendo as suas atividades, história, metas e explica o objetivo do estágio, as tecnologias a ser utilizadas e o cronograma do estágio.
3. O terceiro capítulo – **Tecnologias e Ferramentas Utilizadas** – descreve as tecnologias utilizadas na realização do mesmo;
4. O quarto capítulo – **Trabalho Desenvolvido** – descreve qual foi o projeto proposto, como foi estruturado e implementado e visa demonstrar todo o processo conceitual para a elaboração da aplicação, como os requisitos, os diagramas de sequência e de atividade;
5. O quinto capítulo – **Conclusão** – apresenta as conclusões sobre a realização deste projeto e como foi a experiência na realização do estágio.

Capítulo 2

Descrição da Empresa e do Plano de Estágio

2.1 Introdução

A Latitude[1] é uma empresa multinacional especializada em Projetos de transformação digital do Grupo ReadinessIT, com mais de 400 colaboradores, com presença em quatro países (Portugal, Chile, Peru e Nova Zelândia) e experiência em projetos internacionais. Os seus especialistas seguem rigorosos padrões de qualidade e são capacitados em Centros de Excelência de Tecnologia da Informação do grupo RIT.

2.2 História da Empresa

O grupo RIT foi criado para complementar a oferta da ReadinessIT, uma empresa fundada em 2015 por especialistas com mais de 25 anos de experiência no setor de telecomunicações. A ReadinessIT oferece serviços tecnológicos de alto valor a nível global, e o grupo RIT surgiu com o objetivo de criar empresas especializadas em soluções tecnológicas para problemas específicos. Uma dessas empresas é a Latitudde, que se especializa em áreas como *Software Engineering* e *Low-Code*.

A Latitudde é uma empresa fundada no início de 2021 por 10 pessoas, sendo a maioria do grupo proveniente da empresa mãe. Está agora consolidada e a funcionar com mais de 80 colaboradores, levando a cabo vários projetos para múltiplos clientes. A Latitudde visa diversificar a oferta do grupo e fornecer soluções para uma série de projetos que utilizam diferentes tecnologias em vários setores de negócios, não se limitando às Telecomunicações.

Atualmente, a empresa está envolvida em projetos nos setores da banca, saúde, retalho, governo, e transportes. Outro objetivo é não depender apenas de uma tecnologia específica, e por isso a empresa trabalha com as principais tecnologias de mercado de forma diversificada, tais como *PHP, Python, Java, .NET, Angular, React* e entre outras.

2.3 Horário de Trabalho/Funcionamento

A Latitudde está inserida no Centro da Moagem, localizada no Fundão, e o horário de funcionamento da Moagem é das 09h30m até às 17h30m. Por norma o horário de trabalho de cada colaborador da Latitudde é das 9h até 18h, podendo este depender dos clientes e dos projetos atribuídos a cada colaborador.

2.4 Formação

A Latitudde oferece um programa de formação para novos *developers* chamado *Seven Summits Academy*, com o objetivo de preparar pessoas para a área de programação e potencialmente integrá-las no quadro de funcionários da empresa. A formação que a estagiária fez foi nas *frameworks Angular* [2] e *ASP.NET Core* [3].

2.5 Projeto Proposto

Como foi referido no capítulo **Descrição da Empresa e do Plano de Estágio**, na secção **Formação**, a estagiária fez a formação nas *frameworks* Angular e ASP.NET Core. Por norma, para fazer o desenvolvimento de *back-end .NET* é utilizado:

- *Visual Studio CE 2022*;
- *Postman* [4] para testar a *Application Programing Interface* (API) desenvolvida;
- *Docker*;
- *GIT* [5] e um programa de gestão do *GIT*.

E para o desenvolvimento de *frontend* será feito em *Hypertext Markup Language* (HTML) e *Cascading Style Sheets* (CSS) com o apoio da *framework Angular* e é utilizado o *Integrated Development Environment* (IDE) *Visual Studio Code* (VSCode).

No fim da formação, foi proposto um desafio de *frontend* e *backend* para por em prática os conceitos aprendidos, que irá ser explicado nos capítulos seguintes as ferramentas utilizadas e o projeto em si.

2.6 Cronograma

O estágio iniciou-se a 22 de fevereiro de 2023 e terminou a 9 de junho de 2023, com um total de 616 horas, realizadas desde das 9h até às 18h, como demonstra a figura 2.1.
O registo da assiduidade é feito diariamente na plataforma *Bamboo* que é uma ferramenta disponibilizada pela empresa no qual o objetivo é gerir o registo de horas de todos os colaboradores da empresa.

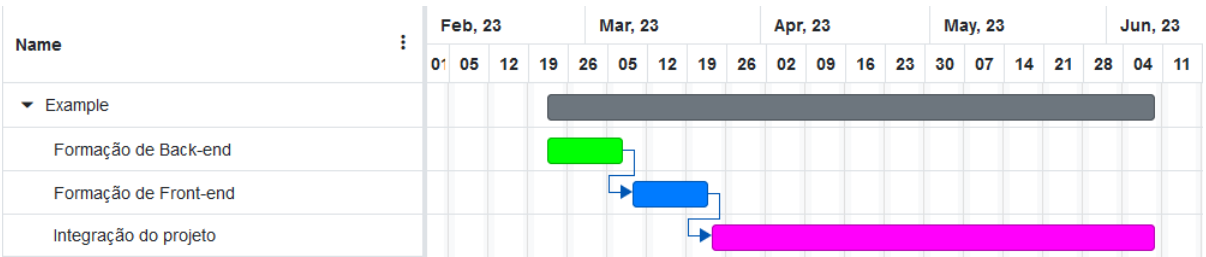

Figura 2.1: Diagrama de Gantt do estágio

2.7 Perfis dos Orientadores

Mário Marques Freire é professor catedrático de Informática na Universidade da Beira Interior e investigador sénior no Instituto de Telecomunicações. Ele completou a sua formação em Engenharia Eletrotécnica e em Informática tem interesses de investigação em redes e sistemas de computadores, incluindo virtualização, computação na nuvem, segurança e privacidade. É autor de 7 patentes internacionais, editor de 8 livros e coautor de mais de 130 artigos científicos. Foi Vice-Reitor da UBI, Presidente da Faculdade de Engenharia da UBI, Diretor do Centro de Informática e Presidente do Departamento de Informática da UBI. É membro do IEEE e da Association for Computing Machinery e foi Coordenador do Colégio de Engenharia Informática da Região Centro e Chair do IEEE Computer Society Chapter Portugal Section.

