Desenvolvimento *Fullstack* de uma Plataforma Web

Nuno Miguel Freire Monteiro

Relatório do Projeto de Estágio
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Tiago Miguel Carrola Simões

Outubro de 2025

Universidade da Beira Interior, Covilhã 10/10/2025

Declaração de Integridade

Eu, Nuno Miguel Freire Monteiro, que abaixo assino, estudante com o número de inscrição M13614 do 2º ciclo 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, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 11/06/2025

Assinado por: **Nuno Miguel Freire Monteiro**
Num. de Identificação: 15888431
Data: 2025.10.09 16:13:24+01'00'

Agradecimentos

Gostaria de começar por agradecer ao Professor Doutor Tiago Miguel Carrola Simões por aceitar ser o meu orientador, e por toda a disponibilidade, acompanha- mento e contributos prestados, especialmente na redação deste relatório, sem os quais a sua concretização teria sido impossível. Agradeço igualmente ao Engenheiro João Gouveia, Líder Técnico na Latitudde, pelo acompanhamento e orientação técnica ao longo dos projetos desenvolvidos durante o estágio curricular. Estendo o meu agradecimento ao Grupo RIT e, em particular, a todas as pessoas com quem tive o privilégio de interagir durante o estágio, com especial destaque para a equipa da Latitudde, em especial ao escritório do Fundão, pelo ambiente, companheirismo, constante disponibilidade para ajudar e por me fazerem sentir integrado na empresa. Agradeço profundamente à minha família, pelo apoio incondicional que sempre me deu, não apenas nesta etapa final do percurso académico, mas ao longo de toda a minha vida. Sem esse suporte, não teria alcançado tudo o que alcancei até hoje. Aos meus amigos, e em particular aos mais próximos. A vossa companhia nos bons momentos e, sobretudo, nos dias mais difíceis, foram essenciais para manter-me motivado ao longo deste percurso. A todos, o meu sincero obrigado.

Resumo

Este relatório de estágio descreve o trabalho realizado durante o estágio na empresa Latitudde. O documento dá uma visão geral da jornada do estagiário como um Full Stack Developer na empresa Latitudde - Digital Enablers, LDA, descrevendo os obstáculos enfrentados e as lições aprendidas durante todo o percurso.
Este estágio focou-se no desenvolvimento de um sistema de transportes flexíveis para o distrito de Castelo Branco. Este sistema tem como objetivo facilitar o transporte em zonas rurais, permitindo aos residentes o acesso aos transportes regulares.
De forma sucinta, o sistema permite que os utilizadores realizem pedidos de viagem entre um ponto rural e um ponto regular (onde passem transportes públicos, tal como autocarros). De seguida, um algoritmo irá alocar os pedidos a motoristas particulares, mais especificamente a taxistas.
O estágio englobou todo o processo da construção de uma aplicação, desde o levantamento de requisitos, a modelagem da base de dados, a formação nas tecnologias necessárias, o desenvolvimento de *frontend* e *backend*, e por fim o suporte após a entrega da aplicação ao cliente.
As tecnologias utilizadas foram maioritariamente Angular para o *frontend*, e ASP.NET Core para o *backend*, duas tecnologias muito utilizadas em vários projetos na empresa.

Palavras-chave

Angular
ASP.NET core
Backend
Base de dados
Castelo Branco
Desenvolvimento de Software
Engenharia Informática
Estágio
Frontend
Full Stack Developer
Full Stack Development
PostgreSQL
Transportes Flexíveis

Abstract

This internship report describes the work carried out during the internship at the company Latitudde. The document provides an overview of the trainee's journey as a Full Stack Developer at Latitudde - Digital Enablers, LDA, describing the obstacles faced and the lessons learned throughout the entire process.
This internship focused on the development of a flexible transport system for the district of Castelo Branco. The goal of this system is to facilitate transportation in rural areas, allowing residents access to regular transport services.
In short, the system allows users to make travel requests between a rural point and a regular point (where public transport, such as buses, passes). Afterwards, an algorithm will allocate the requests to private drivers, more specifically taxi drivers.
The internship encompassed the entire process of building an application, from requirements gathering, database modeling, training in the necessary technologies, the development of *frontend* and *backend*, and finally the support provided after delivering the application to the client.
The technologies used were mainly Angular for the *frontend*, and ASP.NET Core for the *backend*, two technologies widely used in several projects within the company.

Keywords

Angular
ASP.NET core
Backend
Base de dados
Castelo Branco
Computer Science
Flexible Transports
Frontend
Full Stack Developer
Full Stack Development
Internship
PostgreSQL
Software Development

Contents

List of Figures

Lista de Acrónimos

API *Application Programming Interface*

ASP.NET *Active Server Pages for .NET*

CSS *Cascading Style Sheets*

CVRP Capacitated Vehicle Routing Problem

DTO *Data Transfer Objects*

HTML *HyperText Markup Language*

HTTP *Hypertext Transfer Protocol*

i18n *Software Internationalization*

IDE *Integrated Development Environment*

JWT *JSON Web Token*

LINQ *Language Integrated Query*

MVC Model View Controller

ORM *Object-Relational Mapping*

OSRM *Open Source Routing Machine*

SQL *Structured Query Language*

TPF *Transportes Públicos Flexíveis*

TSP Travelling Salesman Problem

UBI Universidade da Beira Interior

VPN *Virtual Private Network*

SPA *Single Page Application*

JSON *JavaScript Object Notation*

Chapter 1

Introdução

1.1 Apresentação do Estágio

O estágio, com duração de 9 meses, teve como foco o desenvolvimento integral de um projeto a partir de uma proposta inicial fornecida pelo cliente. Durante este período, foi necessário percorrer todas as etapas fundamentais do ciclo de vida de um projeto de software, desde a compreensão dos requisitos até á entrega final e suporte pós-implementação.
Nas seguintes subsecções vão ser apresentados os objetivos gerais, a motivação e a relevância do estágio.

1.2 Objetivos Gerais

O projeto consiste em um sistema de mobilidade multimodal, permitindo a deslocação por transportes públicos acessíveis a pessoas em zonas remotas ou rurais. Um cliente que queira usufruir dos serviços terá de registar um pedido com origem e destino através da plataforma online ou por telefone, com o limite de às 15:00 do dia anterior à reserva. De seguida, o pedido será validado por funcionários da câmara de Castelo Branco, que poderão aceitar ou recusar o pedido, fornecendo sempre uma justificação para tal. De seguida, um algoritmo irá atribuir as viagens aos condutores. Caso algum pedido não seja possível ser alocado, este é reagendado para o dia seguinte.
Para atingir o objetivo geral, foram estabelecidos os seguintes objetivos secundários:

- **Formação nas tecnologias e *frameworks***: A formação permite a adquirição de conhecimentos práticos nas ferramentas a utilizar, garantindo assim uma implementação eficiente e com boas práticas.
- **Levantamento e análise de requisitos:** A identificação e descrição dos requisitos funcionais e não funcionais do sistema permite garantir que todas as necessidades do cliente sejam compreendidas e atendidas, facilitando assim também a divisão de tarefas entre os elementos da equipa.
- **Planeamento e modelagem da base de dados:** A estruturação de um esquema relacional eficiente e otimizado permitiu atender a todas as funcionalidades propostas pelo sistema.
- **Gestão de tarefas e planeamento de sprints:** A divisão do trabalho em etapas organizadas e distribuídas ao longo do ciclo de desenvolvimento, com foco na entrega incremental de funcionalidades, permitiu realizar um desenvolvimento eficiente por parte da equipa.

- **Desenvolvimento e integração do sistema:** A construção do *frontend* e *backend* da aplicação, integrando as funcionalidades necessárias para atender aos requisitos especificados pelo cliente.

- **Suporte e manutenção:** A prestação de assistência ao cliente após a entrega do projeto, garantindo o bom funcionamento do sistema ao longo da sua utilização, e o esclarecimento de dúvidas surgidas por parte do cliente.

1.3 Motivação e Relevância

Este projeto surgiu como uma tentativa de fornecer os serviços de transporte público a zonas remotas, sem a necessidade de existirem transportes diários. Com esta abordagem, verifica-se uma poupança nos gastos por parte da câmara de Castelo Branco, e uma redução no impacto ambiental.

1.4 Metodologia de Trabalho

Durante o estágio, foi adotada a metodologia Agile, que se caracteriza por um conjunto de práticas e princípios que promovem a flexibilidade e a colaboração entre os membros da equipa. Esta metodologia consistiu em reuniões diárias, conhecidas como ”dailies”, onde os restantes elementos da equipa se reuniam para esclarecer dúvidas, discutir obstáculos e realizar atualizações sobre o progresso das tarefas que foram assignadas a cada membro. Além disso, foi realizada uma demonstração do progresso a cada duas semanas, designada de ”sprint review”, onde se apresentavam os resultados alcançados e se planeava o trabalho para as próximas duas semanas.

1.5 Organização do Documento

Este relatório está organizado de forma a guiar o leitor através das diferentes etapas do projeto a desenvolver durante o estágio, desde a contextualização do problema até a implementação e conclusão do sistema. A estrutura do documento é a seguinte:

Capítulo 1: Introdução

Este capítulo apresenta uma visão geral do estágio, destacando os objetivos gerais, a motivação e a relevância do projeto. Além disso, são descritos os objetivos secundários que orientaram o desenvolvimento do sistema de mobilidade multimodal.