João Gouveia fez a sua formação em Engenharia Informática na Universidade da Beira Interior e tirou mestrado em Engenharia Informática na mesma instituição. Iniciou a sua experiência profissional na Portugal Telecom em 2014 até 2015, trabalhando na solução Telecom Order Care e desde então continua a trabalhar nesta área. A partir de agosto de 2015 começou a trabalhar como Consultor na ReadinessIT e atualmente é o *Technical Leader* na Latitudde. Trabalhou em muitos projetos na mesma área, utilizando a ferramenta Ericsson denominada Velocity Studio, que permite às empresas construírem a sua própria solução de sistema integrado. Tem trabalhado como consultor de integração de soluções, ajudando as empresas de telecomunicações a atualizar os seus sistemas e a migrar para a era digital. As suas linguagens de domínio são Java, Javascript (com o Angular como estrutura de suporte), e OracleSQL.

Capítulo 3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

Este capítulo encontra-se organizado em quatro secções. Após esta secção introdutória, a secção 3.2 apresenta uma breve descrição das ferramentas utilizadas na construção da aplicação. A secção 3.3 aborda as linguagens de programação utilizadas e a secção 3.4 apresenta uma breve descrição das *Frameworks* usadas no desenvolvimento da aplicação.

3.2 Ferramentas

Para o projeto desenvolvido, foram utilizados as seguintes ferramentas: VSCode, Visual Studio 2022, Docker, PostgreSQL, Keycloak. Estas ferramentas foram utilizadas a pedido do orientador do estágio na empresa.

3.2.1 Visual Studio Code

O VSCode [6] é um editor de código aberto gratuito criado pela Microsoft que foi concebido para uma variedade de fluxos de trabalho de desenvolvimento de software, incluindo o desenvolvimento de web e aplicações. Oferece várias características chave, tais como o completamento inteligente de código, capacidades de depuração, integração com Git, e suporte para múltiplas linguagens de programação e formatos de ficheiro.

O VSCode pode ser utilizado em Windows, macOS e Linux e é versátil o suficiente para ser utilizado como editor de código autónomo ou como parte de uma configuração de desenvolvimento maior.

3.2.2 Visual Studio 2022

O Microsoft Visual Studio [7] é um ambiente de desenvolvimento de *software* que fornece um conjunto abrangente de ferramentas para a construção e depuração das aplicações. Suporta uma vasta gama de linguagens e plataformas de programação, incluindo .NET, C++, e Python, e é utilizado principalmente para o desenvolvimento de Windows.

O Visual Studio fornece um IDE que inclui características como o editor de código, a navegação de código, e ferramentas de depuração. Também inclui uma variedade de ferramentas para construir e implementar aplicações, tais como um sistema de controlo de fontes, ferramentas de automação de construção e implementação, e ferramentas de teste.

Para além das suas características principais, o Visual Studio também suporta inúmeras extensões e *plugins* que podem acrescentar novas funcionalidades ao IDE, tais como suporte para linguagens de programação adicionais, novos editores de código, e outras ferramentas de desenvolvimento.

Esta ferramenta tem várias edições, tais como Visual Studio Community (que é gratuita), Visual Studio Professional, e Visual Studio Enterprise, cada uma oferecendo um conjunto diferente de características e ferramentas para satisfazer as necessidades de diferentes tipos de equipas e de projetos de desenvolvimento.

3.2.3 Docker

Docker [8] é uma plataforma de código aberto para a automatização da implantação, dimensionamento e gestão de aplicações embaladas em *containers* . Os *containers* são uma forma de embalar e isolar aplicações e as suas dependências para poderem funcionar de forma consistente em qualquer sistema, sem serem afetados pelo ambiente subjacente.

O Docker facilita aos programadores a construção, teste e implementação de aplicações, proporcionando um ambiente consistente para a aplicação funcionar, e fornecendo ferramentas para a gestão de contentores e as aplicações que nele correm. Isto ajuda a racionalizar o processo de desenvolvimento e implementação, tornando-o mais rápido e fiável.

3.2.4 PostgreSQL

PostgreSQL [9] é um sistema de gestão de base de dados relacional de código aberto conhecido pelo seu forte suporte de tipo de dados, fiabilidade, e características que satisfazem as necessidades de aplicações complexas e exigentes. Fornece características *SQL* padrão para criar e gerir objetos de base de dados, bem como tipos de dados avançados, tais como matrizes e JSON. Tem também características de segurança robustas, desempenho rápido, e uma poderosa indexação e gestão de transações. O PostgreSQL é amplamente utilizado para aplicações web e móveis, armazenamento de dados, *business intelligence*, e análise de dados em larga escala e processamento de dados geo-espaciais.

3.2.5 Keycloak

O Keycloak [10] é uma solução *open-source* de gestão de identidade e acesso. Fornece funcionalidades tais como a autenticação de utilizadores, autorização e segurança para aplicações e serviços. O Keycloak é uma solução *Identity and Access Management* (IAM) autónoma que pode ser integrada em várias estruturas e plataformas.

Oferece recursos como *Single Sign-on* (SSO), autenticação multifator e gestão de utilizadores. Fornece uma consola de administração fácil de utilizar para gerir utilizadores, funções e permissões e suporta normas como *OAuth 2.0* e *OpenID Connect*, tornando-o interoperável

com outros fornecedores e sistemas de identidade.

Embora possa ser utilizado de forma independente, o Keycloak pode ser integrado em aplicações e estruturas existentes, tais como, *Java Spring*, *Node.js* e .NET, que fornecem bibliotecas e plugins para integrar o Keycloak para fins de autenticação e autorização.

3.2.6 Ocelot .NET

O *Ocelot* [11] é uma biblioteca de código aberto .NET que funciona como um *API Gateway*. Ao fornecer um ponto de entrada centralizado para que as aplicações consigam aceder aos serviços *backend*, ele simplifica a gestão de arquiteturas de microsserviços. É baseado no ASP.NET Core e pode ser integrado com uma variedade de protocolos. As complexidades de comunicação podem ser abstraídas, as questões transversais podem ser tratadas e as arquiteturas de microsserviços .NET escaláveis e fáceis de manter podem ser construídas com o *Ocelot*.

3.2.7 Rest API

Uma *Representational State Transfer Application Programming Interface* (REST API) [12] é um estilo arquitetural utilizado para o design de serviços web. As REST API permitem a comunicação entre aplicações clientes e aplicações ou serviços no lado do servidor. As principais características das REST API são o uso dos métodos HTTP, tais como *GET, POST, PUT, DELETE*, para executar operações de recursos identificados por URLs, que podem ser objetos, dados ou serviços.

Elas oferecem as operações *Create, Read, Update, Delete* (CRUD) que permite às aplicações do lado do cliente criar, ler, atualizar e apagar recursos. Essas operações são geralmente mapeadas para métodos HTTP específicos. Isso inclui o uso de métodos HTTP padrão, códigos de status e *hypermedia* para navegação entre recursos. Tornou-se o padrão para o design de APIs e fornecem uma maneira padronizada para que as aplicações clientes interajam com aplicações ou serviços no lado do servidor da web.