Capítulo 2: Descrição da Empresa

Neste capítulo, é apresentada uma breve descrição da empresa *Latitudde - Digital Enablers, LDA*, incluindo um pouco da sua história, as suas principais atividades, projetos e tecnologias utilizadas. São também destacados os serviços oferecidos pela empresa e o seu papel no contexto do estágio.

Capítulo 3: Plano de Estágio

Este capítulo descreve o planeamento e o cronograma do estágio, detalhando as atividades a realizar ao longo dos nove meses de duração do projeto. São apresentados os métodos de gestão de tarefas, tais como o uso de metodologias ágeis (Scrum), e a divisão do trabalho em *sprints*.

Capítulo 4: Background e Estado da Arte

Aqui é discutido o contexto do projeto, com foco no conceito de transportes flexíveis e na sua aplicação em zonas rurais. São também abordados os algoritmos de rotas utilizados e as tecnologias emergentes relacionadas ao tema.

Capítulo 5: Atividades Desenvolvidas

Este capítulo detalha as atividades a realizar durante o estágio, desde a formação nas tecnologias até ao desenvolvimento do projeto. São descritas as metodologias aplicadas, as tarefas diárias e semanais, e a relação entre o aprendizado académico e a prática profissional.

Capítulo 6: Tecnologias e Ferramentas de Desenvolvimento

Neste capítulo, são apresentadas as principais tecnologias e ferramentas utilizadas no desenvolvimento do projeto, incluindo linguagens de programação, *frameworks*, bibliotecas e sistemas de gestão de bases de dados.

Capítulo 7: Engenharia de Software

Este capítulo descreve os requisitos do sistema, tanto funcionais (o que o sistema deve fazer) quanto não funcionais (desempenho, segurança, usabilidade, etc.), que orientaram o desenvolvimento do projeto.

Capítulo 8: Implementação da Plataforma Web Este capítulo descreve de forma detalhada o processo de implementação técnica da plataforma desenvolvida. São apresentados os principais aspetos relacionados com o desenvolvimento do frontend e do backend, bem como a integração entre ambos. Além disso, são discutidas as decisões tecnológicas adotadas, as bibliotecas e ferramentas utilizadas, e os desafios enfrentados durante o desenvolvimento. O capítulo evidencia ainda a arquitetura implementada, as soluções aplicadas para garantir eficiência e segurança, e a comunicação entre os diferentes componentes do sistema.

Capítulo 9: Demonstração e Validação da Plataforma Web Neste capítulo é apresentada a demonstração prática da plataforma desenvolvida, evidenciando as principais funcionalidades implementadas no frontend e no backoffice. São detalhados os módulos públicos e privados, as interfaces de utilizador, os processos de autenticação, as operações de reserva e gestão de viagens, bem como o sistema de notificações. Este capítulo inclui ainda exemplos visuais e explicações sobre o funcionamento global da aplicação, demonstrando o cumprimento dos requisitos definidos e validando o sistema junto do cliente.

Capítulo 10: Conclusão e Trabalho Futuro O último capítulo apresenta as conclusões finais sobre o estágio e o projeto desenvolvido. São destacadas as aprendizagens adquiridas, as competências técnicas e pessoais desenvolvidas, bem como a relevância do trabalho realizado para o contexto profissional. O capítulo inclui ainda uma reflexão sobre os contributos do estagiário para a equipa e para o projeto, e propõe possíveis melhorias e evoluções futuras da plataforma, com o objetivo de ampliar as suas funcionalidades e otimizar o seu desempenho.

Chapter 2

Descrição da Empresa

2.1 Introdução

Neste capítulo vai ser descrita uma breve apresentação da empresa, bem como a sua história e os principais projetos desenvolvidos por esta.

2.2 Resumo da Empresa

A Latitudde integra o Grupo ReadinessIT, uma empresa multinacional especializada em projetos de Transformação Digital. Com uma equipa composta por mais de 400 colaboradores, a empresa possui escritórios distribuídos por três continentes, localizados em Portugal, Chile, Peru e Nova Zelândia, e desenvolve projetos de alcance internacional.

2.3 Ambiente de Trabalho

A semana do estagiário consistiu em três dias presenciais: segundas, quartas e sextas, enquanto os restantes dias úteis foram realizados remotamente. No entanto, tanto no escritório quanto de forma remota, ocorreram as habituais reuniões, as quais foram realizadas através da plataforma ”Teams” da Microsoft nos dias remotos.

2.4 Breve História da Organização

A Latitudde é uma empresa recente, tendo sido constituída no início de 2021. Esta faz parte do grupo RIT, sendo este estabelecido na região da Beira Interior em 2015. O estabelecimento desta empresa marcou a origem de um novo método de trabalho: o trabalho remoto. A Latitudde tem como objetivos principais a integração no mundo empresarial e o desenvolvimento rápido de software.

2.5 Principais Atividades e Projetos da Empresa

A empresa especializa-se na realização de soluções digitais adequadas às necessidades do cliente, desde a conceção (planeamento e *design*) até à fase final (manutenção e suporte), oferecendo assim o chamado *End-to-End projects*.
Alguns dos mais conhecidos clientes da empresa são:

- Águas de Portugal

- Outsystems
- NOS
- Bio2
- Team Resilience

2.6 Serviços e Tecnologias

A empresa disponibiliza uma ampla gama de serviços tecnológicos, conseguindo assim atender a todas as possíveis demandas específicas a cada projeto. Desde soluções rápidas e flexíveis até arquiteturas complexas, os serviços garantem inovação, escalabilidade e integração eficiente.
Os principais serviços oferecidos são:

- Low-code
- Backend
- Frontend
- Full-Stack

Para a implementação dos seus serviços, a empresa recorre a um conjunto de tecnologias de ponta, garantindo a entrega de soluções robustas, seguras e escaláveis.
As principais tecnologias utilizadas são:

- SQL Server
- AWS
- Outsystems
- Java
- .NET
- Angular
- PHP
- Flutter

2.7 Conclusão

Como podemos observar neste capítulo, a Latitudde, parte integrante do Grupo ReadinessIT, é uma empresa dinâmica e inovadora no setor de Transformação Digital, com uma atuação global e uma equipa diversa e qualificada. Desde a sua fundação em 2021, a empresa tem se destacado pela entrega de soluções digitais completas e adaptadas às necessidades dos seus clientes, com um foco especial em metodologias de trabalho ágeis. Com uma gama de serviços que abrange desde o design até a manutenção, a Latitudde tem conquistado importantes clientes.

Chapter 3

Plano de Estágio

3.1 Introdução

Este capítulo apresenta uma visão detalhada sobre o plano e os objetivos definidos para o estágio curricular na empresa, destacando as atividades realizadas, os desafios que surgiram e as competências que foram adquiridas ao longo do período. O estágio teve como foco o desenvolvimento integral de um projeto a partir de uma proposta inicial fornecida pelo cliente. Durante este processo, foi necessário percorrer todas as etapas fundamentais do ciclo de vida de um projeto de software, desde a compreensão dos requisitos até à entrega final e suporte pós-implementação. Para além da parte técnica, o estágio facilitou a aplicação de boas práticas de gestão, incluindo o planeamento de *sprints*, a divisão eficiente de tarefas e a comunicação contínua com o cliente. Com uma carga horária diária das 9h às 18h em dias úteis, esta experiência proporcionou uma imersão completa no ambiente empresarial e desempenhou um papel crucial na consolidação de competências essenciais para a integração no mercado de trabalho.

3.2 Planeamento

O objetivo principal do estágio consistiu na realização de um projeto de raiz. Com apenas o documento com a proposta do cliente, foram realizados todos os passos necessários para a execução do projeto (exceto a construção dos *designs*), incluindo:

- Formação nas *frameworks* utilizadas para o projeto
- Descrição dos requisitos funcionais e não funcionais;
- Planeamento e construção do esquema da base de dados relacional;
- Desenvolvimento do *Backend*;
- Desenvolvimento do *Frontend*;
- Suporte ao cliente após a finalização do projeto.

3.3 Cronograma

O estágio teve início no dia 05 de Fevereiro e terminou no dia 12 de Junho. O estagiário trabalhou das 9h às 18h nos dias úteis da semana 3.1

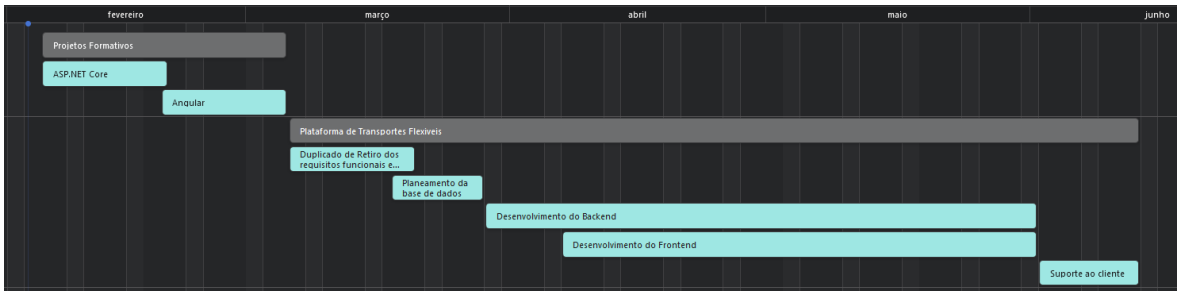

Figure 3.1: Cronograma das atividades desenvolvidas durante o estágio.