3.3 Linguagens

3.3.1 HTML

HTML [13], desenvolvido por Tim Berners-Lee em 1990, é uma linguagem de marcação utilizada para criar páginas que estarão disponíveis na *World Wide Web* (WWW). O código HTML assegura a formatação adequada de texto e imagens para o browser, sem HTML, o browser não saberia como exibir o texto e as imagens como elementos. Para ajudar na formatação de uma página, o HTML utiliza o CSS que é utilizado para modificar a sua aparência.

3.3.2 CSS

CSS [14] é uma linguagem utilizada para descrever o aspeto visual e a formatação de uma página web escrita em HTML. Permite aos designers e programadores controlar o *layout*, fonte, cor e outros elementos de design de uma página, facilitando a criação de websites consistentes, atraentes e funcionais. O CSS também torna mais simples a manutenção e atualização de um website, uma vez que as alterações podem ser feitas numa única folha de estilo, em vez de ter de atualizar cada página individual.

3.3.3 TypeScript

TypeScript [15] é uma linguagem de programação que se baseia em JavaScript, acrescentando anotações de tipo opcional e datilografia estática. Isto permite uma melhor manutenção e escalabilidade de grandes aplicações, bem como facilitar a captura de erros de tipo antes do tempo de execução. O TypeScript é de fonte aberta e mantido pela Microsoft, e é amplamente utilizado com populares estruturas e bibliotecas *frontend* e *backend*. O código TypeScript é também totalmente compatível com o código JavaScript.

3.4 Frameworks

3.4.1 Angular

O Angular [16] é uma estrutura *JavaScript* de código aberto bem conhecida, utilizada para criar aplicações web dinâmicas e envolventes. Desenvolvida pela Google, é normalmente utilizada por programadores para construir *Single-Page Application* (SPA) que correm num navegador web.

O Angular utiliza uma estrutura baseada em componentes, o que significa que a aplicação é dividida em componentes menores e reutilizáveis que podem ser facilmente combinados para formar interfaces de utilizador complexas. Esta estrutura simplifica a gestão e manutenção de grandes aplicações e facilita a construção de componentes reutilizáveis do *User Interface* (UI).

A *framawork* também inclui ferramentas e características robustas para a construção de interfaces de utilizador reativas, tais como ligação de dados bidirecionais, diretivas incorporadas, e injeção de dependência. Além disso, oferece um conjunto abrangente de API para ligação a serviços *backend* e tratamento de dados, tornando-a adequada para o desenvolvimento de aplicações orientadas para dados em larga escala.

O Angular foi a framework escolhida para este projeto por ser a framework mais utilizada na empresa e de ter sido a framework aprendida durante a formação. O Angular é uma framework poderosa e versátil que permite criar aplicações web complexas e envolventes.

3.4.2 ASP .NET Core

O ASP.NET Core [17] é uma estrutura livre, de código aberto e multiplataforma para a construção de aplicações web modernas, baseadas na nuvem. É desenvolvido pela Microsoft e faz parte da plataforma.*NET*. O ASP.NET Core foi desenvolvido para ser rápido, leve e escalável, tornando-o ideal para construir uma vasta gama de aplicações, desde websites simples a aplicações *web* complexas e orientadas para os dados.

O ASP.NET Core também inclui uma série de caraterísticas e ferramentas para construir aplicações *web* modernas, tais como uma arquitetura modular, suporte integrado para API *web*, e integração com *frameworks front-end* populares, tais como Angular e React. E também fornece um ambiente seguro e estável para a construção e implementação de aplicações, com caraterísticas tais como proteção integrada contra ameaças de segurança comuns, incluindo o *Cross-Site Scripting* (XSS) e o *Cross-Site Request Forgery* (CSRF).

Capítulo 4

Trabalho Desenvolvido

4.1 PluralSight

Como foi referido nos capítulos anteriores, a estagiária recebeu uma formação em que será feita pelo *PluralSight* e depois da formação foi desafiada a por em práticas competências que adquiriu. Durante a formação na Pluralsight Skills, adquiri competências na *framework* Angular para o *frontend* e no ASP.NET Core para o *backend*, assim como no uso do *Docker* para executar o *backend* e estabelecer a ligação entre o *backend* e o *frontend* utilizando o *Ocelot*. A Pluralsight Skills é uma solução de formação e desenvolvimento que ajuda as organizações a abordar lacunas de competências em áreas críticas, como a cloud, segurança, desenvolvimento de *software* e dados. Além disso, fornece visibilidade sobre competências e papéis, permitindo maximizar a capacidade das equipas existentes e acelerar a entrega de produtos.

Figura 4.1: Logótipo da PluralSight

4.2 Projeto Proposto

Após a formação, foi proposto inicialmente replicar uma aplicação front-end de um comments section do website *Frontend Mentor* utilizando a framework Angular e a framework Material Design. Após o *frontend* estar completo, foi feita a aplicação *backend* para fazer a comunicação com o *frontend*, deixando assim de haver dados estáticos na aplicação.

4.3 Estrutura do Projeto

A projeto é construído por uma *web app* e por uma *server app* que está encapsulada num *Docker Container*, como exemplifica a estrutura mostrada na figura 4.2.

As principais componentes do sistema desenvolvido são:

- *Web App* - A *web app* é a interface a que os utilizadores acedem por um *browser*. Tem uma interface gráfica do utilizador para os utilizadores interagirem com a aplicação. Para solicitar e receber dados do servidor, a *web app* comunica com a *API Gateway* que está a ser executada num *Docker Container*;

- *API Gateway* - A *API Gateway* atua como um ponto de entrada dentro do *Docker Container*. A aplicação gere o encaminhamento dos pedidos, a autenticação e a autorização. O gateway da API transmite os pedidos autenticados para os endpoints do servidor correto e, ao mesmo tempo, aplica as normas de segurança;

- *Server* (ASP.NET Core) - O servidor processa os pedidos do cliente enquanto é executado num *Docker Container*. Aceita pedidos da *API Gateway*, executa as tarefas necessárias e, em seguida, entrega os resultados. O serviço de autenticação e a base de dados são os dois principais serviços com os quais o servidor interage;

- *Auth* - A autenticação e a permissão do utilizador são tratadas pelo serviço de autenticação, que também está a funcionar dentro do *Docker Container*. Ele garante que apenas os utilizadores que foram autorizados possam aceder a recursos protegidos e realizar determinadas tarefas. As credenciais do utilizador são armazenados na própria base de dados do serviço de autenticação;

- Armazenamento de Dados - A aplicação utiliza duas bases de dados diferentes, contidas no *Docker Container*:

  - Base de dados da aplicação - Local onde se encontram de forma persistente os dados da aplicação, como por exemplo, os *Chats* e os detalhes dos utilizadores;
  - Base de dados de autenticação - Esta base de dados destina-se apenas ao serviço de autenticação e é utilizada para armazenar informações relacionadas com o utilizador, tais como credenciais. Apenas o serviço de autenticação que opera no *Docker Container* tem acesso e controlo sobre esta base de dados.