3.4 Conclusão

Neste capítulo pudemos observar um pouco do plano de estágio. Este plano descreveu as atividades, objetivos e desafios de um estágio curricular com duração de 9 meses, cujo foco foi o desenvolvimento de um projeto a partir de uma proposta inicial fornecida pelo cliente. Foram abrangidas todas as etapas do ciclo de vida de um projeto de software, desde a compreensão dos requisitos, o desenvolvimento do *backend* e *frontend*, a gestão de tarefas, e o suporte pós-implementação. Isto permitiu ao estagiário aprender a aplicar boas práticas de gestão, como o planeamento de sprints e a comunicação contínua com o cliente.
O estágio teve início no dia 05 de fevereiro de 2025 e terminou em 12 de junho de 2025, com uma carga horária diária de 9h às 18h, em dias úteis.

Chapter 4

Background e Estado da Arte

4.1 Introdução

Neste capítulo será dada uma breve descrição geral do projeto desenvolvido, bem como das suas partes mais fundamentais. Irão ser também discutidos outros projetos semelhantes e já existentes.

4.2 Transportes Flexíveis

O objetivo principal do projeto ”Transportes Flexíveis” consiste no facilitamento de transportes em zonas rurais, nomeadamente no distrito de Castelo Branco, que não tenham ou não justifiquem a sua presença nas rotas diárias dos transportes públicos regulares, como por exemplo os autocarros. Esta aplicação permite aos clientes agendar viagens desde uma localidade rural até a uma localidade presente nas rotas dos transportes regulares. Para isto, o utilizador, através da plataforma online, agenda uma viagem para um certo dia. De seguida, um funcionário da câmara de Castelo Branco irá confirmar ou recusar a viagem. Caso seja confirmada, este pedido irá, juntamente a outros, ser designado a um taxista, sendo também calculada uma rota eficiente para o transporte destes. Por fim, a plataforma irá também calcular o valor cobrado aos clientes, bem como o valor a ser dado aos taxistas.

4.3 Algoritmo de Rotas

Esta aplicação usufrui de um algoritmo de rotas. Este foi construído com o auxílio da biblioteca da *OR-TOOLS* da Google, mais especificamente com a biblioteca *Vehicle Routing*, e com o *OSRM*. O *OSRM* busca as coordenadas e as distâncias entre pontos (localidades). E a biblioteca de rotas cria uma rota ótima que passe entre todos os pontos. Caso não seja possível passar por todos os pontos de uma só vez (por exemplo, se o carro estiver com lotação máxima) o algoritmo irá remover alguns pontos, e tentar de novo. Quando encontrar uma opção ótima com o maior número de localidades, ele repete o processo para as localidades restantes. Por fim, o horário dos pedidos será atualizado consoante o tempo médio entre as paragens.
A escolha deste algoritmo em contraste a outros deveu-se ao grande suporte por parte da Google, bem como de outros utilizadores, que estão sempre prontos a responder a perguntas e a dar suporte.

4.4 Aplicação

Apresenta□se de seguida a aplicação organizada por camadas e componentes, evidenciando as responsabilidades, dependências e pontos de integração. Esta visão prepara a discussão das decisões tecnológicas sobre o modelo de comunicação entre componentes.
Frontend:
O *frontend* de uma aplicação web é a parte responsável pela interação visual do utilizador à plataforma. Na atualidade, o estado de desenvolvimento em *frontend* está a mudar para um foco na estabilidade ao invés de inovação. Profissionais nesta área focam-se na especificação em tecnologias e *frameworks* já establecidas ao invés da adaptação a tendências emergentes. Esta mudança indica uma mudança no paradigma do desenvolvimento, em que os desenvolvedores focam-se no aumento da produtividade sem a necessidade de adaptação a novas tecnologias.
Em termos de metodologias, tem sido mais prevalente a técnica de componentização, em que projetos são separados por componentes reutilizáveis e mantidos em separado.
Por fim, o SSR está a começar a ser mais prevalente, permitindo que páginas sejam pré-renderizadas antes de chegar ao cliente, promovendo performance.
Backend:
Enquanto que o *frontend* se foca na parte estética e interativa da aplicação, o *backend* é o responsável pela maior parte das operações, tal como verificação e armazenamento de dados. *Frameworks* modernas tal como .NET oferecem funcionalidades robustas que facilitam o desenvolvimento web. Elas não apenas aumentam a produtividade como oferecem uma fácil integração de práticas tais como *RESTful* API.

4.5 Tecnologias Emergentes em Transportes Flexíveis

O modelo de transportes flexíveis começou a ganhar relevância na Europa nos anos 1990, quando municípios e governos locais perceberam que manter linhas de transporte fixas em zonas rurais ou periféricas era financeiramente inviável. Países como a Suécia, Alemanha e Reino Unido foram pioneiros na implementação de soluções de transporte sob demanda, usando inicialmente centrais de atendimento telefónico para organizar as reservas.
Neste momento, existem já em Portugal várias medidas de transportes flexíveis, e existem planos para mais medidas no futuro.
Atualmente, existem plataformas de mobilidade nos seguintes distritos:

- Braga
- Guimarães
- Lisboa
- Coimbra

O conceito de Transportes Públicos Flexíveis (TPF) tem vindo a evoluir. Inicialmente, apenas eram considerados pedidos com antecedência, sendo agora possível responder a pedidos efetuados na hora da viagem.

4.6 Conclusão

Neste capítulo podemos refletir na importância, evolução e implementação dos serviços de Transportes Públicos Flexíveis (TPF). Podemos também verificar a complexidade destes sistemas, envolvendo uma plataforma web, *backend* e uma base de dados, bem como um algoritmo de rotas.

Chapter 5

Atividades Desenvolvidas

5.1 Introdução

Neste capítulo serão descritas as atividades realizadas durante o período do estágio, desde a formação do estagiário até à entrega final do projeto ao cliente.

5.2 Diagrama de Atividades

O estágio teve início no dia 05 de Fevereiro, e termina no dia 12 de Junho. Nas primeiras semanas, o estagiário realizou formações online nas tecnologias de ASP.NET Core e Angular. Após este período inicial, a equipa de desenvolvimento inferiu os requisitos funcionais e não funcionais, o esquema da base de dados e o plano de trabalho. O resto do tempo de estágio foi reservado ao desenvolvimento do projeto, bem como ao suporte ao cliente. 5.1

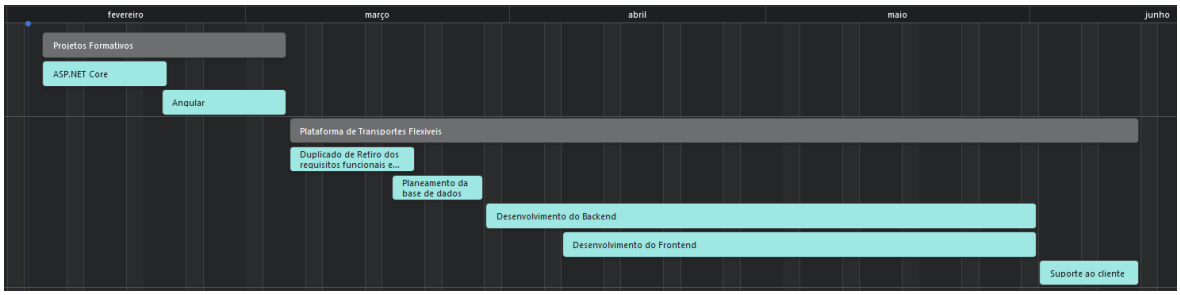

Figure 5.1: Cronograma das atividades desenvolvidas durante o estágio.

5.3 Atividades Realizadas

Nesta secção e seguintes subsecções irão ser descritas as metodologias de trabalho que adotadas, as diversas tarefas diárias e semanais e por fim a integração com o aprendizado académico.

5.3.1 Metodologias Utilizadas

No decorrer do projeto, foram adotadas práticas ágeis para a organização e acompanhamento do progresso das tarefas, nomeadamente o modelo *Scrum*. Existiram reuniões diárias com o intuito de alinhar as atividades, identificar bloqueios e ajustar prioridades. Além disso, o trabalho foi dividido em *sprints* quinzenais, permitindo o planeamento incremental e entregas contínuas. Esta abordagem garantiu uma maior visibilidade do progresso, a promoção da colaboração entre os membros da equipa e assegurou a adaptação constante às necessidades do projeto.

5.3.2 Tarefas Diárias/Semanais

O dia a dia do estagiário iniciou-se com a verificação de novos *e-mails*, seguida da sincronização com o ambiente remoto de trabalho. Em seguida, ocorre a *daily meeting*, onde são partilhadas as atividades realizadas no dia anterior, o planeamento das tarefas do dia atual e o esclarecimento de eventuais dúvidas com os colegas da equipa. A cada duas semanas, foi realizado um levantamento do progresso (*sprint*), identificado o que foi concluído e o que não foi, e procedeu-se ao planeamento da *sprint* seguinte.