Ao encapsular todo o sistema (*backend*) num *Docker Compose* agilizam-se as tarefa de replicar, implantar, escalar e gerir a aplicação, mantendo a sua consistência e minimizando os problemas de compatibilidade em diferentes ambientes, podendo assim ter diferentes pontos de acesso aos dados e à lógica da aplicação.

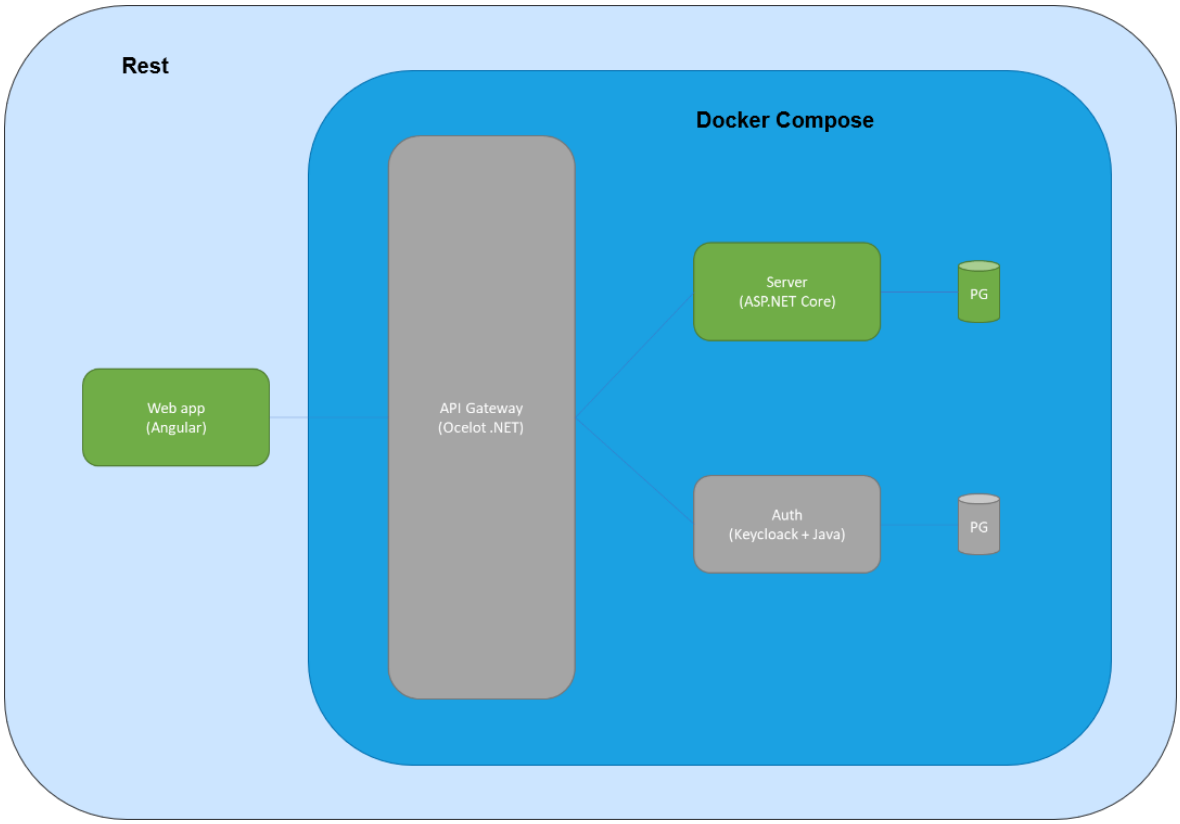

Figura 4.2: Representação esquemática da estrutura do projeto.

4.4 Engenharia de Software

Neste capítulo tratamos de toda a engenharia de *software* correspondente à aplicação, tais como os requisitos funcionais e não funcionais e ainda os diagramas, nomeadamente os de sequência e de atividade.

4.4.1 Análise de Requisitos

4.4.1.1 Requisitos Funcionais

Na tabela 4.1 são descritos os requisitos funcionais. A tabela encontra-se dividida em duas colunas, a coluna de requisitos e a coluna da descrição de cada requisito. A coluna de requisitos descreve o objetivo do requisito. A coluna de descrição fornece mais detalhes sobre o requisito, incluindo as restrições e as condições de aceitação.

Requisitos	Descrição
RF01	A aplicação deve permitir que um utilizador se registe, fornecendo pelo menos um nome de utilizador, um endereço de e-mail e uma palavra-passe.
RF02	A aplicação deve permitir que um utilizador faça *login* fornecendo o seu nome de utilizador e palavra-passe.
RF03	A aplicação permite que um *chat* seja apagado.
RF04	A aplicação deve permitir que os utilizadores enviem mensagens entre si.
RF05	A aplicação deve permitir que um utilizador termine a sua sessão.

Tabela 4.1: Requisitos Funcionais

4.4.1.2 Requisitos Não Funcionais

Na tabela 4.2 são descritos os requisitos não funcionais. A tabela encontra-se dividida em duas colunas, a coluna de requisitos e a coluna da descrição de cada requisito.

Requisitos	Descrição
RNF01	A aplicação é compatível com os seguintes *browsers*: *Google Chrome*, *Microsoft Edge*.
RNF02	A aplicação deve ter uma conexão à Internet
RNF03	A aplicação precisa de ter uma conexão à base de dados para guardar os dados dos utilizadores, as mensagens e os *chats*. A base de dados deve ser segura e confiável.
RNF04	A aplicação apresenta uma interface *user-friendly*. A interface deve ser fácil de usar e navegar.