5.3.3 Integração do Saber Académico

O estágio revelou-se uma extensão prática do conhecimento adquirido durante a formação académica. Conceitos fundamentais, como o levantamento de requisitos, desenvolvimento de software e gestão de bases de dados, foram aplicados em contexto real, reforçando e expandindo o que foi aprendido na universidade. Além disso, esta experiência permitiu explorar várias ferramentas e metodologias, como o scrum e o controlo de versões com o Git (em específico, o GitLab).
Outro aspeto é a importância das competências transversais, como o trabalho em equipa e a comunicação eficaz, que, embora abordadas de forma mais teórica no ambiente académico, foram desenvolvidas de forma mais significativa durante o estágio, devido à necessidade de interação contínua com outros membros da equipa. Por último, o estágio também forneceu a oportunidade de enfrentar problemas concretos, o que ajuda a ter uma melhor compreensão de como converter os conhecimentos teóricos em soluções práticas.

5.4 Descrição das Atividades

A criação da aplicação envolve uma arquitetura bem definida, composta por um *backend* robusto e um *frontend* dinâmico e interativo.
Backend:
O *backend* foi desenvolvido com foco na criação de uma *Application Programming Interface* (API) segura e eficiente para servir às diversas funcionalidades do sistema. Foram seguidos os princípios *RESTful*, garantindo uma estrutura padronizada para requisições e respostas. As principais tarefas incluiram:

- **Definição da arquitetura do servidor:** Projeção de uma estrutura organizada para o *backend*, com divisão clara entre controladores, serviços e repositórios, garantindo a separação de responsabilidades, e por fim uma versão inicial do esquema da base de dados.
- **Implementação de :** Desenvolvimento de *endpoints* para operações *CRUD*(Create, Read, Update, Delete) relacionadas às principais entidades do sistema.
- **Autenticação e Autorização:** Implementação de um sistema de autenticação *JSON Web Token* (JWT) para garantir que o acesso às funcionalidades protegidas da aplicação são apenas acedidas por utilizadores autorizados.

- **Validação de dados:** Utilização de bibliotecas para validação de dados recebidos do *frontend*, assegurando a integridade das informações armazenadas na base de dados.
- **Integração com a base de dados:** Comunicação eficiente com a base de dados relacional através de um *Object-Relational Mapping* (ORM), otimizando consultas e transações.
- **Gestão de erros:** Implementação de um sistema de tratamento de erros consistente, com respostas claras em caso de falhas.

Frontend:
O *frontend* foi desenvolvido com foco na criação de uma interface amigável, responsiva e intuitiva para os utilizadores finais. Esta abordagem segue princípios modernos de design e usabilidade, proporcionando uma excelente experiência ao utilizador.
As principais tarefas incluiram:

- **Estruturação da aplicação:** Organização do código em componentes reutilizáveis, promovendo uma fácil manutenção e escalabilidade do sistema.
- **Design dinâmico e responsivo:** Desenvolvimento de interfaces adaptáveis a diferentes tamanhos de tela, garantindo acessibilidade tanto em dispositivos móveis quanto em computadores.
- **Design dirigido a diferentes linguagens:** Desenvolvimento de uma interface que considera a internacionalização, permitindo a adaptação da aplicação para português ou inglês, e assim a inclusão de utilizadores não nativos.
- **Implementação de um mapa interativo**: Implementação de um mapa interativo, utilizando o serviço aberto *OpenStreetMaps*, de modo à fácil visualização do mapa da região, e dos pontos de interesse face ao tema da aplicação.
- **Interação com a API:** Integração eficiente com os *endpoints* do *backend* para o envio e o recebimento de dados em tempo real.
- **Gestão de estado:** Implementação de ferramentas para gerir o estado da aplicação, assegurando uma atualização fluida das interfaces de utilizador.
- **Validação no cliente:** Implementação de validações no *frontend* para assegurar a consistência dos dados antes de serem enviados ao *backend*.

5.5 Conclusão

Neste capítulo foram descritas as atividades que o estagiário realizou ao longo do estágio, desde a formação inicial nas tecnologias necessárias até ao desenvolvimento completo da aplicação, tanto no *backend* quanto no *frontend*. Foram também descritas as atividades diárias que o estagiário realizou.

Através do uso de metodologias ágeis, como o Scrum, e da interação constante com a equipa, o estagiário teve a oportunidade de aplicar os conhecimentos adquiridos durante o curso.
As atividades no *frontend* terão como foco o desenvolvimento de uma interface responsiva e amigável, de modo a ser acessível a todos os utilizadores.
As atividades no *backend* envolverão a criação de uma API eficiente e segura, com ênfase na implementação de *endpoints*, autenticação, validação de dados e integração com a base de dados.
O estágio proporcionou a opurtunidade de experienciar o ciclo de vida completo de um projeto de software, desde o planeamento até a entrega final, preparando assim o estagiário para desafios futuros no mercado de trabalho.

Chapter 6

Tecnologias e Ferramentas de Desenvolvimento

6.1 Introdução

Neste capítulo são descritas as principais tecnologias utilizadas ao longo do estágio, tais como as linguagens de programação e ferramentas.

6.2 Linguagens de Programação e Marcação

A lista seguinte apresenta as linguagens de programação e marcação que foram utilizadas no decorrer do desenvolvimento da aplicação.
Estas tecnologias foram escolhidas com base na escolha das *frameworks* 6.5, e também pela sua utilização prevalente no mercado de trabalho.

- HTML: *HyperText Markup Language* ou ”Linguagem de Marcação de Hipertexto” é uma linguagem de marcação utilizada na construção de páginas Web
- CSS: *Cascading Style Sheets* é um mecanismo que permite adicionar estilos às páginas web, tais como cores, fontes de texto, etc.
- JavaScript: Linguagem de programação amplamente utilizada para desenvolvimento web, que permite criar interatividade em páginas e aplicações através do navegador
- TypeScript: superconjunto sintático estrito do JavaScript que adiciona tipagem estática e recursos avançados para melhorar a escalabilidade e a manutenção de projetos.
- C#: Linguagem de programação moderna e orientada a objetos, amplamente utilizada para desenvolvimento de aplicações web com a plataforma .NET.
- SQL: Linguagem padrão para gerenciamento de bases de dados relacionais, permitindo criar, consultar, atualizar e manipular dados.
- LINQ: Ferramenta integrada ao C# que facilita consultas em coleções e bancos de dados, utilizando uma sintaxe declarativa semelhante ao SQL.

6.3 PostgreSQL

Sistema de gestão de base de dados relacional open-source, reconhecido pela sua robustez, extensibilidade e conformidade com padrões SQL.
Em comparação com outros sistemas de base de dados, este apresentou várias vantagens, tal como a compatibilidade com dados geoespaciais, algo que foi útil no contexto deste projeto

e a garantia da integridade dos dados através do suporte às transações com Atomicidade, Consistência, Isolamento e Durabilidade (ACID).

6.4 Ferramentas de Desenvolvimento

A lista seguinte apresenta as ferramentas essenciais que foram utilizadas no desenvolvimento de software, permitindo a otimização de tarefas diárias, a automatização de processos, e a facilitação da comunicação entre os diferentes elementos da equipa, como por exemplo o Microsoft Teams 6.4
Para além disto, ferramentas como o Visual Studio Code 6.4, que incluem desde raiz diferente suporte a diferentes tecnologias, garantiram uma maior qualidade do código desenvolvido.
Visual Studio 2022: Ambiente de desenvolvimento integrado (IDE) da Microsoft, utilizado principalmente para desenvolvimento de aplicativos em C#, .NET, Python, e outras linguagens. Ofereceu recursos avançados como depuração, integração com o Git, ferramenta de resolução de conflitos entre *branches*, etc.
Visual Studio Code: Editor de código-fonte open-source da Microsoft, altamente customizável e com uma vasta gama de extensões. Suportou várias linguagens e foi amplamente utilizado para desenvolvimento de aplicações.
Docker: Plataforma que automatiza a implantação de aplicações em containers, permitindo que o software seja executado de forma consistente em diferentes ambientes, sem a necessidade de alterar as configurações do sistema operativo.
DBeaver: Ferramenta open-source de administração e gerenciamento de bancos de dados. Suporta uma variedade de sistemas, incluindo PostgreSQL e MySQL. Ofereceu uma interface gráfica para facilitar a interação com as bases de dados.
VPN: Rede privada virtual que cria uma conexão segura e criptografada entre o utilizador e a internet, permitindo a navegação de forma anônima. Permitiu também aceder a recursos de rede remota de forma segura, tal como máquinas remotas.
Swagger: Ferramenta que ajudou no design, documentação e teste de APIs *RESTful*. Ofereceu uma interface visual para interagir com os *endpoints* e gerou documentação de forma automatizada.
Git: Sistema de controle de versão distribuído, amplamente utilizado no desenvolvimento de software para gerir alterações no código-fonte e facilitar o trabalho colaborativo.
GitLab: Plataforma de DevOps que fornece repositórios Git, integração contínua e ferramentas de monitorização. Foi usada para o desenvolvimento colaborativo, permitindo a criação e designação de tarefas, bem como a automação de fluxos de trabalho.
Microsoft Teams: Plataforma de colaboração da Microsoft que integra mensagens, chamadas, reuniões e arquivos em um ambiente de trabalho unificado, facilitando a comunicação e colaboração entre equipas.
Figma: Ferramenta de design colaborativo baseada na web, amplamente utilizada para criar interfaces de usuário (UI) e protótipos.
MobaXterm: Aplicativo para Windows que fornece um terminal avançado, com suporte para SSH, RDP, SFTP, entre outros, oferecendo um ambiente unificado para administração

de sistemas remotos e execução de comandos.

6.5 Web Frameworks

Na seguinte lista são mostradas as *frameworks* que serviram como fundação no projeto desenvolvido:
Angular: Framework open-source amplamente utilizada para a construção de aplicações web dinâmicas e escaláveis, desenvolvida pela Google. Programada em TypeScript, oferece uma arquitetura robusta e modular, com uma abordagem orientada a componentes para o desenvolvimento de interfaces de usuário complexas. Incluiu suporte nativo para a criação de *Single Page Applications* (SPAs) e uma poderosa integração com *APIs RESTful*. Por fim, a capacidade de reutilização, e a fácil manutenção de componentes, tornou-a ideal para grandes projetos empresariais.
ASP.NET Core: Framework open-source da Microsoft para a construção de aplicações web modernas e APIs robustas. Baseada em C#, é conhecida pela sua alta performance, segurança e flexibilidade. O ASP.NET Core ofereceu também suporte multiplataforma, permitindo a execução em Windows, Linux e macOS. Além disso, possuiu uma integração nativa com serviços do Azure (também da *microsoft*). Por fim, a inclusão de injeção de dependências, de *middlewares* configuráveis e de formas de autenticação moderna, tornou esta framework uma escolha confiável para aplicações empresariais escaláveis e seguras.

6.6 Bibliotecas e API's Open Source

A seguir são mostradas as diferentes bibliotecas de código aberto que foram úteis durante o desenvolvimento da aplicação.
Estas ferramentas permitiram uma fácil integração de certas tecnologias robustas e complexas no projeto, facilitando assim o desenvolvimento visto não ter existido a necessidade de as desenvolver de raiz.
Google OR-TOOLS: Biblioteca open-source da Google que oferece uma série de ferramentas para otimização combinatória, como algoritmos para resolver problemas de roteamento e fluxo de rede.
OpenStreetMaps API: API open-source que fornece dados geográficos detalhados e atualizados do mundo inteiro. Permitiu acesso a mapas, informações de locais e funcionalidades de roteamento.
OSRM: Open Source Routing Machine (OSRM) é uma ferramenta que oferece serviços rápidos de roteamento e cálculo de rotas em dados do OpenStreetMap, utilizada em aplicações de navegação e planeamento de trajetos.

6.7 Conclusão

Neste capítulo foi possível ver as diversas tecnologias e linguagens que foram utilizadas durante o estágio.

A formação e utilização nas *frameworks* Angular, ASP.NET Core, juntamente com as suas respetivas linguagens de programação, permitiram o desenvolvimento de uma aplicação completa, eficiente e escalável. As bibliotecas como o OR-TOOLS permitiram aos desenvolvedores pouparem tempo na construção de métodos já existentes.
As ferramentas como o Visual Studio, Docker, DBeaver, etc. permitiram uma fácil visualização de dados e um fácil desenvolvimento/resolução de erros, o que permitiu um fluxo de trabalho melhor.
Por fim, ferramentas como Microsoft Teams e Figma permitiram uma fácil comunicação entre os diversos elementos de equipa, garantindo uma fácil colaboração.

Chapter 7

Engenharia de Software

7.1 Introdução

O desenvolvimento da aplicação de transporte seguiu princípios fundamentais de engenharia de software, garantindo a criação de um sistema robusto, seguro e escalável. Este capítulo apresenta as metodologias, arquiteturas e práticas adotadas durante o processo de desenvolvimento, desde a análise de requisitos até à implementação final.

A aplicação foi concebida com uma arquitetura orientada a serviços, separando claramente as responsabilidades entre o *frontend* (interface do utilizador) e o *backend* (lógica de negócio e gestão de dados). Esta separação permite maior flexibilidade, manutenibilidade e escalabilidade do sistema.

7.2 Análise de Requisitos

A fase de análise de requisitos constituiu a base fundamental para o desenvolvimento da aplicação. Foi realizada uma análise detalhada das necessidades dos diferentes tipos de utilizadores, resultando na identificação de requisitos funcionais e não funcionais essenciais para o sucesso do projeto.

7.2.1 Requisitos Funcionais

Os requisitos funcionais definem as funcionalidades específicas que o sistema deve fornecer aos seus utilizadores. Foram identificados os seguintes requisitos funcionais principais:7.2.1

Código	Descrição
RF01	O sistema deve permitir o registo de novos utilizadores através de um formulário com validação de dados.
RF02	O sistema deve permitir a autenticação de utilizadores através de email e palavra-passe.
RF03	O sistema deve permitir a edição de dados pessoais do utilizador (nome, morada, NIF, contactos).
RF04	O sistema deve permitir a alteração de credenciais de acesso (email e palavra-passe).
RF05	O sistema deve permitir a criação de pedidos de viagem com seleção de origem e destino.
RF06	O sistema deve calcular e apresentar o custo da viagem e hora de partida.
RF07	O sistema deve pré-preencher dados do utilizador em reservas subsequentes.
RF08	O sistema deve enviar confirmação por email após a submissão do pedido.
RF09	O sistema deve permitir a visualização do histórico de viagens do utilizador.
RF10	O sistema deve permitir o cancelamento e edição de pedidos nos estados “Criado” e “Em Reagendamento”.
RF11	O sistema deve notificar o utilizador sobre mudanças de estado dos seus pedidos.
RF12	O sistema deve permitir a visualização e gestão de notificações.
RF13	O sistema deve permitir marcar notificações como lidas.
RF14	O sistema deve apresentar um mapa interativo com paragens, zonas e agregados.
RF15	O sistema deve permitir filtrar informações no mapa conforme preferências do utilizador.
RF16	O sistema deve apresentar informações detalhadas das paragens ao selecionar marcadores.
RF17	O sistema deve permitir o reporte de problemas através de formulário dedicado.
RF18	O sistema deve disponibilizar página de contactos com informações de funcionamento.
RF19	O sistema deve permitir comunicação direta com a equipa de suporte.
RF20	O sistema deve permitir a confirmação de pedidos por funcionários autorizados.
RF21	O sistema deve permitir a gestão de motoristas registados na plataforma.
RF22	O sistema deve permitir a criação e edição de contas de utilizadores.
RF23	O sistema deve permitir a atribuição e revogação de permissões específicas.

Table 7.1: Requisitos Funcionais do Sistema

7.2.2 Requisitos Não Funcionais

Os requisitos não funcionais definem as características de qualidade que o sistema deve possuir, estabelecendo critérios de desempenho, segurança, usabilidade e outros atributos essenciais.7.2.2

Código	Descrição
RNF01	O sistema deve implementar autenticação segura para todos os utilizadores.
RNF02	O sistema deve utilizar controlo de acesso baseado em roles e permissões.
RNF03	O sistema deve redirecionar utilizadores não autorizados para páginas apropriadas.
RNF04	O sistema deve proteger dados sensíveis através de encriptação adequada.
RNF05	O sistema deve validar todos os inputs do utilizador para prevenir ataques de injeção.
RNF06	A interface deve ser intuitiva e de fácil navegação para utilizadores não técnicos.
RNF07	O sistema deve ser responsivo e funcionar adequadamente em dispositivos móveis.
RNF08	O sistema deve fornecer feedback claro sobre o estado das operações.
RNF09	As páginas devem carregar em menos de 3 segundos em condições normais de rede.
RNF10	O sistema deve seguir padrões de acessibilidade web (WCAG 2.1).
RNF11	O sistema deve ter uma disponibilidade mínima de 99.5%.
RNF12	O sistema deve recuperar automaticamente de falhas menores.
RNF13	O sistema deve manter a integridade dos dados em todas as operações.
RNF14	O sistema deve realizar backups automáticos dos dados críticos.
RNF15	O sistema deve suportar pelo menos 100 utilizadores simultâneos.
RNF16	As consultas à base de dados devem ser otimizadas para resposta rápida.
RNF17	O sistema deve implementar cache para melhorar a performance.
RNF18	O mapa interativo deve carregar e renderizar em menos de 2 segundos.
RNF19	O código deve seguir padrões de desenvolvimento bem definidos.
RNF20	O sistema deve ser modular para facilitar futuras expansões.
RNF21	A documentação técnica deve ser completa e atualizada.
RNF22	O sistema deve incluir logs detalhados para facilitar debugging.

Table 7.2: Requisitos Não Funcionais do Sistema

7.3 Arquitetura do Sistema

Esta secção descreve o padrão arquitetural adotado e a sua materialização na solução, detalhando as camadas, os componentes principais e as interações. O objetivo é justificar as opções tomadas e evidenciar como a arquitetura suporta os requisitos funcionais e não funcionais.