Tabela 4.2: Requisitos Não Funcionais

4.4.2 Diagramas

4.4.2.1 Diagrama de Sequência

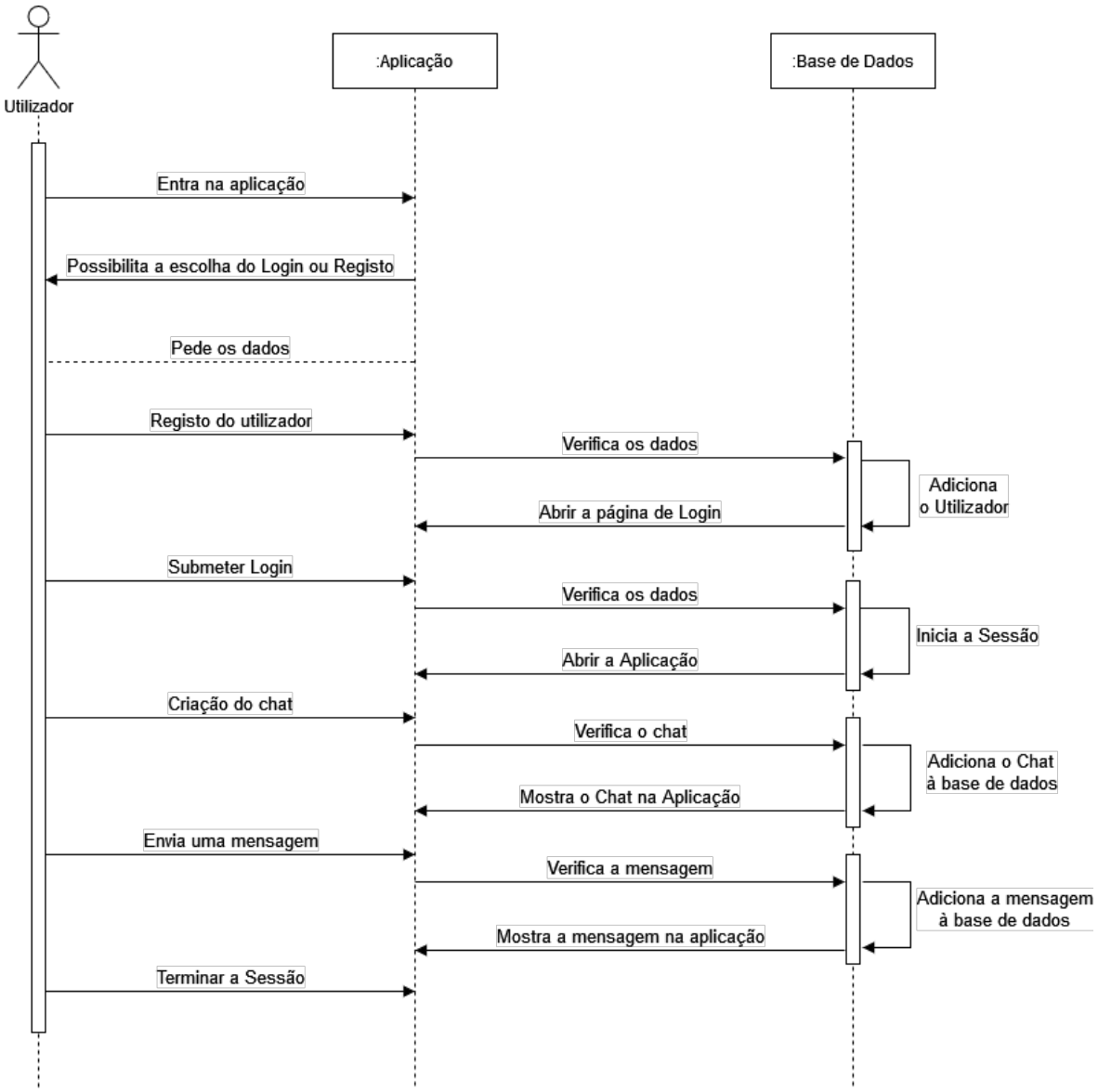

Figura 4.3: Diagrama de Sequência do Registo, *Login*, Criação do *chat* e envio de mensagens

Na figura 4.3, podemos observar o diagrama de sequência onde é demonstrado o processo do Registo e *Login* do utilizador. Após o registo, as credenciais do utilizador vão ser guardadas numa base de dados e seguidamente, o utilizador poderá fazer o *login* com esses dados. Após o *login*, o utilizador irá para a página inicial da aplicação e poderá criar os *chats* com os utilizadores que já estiverem com as suas credenciais guardadas na base de dados e depois consegue mandar mensagens nesse *chat* e essas irão ser verificadas e guardadas na base de dados e seguidamente, a mensagem irá ser mostrada no *chat*. No fim, o utilizador poderá terminar a sua sessão.

4.4.2.2 Diagrama de Atividade

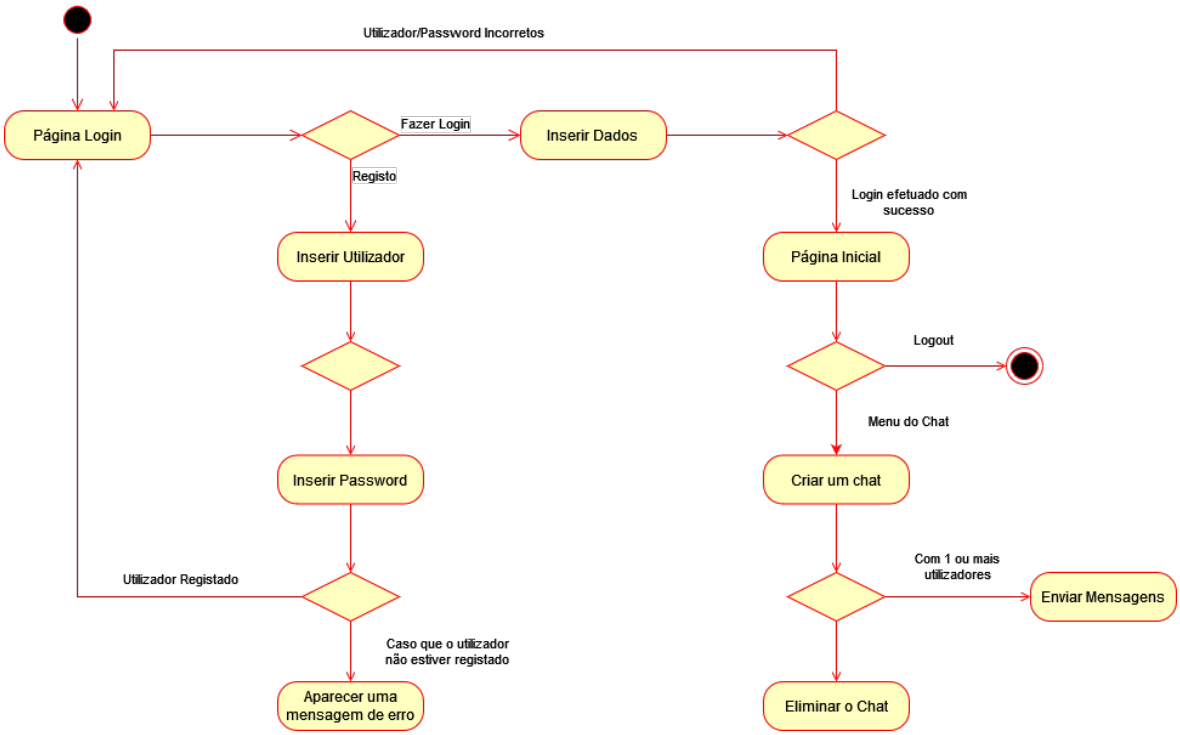

Figura 4.4: Diagrama de Atividade do Registo, *Login*, Criação do *chat* e envio de mensagens

Na figura 4.4, podemos observar o diagrama de atividade relativo aos processos de Registo, *Login* e o envio de mensagens. Na página do *login*, o utilizador tem duas opções, fazer o *login* ou fazer o registo. Caso que o utilizador não tenha feito o registo e tentou fazer o *login*, irá aparecer uma mensagem de erro. Para efetuar o registo, o utilizador tem que inserir o seu nome e a sua *password* e no fim do registo, o utilizador voltará à página do *login*. Ao efetuar o *login*, caso o *username* ou a *password* estejam incorretos, aparecerá uma mensagem de erro a dizer que as credenciais estão incorretas. Após o *login* ser efetuado com sucesso, o utilizador irá ser redirecionado para uma página em que poderá criar um *chat* com um ou mais utilizadores, aceder aos *chats*, eliminar esses *chats* e ter a opção de fazer *logout*.

4.5 Configuração da Aplicação

Para conseguir configurar a aplicação localmente, foi necessário fazer os seguintes passos:

- Foi necessário configurar o *Keycloak* para se conseguir conectar à aplicação e para isso foi preciso entrar na consola do mesmo para fazer as configurações necessárias, através da consola de Administrador, conforme mostrado na figura 4.5;

- No *Keycloak*, criou-se um *realm* para depois conectar à *API Gateway* a partir de um ficheiro *JSON*, conforme mostrado na figura 4.6;

- Para correr o *frontend*, teve-se que instalar o Angular na diretoria da aplicação e seguidamente correr o comando *npm install* para instalar a pasta dos *node_modules*. Para terminar a instalação dos *node_modules*, o utilizador executa o comando *ng serve* para correr o *frontend* na porta 4200;

- Para correr o *backend*, o utilizador tem que ter instalada a aplicação *Docker Desktop* e em seguida ir à diretoria do ficheiro *docker compose* e executar o comando *docker--compose -f "docker-compose.dev.yml" up –force-recreate –build*. Por fim, a aplicação irá abrir num *browser* no *URL localhost:4200* e será apresentada a página de *Login* da aplicação. Após o utilizador ter feito o Registo ou *Login*, este será redirecionado para a página inicial da aplicação.

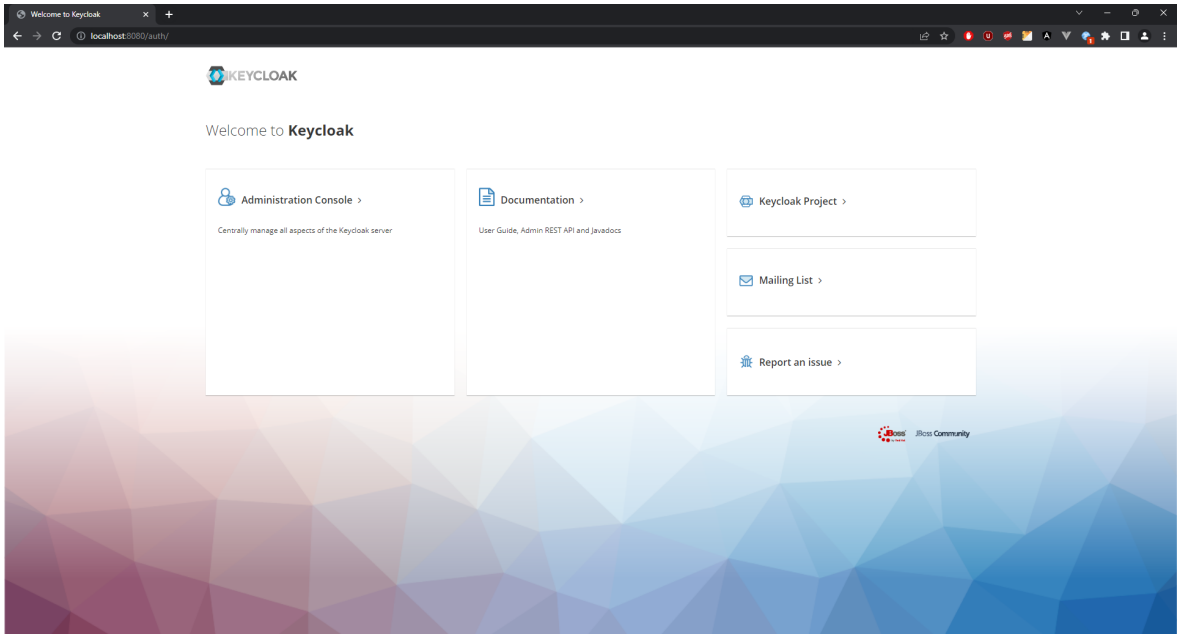

Figura 4.5: Página do *Keycloak*

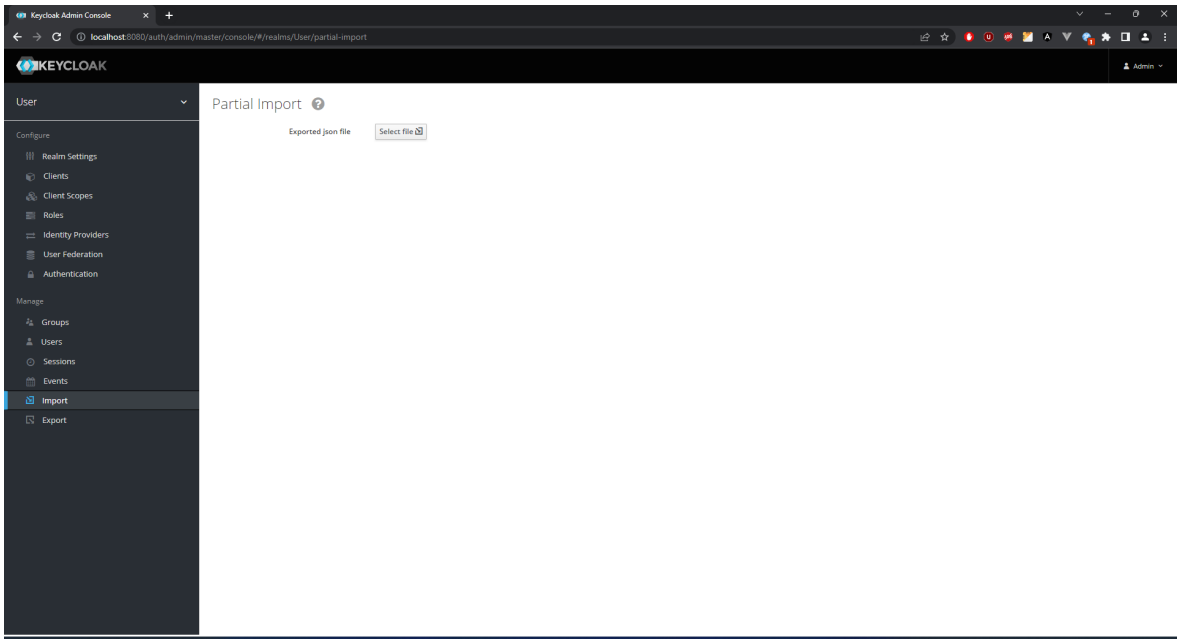

Figura 4.6: Importar ficheiro *JSON* para a criação do *realm*

4.6 Interface da Aplicação

Como foi dito na secção anterior, quando o utilizador inicia a aplicação, este irá para a página de *login* para se autenticar ou até mesmo para realizar o registo. A página de *login* é composta por dois campos: um campo para o nome de utilizador e outro para a palavra-passe. Após o utilizador introduzir os seus dados, pode clicar no botão "Login" para entrar na aplicação. Se o utilizador não tiver uma conta, pode clicar no botão "Registar" para criar uma conta.4.7.