7.3.1 Padrão Arquitetural

A aplicação adota uma arquitetura em camadas (layered architecture) com separação clara entre *frontend* e *backend*. Este padrão permite:

- Separação de responsabilidades entre apresentação e lógica de negócio
- Maior flexibilidade para alterações futuras
- Facilidade de manutenção e testing
- Possibilidade de desenvolvimento paralelo por equipas distintas

7.3.2 Componentes Principais

7.3.2.1 Frontend

O frontend é responsável pela interface do utilizador e inclui:

- Interface de reserva de viagens
- Mapa interativo com visualização geográfica
- Sistema de autenticação e gestão de perfil
- Páginas de histórico e notificações
- Formulários de contacto e suporte

7.3.2.2 Backend

O *backend* implementa a lógica de negócio e inclui:

- API REST para comunicação com o *frontend*
- Sistema de autenticação e autorização
- Gestão de dados de utilizadores e viagens
- Algoritmo de planeamento de rotas
- Interface administrativa (backoffice)

7.4 Gestão de Segurança

7.4.1 Sistema de Autenticação

O sistema implementa autenticação baseada em credenciais (email e palavra-passe) com as seguintes características:

- Validação de formato de email
- Requisitos mínimos de complexidade para palavras-passe
- Sessões seguras com timeout automático
- Proteção contra ataques de força bruta

7.4.2 Controlo de Acesso

Foi implementado um sistema robusto de controlo de acesso baseado em:

7.4.2.1 Roles (Funções)

- **Cliente** - Acesso às funcionalidades básicas de reserva e consulta
- **Funcionário** - Acesso a funcionalidades administrativas específicas
- **Administrador** - Acesso completo a todas as funcionalidades

7.4.2.2 Permissões Granulares

- Cada página do backoffice tem uma role associada
- Cada ação tem uma permissão específica
- Verificação automática de permissões em tempo real
- Redirecionamento seguro para utilizadores não autorizados

7.5 Design de Interface

7.5.1 Princípios de Usabilidade

O design da interface seguiu princípios fundamentais de usabilidade:

- **Simplicidade** - Interface limpa e intuitiva
- **Consistência** - Padrões visuais uniformes em toda a aplicação
- **Feedback** - Confirmações claras de ações do utilizador
- **Prevenção de Erros** - Validação de dados e mensagens orientativas
- **Acessibilidade** - Compatibilidade com dispositivos móveis

7.5.2 Experiência do Utilizador

7.5.2.1 Fluxo de Reserva

O processo de reserva foi otimizado para máxima eficiência:

1. Seleção intuitiva de origem e destino
2. Pré-preenchimento automático de dados
3. Visualização clara de custos e horários
4. Confirmação por email automática

7.5.2.2 Gestão de Estado

O sistema mantém o utilizador informado através de:

- Estados claros dos pedidos (Criado, Aguarda Confirmação, Pendente)
- Notificações automáticas de mudanças
- Histórico completo e organizado

7.6 Metodologia de Desenvolvimento

Esta secção descreve o processo adotado desde o planeamento até ao *deployment*, destacando as fases, práticas de qualidade e sua relação com a entrega contínua; nas subseções seguintes detalham-se as etapas e os mecanismos de garantia de qualidade.

7.6.1 Processo de Desenvolvimento

O desenvolvimento seguiu uma abordagem iterativa e incremental, com foco na entrega contínua de valor. As principais fases incluíram:

1. **Análise de Requisitos** - Identificação e documentação de necessidades
2. **Design da Arquitetura** - Definição da estrutura do sistema
3. **Desenvolvimento Incremental** - Implementação por funcionalidades
4. **Testes Contínuos** - Validação regular das funcionalidades
5. **Deployment e Monitorização** - Implementação e acompanhamento

7.6.2 Práticas de Qualidade

Para garantir a qualidade do software, foram adotadas as seguintes práticas:

- **Code Review** - Revisão de código por pares
- **Testes Automatizados** - Suíte de testes unitários e de integração
- **Documentação Contínua** - Manutenção de documentação atualizada
- **Controlo de Versões** - Gestão adequada do código fonte
- **Monitorização** - Acompanhamento de métricas de performance

7.7 Conclusão

A aplicação de transporte desenvolvida representa uma implementação sólida dos princípios de engenharia de software. A arquitetura escolhida, baseada na separação entre *frontend* e *backend*, proporciona flexibilidade e manutenibilidade, enquanto o sistema robusto de segurança garante a proteção adequada dos dados dos utilizadores.
A análise detalhada de requisitos funcionais e não funcionais permitiu criar uma solução que atende às necessidades específicas dos utilizadores, mantendo elevados padrões de qualidade, segurança e usabilidade. O sistema de gestão baseado em roles e permissões oferece controlo granular sobre o acesso às funcionalidades, essencial para um ambiente multi-utilizador.
As práticas de desenvolvimento adotadas, incluindo testes contínuos, revisão de código e documentação adequada, contribuem para a sustentabilidade a longo prazo da solução. A

interface intuitiva e responsiva garante uma experiência positiva do utilizador em diferentes dispositivos e contextos de uso.

Este projeto demonstra como a aplicação rigorosa de metodologias de engenharia de software pode resultar numa solução robusta, escalável e adequada às necessidades reais dos utilizadores finais.

Chapter 8

Implementação da Plataforma Web

8.1 Introdução

Neste capítulo irá ser descrito, de forma detalhada, a implementação técnica da plataforma desenvolvida. Como o estágio envolveu todo o processo da aplicação (ou seja, uma abordagem *fullstack*, serão discutidas as camadas do *frontend*, *backend*, a integração/comunicação entre ambos, os sistemas externos de apoio e por fim as ferramentas e tecnologias utilizadas

8.2 Detalhes de Desenvolvimento do Frontend

Nesta secção irá ser abordada em detalhe a implementação do *frontend*, detalhando as tecnologias e metodologias utilizadas, e o porquê.
O *frontend* da aplicação corresponde à camada de interação com o utilizador, responsável pela apresentação da informação e pela recolha de dados. Seguidamente, são detalhadas as opções tecnológicas, a arquitetura e as funcionalidades implementadas.

8.2.1 Estrutura e Arquitetura Utilizada

Para a realização do *frontend* foi utilizada a *framework* Angular. A escolha desta tecnologia foi motivada pelos anos de experiência prévia de alguns elementos da equipa, levando a uma vantagem significativa tanto no desenvolvimento em si como na partilha de conhecimentos com os restantes membros. Um outro motivo passou pelo facto de ser uma tecnologia muito utilizada e com muita documentação e tutoriais, o que levou a uma rápida aprendizagem por parte dos elementos da equipa que não contam com muita experiência na área.
Em termos de metodologia, foi utilizada uma abordagem modular baseada em componentes, promovendo a reutilização de componentes, escalabilidade e facilidade de manutenção.
Por fim, a aplicação foi dividida em três áreas principais:

- Autenticação: área que inclui os componentes responsáveis pela autenticação de utilizadores (*login*), registo e recuperação da palavra-passe. Esta área usufrui de um esquema com responsividade, garantindo uma boa experiência em dispositivos de vários tamanhos.

- Parte pública: área acessível sem autenticação, contendo todos os componentes acessíveis aos utilizadores, tal como o pedido de viagens, o mapa com as paragens disponíveis, uma página de contactos, entre outros. Esta área é a área principal da aplicação, sendo a área que aparece quando se realiza uma pesquisa pela plataforma nos motores de

Figure 8.1: Componente reutilizável de *card*

busca. Tal como a área de autenticação, esta também usufrui de um sistema responsivo.

- Backoffice: área reservada a utilizadores autenticados com permissões específicas, inclui todas as funcionalidades necessárias para a manutenção da plataforma, tal como a criação/edição de paragens, edição de permissões dos utilizadores, visualização de dados estatísticos, entre outros. A segurança desta secção deve-se ao uso de guardas de rota (*AuthGuards)*, que verificam as permissões do utilizador autenticado e garantem que um utilizador sem permissões não consiga aceder a esta página. Ao contrário das áres de autenticação e da parte pública, esta não usufrui de responsividade, não funcionando corretamente em dispositios mais pequenos, tal como *smartphones*. Isto deve-se ao facto de esta área ser apenas usada em contexto de trabalho (em computadores), por funcionários da câmara.

Todas estas áreas possuem também, nos seus formulários, verificação de dados, garantindo que não ocorrem pedidos ao *backend* de dados incorretos.
Por fim, existem também componentes reutilizáveis. Estes componentes são utilizados globalmente em páginas em que sejam necessários, como por exemplo *cards* 8.1. Esta abordagem ajuda a ter um código mais limpo, visto não ser necessária a programação do mesmo componente em várias páginas, e permite uma fácil alteração global do comportamento ou estilo deste.

8.2.2 Tecnologias e Bibliotecas Utilizadas

Tal como mencionado anteriormente, o desenvolvimento foi realizado com base na *framework* Angular, com *TypeScript*. Esta é conhecida pela sua robustez, suporte da comunidade e facilidade de integração com *APIs REST*. Para além do núcleo do Angular, foram utilizadas várias bibliotecas e ferramentas auxiliares:

- Angular Router: utilizado na gestão da navegação entre as diferentes páginas da aplicação. Através da configuração de rotas no formato de árvore e da utilização de parâmetros dinâmicos, foi possível construir uma aplicação *Single Page Application* (SPA), garantindo uma navegação fluída sem recarregamento total da página.

- *AuthGuards*: é uma função do Angular que permite validar o acesso a um módulo ou componente. Por exemplo, é possível usar um Guard para impedir que um utilizador não autenticado aceda a uma determinada página.

  Existem vários tipos de Guards:

  - *CanActivate*: verifica se o utilizador pode entrar numa rota.
  - *CanLoad*: impede que um módulo seja carregado se certas condições não forem cumpridas.