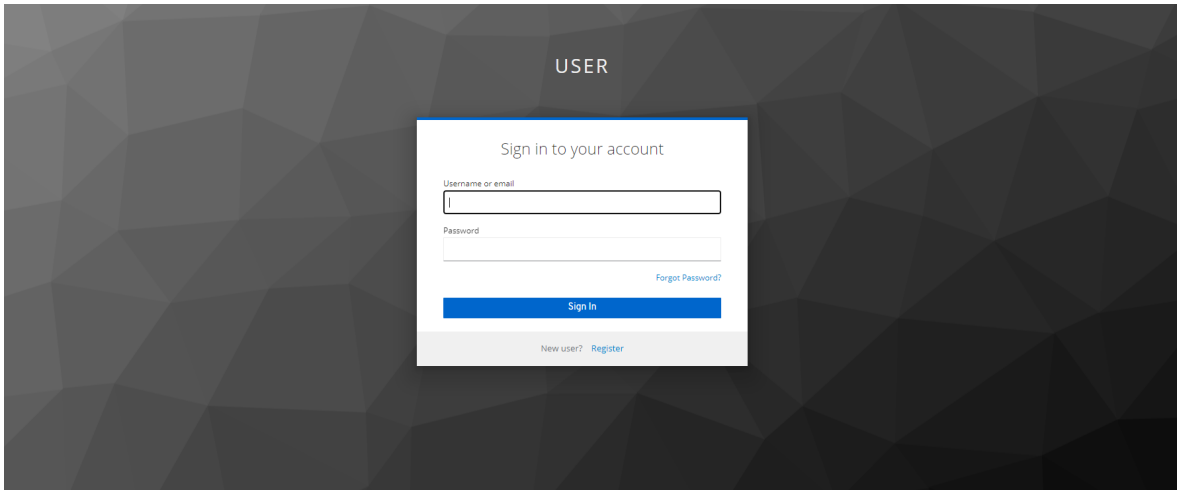

Figura 4.7: Página de Login

A página inicial, mostrada na figura 4.8, é composta por uma lista de *chats*, um botão para fazer *logout* e uma barra lateral. A lista de chats mostra os chats em que o utilizador está a participar. O botão para fazer *logout* permite que o utilizador saia da aplicação. A barra lateral, mostrada na figura 4.9, mostra os chats em que o utilizador não está a participar. O utilizador pode clicar no botão ”Criar chat” para criar um chat. O utilizador pode clicar no botão ”Apagar chat” para apagar um chat.

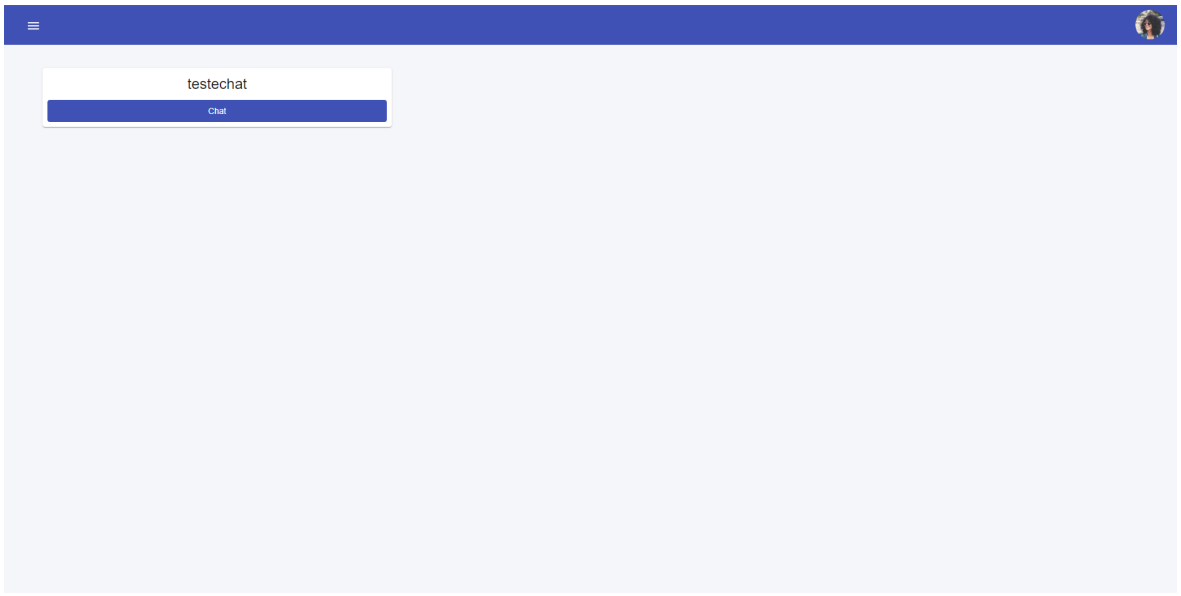

Figura 4.8: Página Inicial

A partir da página inicial, o utilizador consegue ver os chats que ele pode participar e quando ele carrega num dos chats irá ser redirecionado à página dos chats e aí ele pode interagir com um ou mais utilizadores. O utilizador pode enviar mensagens, apagar a mensagem e pode editar a mensagem, como mostra a figura 4.10.

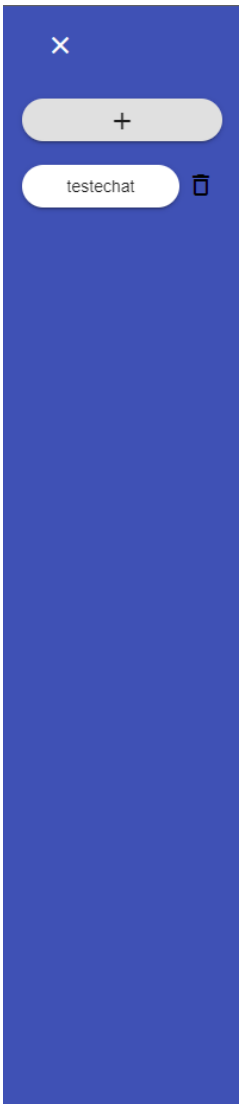

Figura 4.9: Barra Lateral dos *Chats*

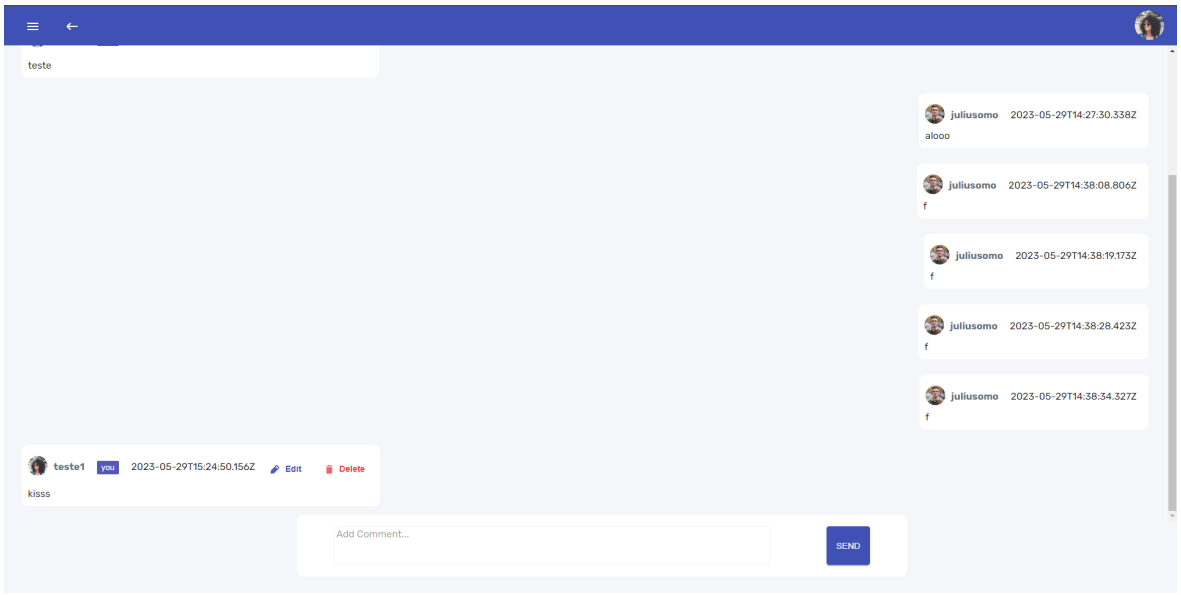

Figura 4.10: Página do *Chat*

Capítulo 5

Conclusão

Este estágio foi uma experiência verdadeiramente enriquecedora e transformadora para o meu futuro. Durante o período do estágio, adquiri experiência prática na minha área de estudo e aprendi com profissionais experientes na indústria. A orientação e apoio fornecidos pelos meus mentores, colegas de trabalho, amigos e família foram inestimáveis para o meu crescimento e desenvolvimento como profissional.

Uma das mais significativas aquisições do meu estágio foi ter a oportunidade de trabalhar em projetos e desafios do mundo real. Isso proporcionou-me uma compreensão mais profunda da indústria e permitiu aplicar as teorias e conceitos que aprendi na sala de aula a cenários práticos. Estou grata pela oportunidade de aprender com os meus erros e de receber feedback construtivo dos meus mentores e colegas de trabalho.

De modo geral, este estágio foi uma oportunidade de aprendizagem fenomenal que me ajudou a obter uma apreciação mais profunda do trabalho e da indústria. Uma parte fundamental desse processo foi o desenvolvimento da aplicação, onde pude aplicar os conhecimentos e habilidades adquiridos durante a minha formação. Através da utilização de *frameworks* como o *Angular* e o *ASP.NET Core*, bem como do Docker para a gestão de conteinerização e do *Ocelot* para a ligação entre o *frontend* e o *backend*, pude desenvolver uma aplicação funcional e aplicar boas práticas de desenvolvimento.

Estou confiante de que as competências, conhecimentos e experiências que adquiri durante este estágio serão fundamentais para os meus esforços futuros. Estou entusiasmada por aplicar essas aprendizagens em projetos futuros e contribuir para o sucesso da indústria em que estou inserida.

Bibliografia

[1] Latitudde. (2023) Latitudde. Último acesso a 6 de fevereiro de 2023. [Online]. Available: https://latitudde.com/ 3

[2] Google. (2023) Introduction to the angular docs. Último acesso a 6 de fevereiro de 2023. [Online]. Available: https://angular.io/docs 4

[3] Microsoft. (2023) Overview of asp.net core. Último acesso a 6 de fevereiro de 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0 4

[4] Abhinav Asthana, Ankit Sobti and Abhijit Kane. (2023) Postman. Último acesso a 8 de fevereiro de 2023. [Online]. Available: https://www.postman.com/company/about-postman/ 4

[5] Linus Torvalds and Junio Hamano. (2023) git –distributed-even-if-your-workflow--isnt. Último acesso a 8 de fevereiro de 2023. [Online]. Available: https://git-scm.com/ 4

[6] Martin Heller. (2022) What is visual studio code? microsoft's extensible code editor. Último acesso a 8 de fevereiro de 2023. [Online]. Available: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html 7

[7] Kolade Chris. Visual studio vs visual studio code – what's the difference between these ide code editors? Último acesso a 9 de fevereiro de 2023. [Online]. Available: https://www.freecodecamp.org/news/visual-studio-vs-visual-studio-code/ 7

[8] Docker Inc. (2023) Docker overview. Último acesso a 9 de fevereiro de 2023. [Online]. Available: https://docs.docker.com/get-started/overview/ 8

[9] AWS. (2023) What is postgresql? Último acesso a 10 de fevereiro de 2023. [Online]. Available: https://aws.amazon.com/rds/postgresql/what-is-postgresql/ 8

[10] Bartłomiej Żyliński. (2021) What keycloak is and what it does? Último acesso a 28 de maio de 2023. [Online]. Available: https://dzone.com/articles/what-is-keycloak-and-when-it-may-help-you 8

[11] Alberto Moreno. (2023) Building api gateway on .net with ocelot. Último acesso a 05 de junho de 2023. [Online]. Available: https://arbems.com/en/building-api-gateway-on-net-with-ocelot/ 9

[12] Jamie Juviler. (2022) Rest apis: How they work and what you need to know. Último acesso a 05 de junho de 2023. [Online]. Available: https://blog.hubspot.com/website/what-is-rest-api 9

[13] Tomas Laurinavicius. What is html? Último acesso a 10 de fevereiro de 2023. [Online]. Available: https://webflow.com/blog/what-is-html 9

[14] Artūras B. (2023) What is css and how does it work? Último acesso a 10 de fevereiro de 2023. [Online]. Available: https://www.hostinger.com/tutorials/what-is-css 10

[15] Meredith Shubel. (2022) What is typescript? Último acesso a 10 de fevereiro de 2023. [Online]. Available: https://thenewstack.io/what-is-typescript/ 10

[16] Google. (2022) What is angular? Último acesso a 10 de fevereiro de 2023. [Online]. Available: https://angular.io/guide/what-is-angular 10

[17] Under The Hood Learning. (2021) Asp.net core: What is it? Último acesso a 10 de fevereiro de 2023. [Online]. Available: https://www.underthehoodlearning.com/asp-net-core-what-is-it/ 11