  Os *Guards* funcionam como filtros e ajudam a tornar a aplicação mais segura, controlada e eficiente ao evitar o carregamento desnecessário de JavaScript.

- *Software Internationalization* (i18n): um módulo de JavaScript que permite a tradução dinâmica de texto em vários idiomas. Para isto, é necessário criar ficheiros *JavaScript Object Notation* (JSON) para cada uma das línguas disponíveis. Neste projeto foram apenas consideradas as línguas de Português de Portugal e Inglês do Reino Unido.

- PrimeNG: biblioteca gráfica de componentes visuais e funcionais baseada em Angular. A existência desta biblioteca permitiu a aceleração do desenvolvimento, através da utilização de componentes pré-feitos responsivos.

- Leaflet: biblioteca open-source, construída em JavaScript, de mapas interativos. Esta ferramenta foi utilizada, juntamente com o OpenStreetMaps, para a implementação do mapa na aplicação, permitindo a visualização das zonas e das paragens no distrito, e a alteração dinâmica das coordenadas das paragens (funcionalidade disponível no *backend*).

- TailwindCSS: framework utilitário de *Cascading Style Sheets* (CSS) que permite aplicar classes diretamente no HTML, evitando assim a necessidade de (re)escrever folhas de estilo separadas. A sua abordagem de *utility-fist* serviu como base na criação de *layouts* responsivos.

Estas tecnologias, em conjunto, proporcionaram uma base sólida para o desenvolvimento de uma aplicação web completa, interativa e adaptada às necessidades dos diferentes perfis de utilizador.

8.2.3 Autenticação e Registo

Nesta secção vai ser exploradas em mais detalhe as diferentes funcionalidades de cada uma das áreas.

8.2.3.1 Funcionalidades da área de Autenticação

Login

Conta com um formulário com validação de um *email* válido, e de palavra-passe, bloqueando o utilizador de se tentar autenticar caso uma destas regras falhe. Após a correta inserção dos dados é enviado um pedido ao *backend*. A resposta do *backend* pode ser de falha (no caso de o *email* palavra-passe estarem incorretos, ou não existirem) ou de sucesso, onde é também enviado um *token* que irá permitir ao utilizador realizar pedidos com autenticação sem a necessidade de se re-autenticar. Este token é armazenado no navegador do cliente, e tem validade de 2 horas.

Registo
Conta também com um formulário com validação de um *email* válido, de uma palavra-passe que cumpra os seguintes requisitos:

- Tamanho mínimo de oito caracteres
- Existência de no mínimo uma letra maiúscula
- Existência de no mínimo uma letra minúscula
- Existência de no mínimo um número
- Existência de no mínimo um carácter especial

, e por fim do preenchimento do campo de confirmação de palavra-passe com a mesma palavra-passe usada no campo acima. Caso uma destas verificações falhe, o utilizador não consegue submeter o seu pedido.
Após o utilizador submeter o pedido, os dados são enviados ao *backend*, podendo este dar uma resposta de falha, ou de sucesso. No caso de sucesso, o utilizador é redirecionado para uma página de confirmação de conta, onde deve submeter um código enviado para o seu *email*. No caso de o código ser correto, este é redirecionado para a página inicial da parte pública.
Recuperação de palavra-passe
Conta também com um formulário com validação de um email válido. Após o pedido ser feito ao *backend*, se o *email* existir na base de dados, é devolvida uma mensagem de sucesso, e enviado um email com um link ao utilizador. O utilizador deve então seguir esse link, e submeter a sua nova palavra-passe, com validações semelhantes ao processo de criação de conta. Após a inserção correta, este é redirecionado novamente para a página de *login*.
Estas validações são feitas graças a sistemas já embutidos no Angular, nomeadamente o *Reactive Forms Module*. Este módulo permite criar e gerir formulários de forma programática, com total controlo sobre os estados e a validação dos campos. Duas funcionalidades do Angular foram também essenciais para a validação dos dados:

- *FormControl* e *FormGroup*: estruturas que representam os campos e o conjunto do formulário, respetivamente. Estas entidades permitem associar regras de validação diretamente nos campos, tal como o *Validator.required* para os campos que são obrigatórios, e o *Validators.email* para garantir que o texto introduzido tem formato de email.

8.2.3.2 Funcionalidades da Área da Parte Pública

Utilizador autenticado vs. não autenticado
A área pública muda ligeiramente caso o utilizador esteja autenticado. No modo convidado (sem autenticação), este é obrigado a introduzir os seus dados básicos (*email*, nome e contacto) quando realiza um pedido de viagem.
Em contraste, no caso de o utilizador estar autenticado, os dados pessoais são automaticamente preenchidos na realização do pedido.

8.2.4 Mapa Dinâmico

Nesta secção irá ser explorada a implementação do mapa dinâmico na aplicação.
Devido ao tema da aplicação, é fundamental a existência de um mapa. Este é responsável por demonstrar intuitivamente as disponíveis paragens e as suas respetivas zonas, através de pontos de coordenadas associados a cada um destes.
Ao cliente, o mapa permite verificar quais as paragens existentes e onde se encontram, e obter visualização da estimativa do percurso ao realizar um pedidos entre dois pontos.
Quanto aos funcionários, o mapa permite a alteração das posições das paragens/zonas de forma intuitiva, através de um icon que pode ser movido ao longo do mapa. Este depois converte a informação de modo a poder ser guardada e carregada da base de dados.
Esta implementação é possível graças ao Leaflet, que desempenha o papel central na renderização e interação do mapa no frontend. Esta biblioteca, leve e flexível, permite adicionar camadas, marcadores, rotas e eventos personalizados, garantindo uma experiência fluida e adaptável a diferentes dispositivos. Sendo esta biblioteca apenas responsável pela parte visual, é também usado o OpenStreetMaps para a base cartográfica que alimenta o Leaflet. Esta ferramenta de código aberto fornece os dados geográficos que permitem desenhar o mapa.
A combinação de ambas as tecnologias permite a implementação de uma solução robusta, livre de dependências comerciais, e apta a lidar com as diferentes necessidades de detalhe geográfico.

8.2.5 Notificações

Nesta seção irá ser explicado o sistema de notificações do utilizador.

O sistema de notificações no frontend é responsável por informar o utilizador sobre eventos relevantes a si, tal como atualizações dos seus pedidos (agendado, recusado, etc.). Esta funcionalidade garante uma comunicação eficiente e contextual entre o sistema e o utilizador, melhorando a usabilidade e a experiência geral da aplicação.

A interface deste sistema é implementada recorrendo à biblioteca PrimeNG, que oferece componentes predefinidos, dinâmicos e personalizáveis.

Cada notificação possui um estado, que pode ser lida ou não lida. Este estado é armazenado e gerido de forma a permitir ao utilizador distinguir entre mensagens novas e antigas, assegurando também a persistência dessa informação. Quando o utilizador visita o site é feito um pedido assíncrono ao *backend*, que verifica se existem notificações por ler e. Caso existam, é colocado um pequeno ícon com o número de notificações por ler.

8.2.6 Áreas Reservadas

O *backoffice* destina-se apenas a utilizadores com certas permissões. Estas permissões são agrupadas em *roles*. Nesta área apenas podem aceder utilizadores com *roles* de funcionário,

motorista ou administrador, sendo depois certas funcionalidades apenas disponíveis consoante as permissões. Por exemplo, o administrador é o único que pode alterar a *role* dos restantes utilizadores. Isto é possível a permissões e *roles* pré-definidos na base de dados, e aos *AuthGuards*. Por fim, é importante mencionar que estas informações são armazenadas no *token* fornecido no momento de autenticação, sendo por isso necessário que, no caso de um utilizador ter permissões alteradas, este ter de terminar a sua sessão e voltar a autenticar, de modo a obter um novo token.

8.2.7 Gestão de Erros

A aplicação inclui também um sistema de tratamento de erros. Quando ocorre um erro no *backoffice* este retorna uma resposta com um código de erro (por exemplo, 500) e uma mensagem em forma de código (por exemplo, user.login.failed). No *frontend* estes códigos são usados para saber o que aconteceu e dar o correto *feedback* ao utilizador.

8.2.8 Integração do Backend no Frontend

O *backend* foi desenvolvido para assegurar a lógica de negócio, a gestão de dados e a comunicação com o *frontend*. Nas subsecções seguintes são descritos a arquitetura, a comunicação com a base de dados e o mapeamento de modelos.

8.2.8.1 Comunicação

A comunicação entre o *frontend* e o *backend* é realizada através de chamadas *Hypertext Transfer Protocol* (HTTP), usando o módulo *HttpClient* fornecido pelo Angular. Os dados são trocados com o formato JSON, o que facilita a serialização e desserialização de objetos. Visto que alguns *endpoints* são usados por diversos componentes, foram criados vários serviços para as várias áreas existentes.
A comunicação foi ainda realizada de forma reativa, utilizando listeners implementados através do sistema de observables do Angular (via RxJS). Ou seja, sempre que era efetuado um pedido HTTP ao backend, a aplicação subscrevia-se ao resultado da operação, aguardando a respetiva resposta dentro de um tempo máximo. Este modelo permitiu executar ações específicas com base na resposta recebida, tais como apresentar mensagens de sucesso ou erro, redirecionar o utilizador, ou atualizar o estado da interface de forma automática.

8.2.8.2 Autenticação

Tal como mencionado anteriormente, a aplicação utilizou o mecanismo de JWT. Após o *login* o *backend* devolve um token com os dados e permissões do utilizador, que é depois armazenado no navegador. Este token é incluído automaticamente nos cabeçalhos de todos os pedidos feitos, através de um *middleware* que interceta todos os pedidos e o insere.

8.3 Detalhes de Desenvolvimento do backend

Nesta secção irá ser abordada em detalhe a implementação do *backend*, detalhando as tecnologias e metodologias utilizadas, e o porquê.
Irá também ser falado em detalhe da implementação do algoritmo de rotas.
O *backend* foi desenvolvido para assegurar a lógica de negócio, a gestão de dados e a comunicação com o frontend. Nas subsecções seguintes será descrita a arquitetura, a comunicação com a base de dados e o mapeamento de modelos.

8.3.1 Estrutura e Arquitetura Utilizada

O *backoffice* foi implementado usando o padrão Model View Controller (MVC) adaptado para uma arquitetura em camadas. Este padrão promove uma melhor separação de responsabilidades do sistema e facilita a manutenção e posterior evolução do sistema, e é dividido em três distintas camadas: modelo, visão e controlo.
A camada do modelo é responsável pela representação dos dados da aplicação. Inclui as entidades, os objetos de transferência de dados (DTO) e os serviços que implementam a lógica.

A camada de controlo, pela sua vez, gere a comunicação entre o utilizador (*frontend*) e o sistema, sendo que recebe e processa os pedidos, e posteriormente encaminha os resultados adequados para a camada de visão.
Por fim, a camada de visão é responsável pela apresentação dos dados ao utilizador. No caso deste projeto, esta camada não foi utilizada.

8.3.2 Tecnologias e Bibliotecas Utilizadas

Neste capítulo irá ser descrito e explicadas as diferentes tecnologias e bibliotecas utilizadas na construção do *backend*.

- **Controllers** - Esta é a camada responsável por expor os *endpoints* da API, sendo este o ponto de entrada de todos os pedidos HTTP realizados. Cada *controller* está associado a um determinado conjunto de funcionalidades, e define as rotas. Por exemplo, o *controller* de autenticação é responsável por tudo o que tem a ver com autenticação, tal como o registo, o recuperar conta, etc. Cada *controller* define uma rota, e cad método uma sub-rota (por exemplo, auth/login, sendo auth definida pelo *controller* e*login* pelo método).
  É também nos *controllers* (mais especificamente nos métodos) que é feito a verificação de dados e de permissões

- **Services** - Esta é a camada responsável pela lógica por detrás da aplicação. Funcionam como intermediários entre os *controllers* e os *repositories*. Por exemplo, no contexto de autenticação, é no *repository* que se encontra a lógica necessária para gerar o *token*.

- **Repositories** - Esta é a camada responsável pela interação com a base de dados, abstraindo as operações de leitura e escrita. Resumidamente, fornecem uma interface

para acesso a dados sem ter a preocupação de como a base de dados está implementada. Este nível de abstração garante também uma maior segurança.

- **Entity Framework Core** - Consiste numa versão leve, extensivel e multi-plataforma da *Entity Framework*, que por sua vez é uma *framework* ORM da Microsoft para .NET.
- **JWT** - Padrão aberto para transmitir informações de forma segura entre duas partes como um objecto JSON. Estes *Tokens* são assinados digitalmente e enviados pelo *frontend* em cada pedido, e validado pelo *backoffice* nos *controllers*.
- **Swagger** - Serviço usado para gerar automaticamente a documentação da API, bem como um ambiente interativo para testes dos *endpoints*. Usado apenas em contexto de desenvolvimento.
- **AutoMapper** - Biblioteca que facilita a conversão entre objetos de diferentes camadas, nomeadamente entre entidades e modelos de dados.

8.3.3 Algoritmo de Rotas

Um dos requisitos deste projeto passa pela implementação de um sistema que calcule aos condutores uma rota ótima para a realização do maior número de pedidos possível. Esta rota é calculada com base nos pontos de origem e destino, bem como nas lotações máximas dos veículos utilizados pelos motoristas.
Esta tarefa coube na sua totalidade ao estagiário, desde a procura de técnicas e de bibliotecas até à sua implementação e testes.
Nas próximas subsecções é apresentada a introdução ao problema, o funcionamento geral do algoritmo e os detalhes técnicos da sua implementação.

8.3.3.1 Funcionamento geral

Nesta secção descreve-se o fluxo operacional do algoritmo conforme executado diariamente. Esta visão de alto nível enquadra os critérios de seleção e ordenação antes dos detalhes de implementação.
O algoritmo corre diariamente às 15:00. Primeiramente, filtra os pedidos com base nos seguintes critérios:

- Pedidos no estado ”pendente”, isto é, os pedidos que foram confirmados pelos funcionários no *backoffice*;
- Pedidos com número de passageiros menor ou igual à capacidade dos veículos disponíveis;
- Pedidos cuja data de partida/chegada seja no dia a seguir ao cálculo da rota;

De seguida, é obtida uma lista de todos os condutores disponíveis nesse dia e nessa zona, e são considerados por ordem decrescente ás lotações dos respetivos carros.
A seguir, é calculada todas as possíveis rotas de origem - destino que o condutor consegue fazer de uma só vez (por exemplo, se dois pedidos com 2 passageiros tiverem origem no ponto

A, com destinos dos pontos B e C, a rota irá ser A -> B -> C, assumindo que o veículo tem no mínimo 5 lugares), e mantendo os horários de destino (consegue chegar ao destino dentro do limite de 10 minutos da hora estabelecida pelo cliente).
Após cada rota ótima calculada, o algoritmo designa um condutor para essa rota e verifica se existem pedidos por realizar. Caso existam, o algoritmo repete o processo, porém com algumas condições:

- Vários motoristas - no caso de existirem mais motoristas, o algoritmo é modificado para considerar a nova lotação máxima, e os horários da rota anterior não são consideradas (isto é, ambas as rotas podem ser realizadas paralelamente);
- Um motorista - no caso mais comum de apenas estar disponível um motorista, o algoritmo repete o processo para a nova rota, mas tendo em conta a rota anterior (ou seja, a rota apenas começa na hora em que a rota anterior acaba);

Por fim, as rotas são armazenadas na base de dados, e os pedidos são atualizados entre duas opções:

- Os pedidos que consigam ser realizados pelas rotas definidas, passam ao estado de ”agendado”.
- Os pedidos que não consigam ser realizados pelas rotas definidas (por exemplo, o condutor não consegue realizar todos os pedidos dentro do tempo útil), passam ao estado de ”em reagendamento”.

8.3.3.2 Parte técnica

Para a implementação deste algoritmo foram utilizadas principalmente duas ferramentas: OR-Tools e OSRM (Open Source Routing Machine).
OSRM: O OSMR permitiu a obtenção das informações necessárias nos cálculos das rotas, nomeadamente as distâncias e tempos médios entre pontos geográficos. Estes dados são utilizados juntamente com o OR-Tools, que usa esta matriz para calcular as rotas ótimas.
OR-Tools: O OR-Tools, tal como mencionado anteriormente, é uma biblioteca *open-source* da Google com o objetivo de otimizar problemas de rotas. Para este caso foi usado o algoritmo Capacitated Vehicle Routing Problem (CVRP), e depois adaptado para o caso específico.
Matriz de tempos Através da ferramenta *Open Source Routing Machine* (OSRM) 8.3.4.1 foi possível obter uma matriz com os tempos entre os diversos pontos geográficos. Devido à pré-existencia na base de dados das coordenadas dos pontos a serem considerados pelo algoritmo, não foi necessário a pesquisa pelos pontos, sendo a ferramenta usada exclusivamente na obtenção da matriz.
CVRP O CVRP consiste no Travelling Salesman Problem (TSP) mas para encomendas, ou seja, com restrições de retiradas e entregas, e com limite de capacidade dos veículos.
A biblioteca proporcionada pela Google dá como exemplo uma empresa que pretende entregar encomendas aos clientes, tendo cada veículo a capacidade total de 15 unidades a cada momento. A maior diferença deste exemplo para o objetivo dos transportes flexíveis está

no facto de se assumir que cada veículo tem de regressar ao ponto inicial (ou seja, apenas apanha encomendas no armazém), sendo no exemplo a seguir dado pelo ponto 'o'. 8.2

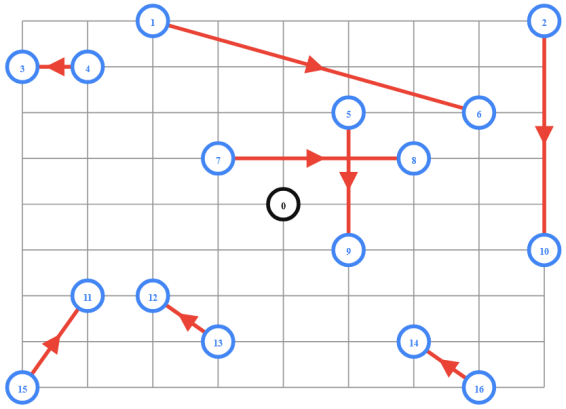

Figure 8.2: Exemplo de entregas/retiradas

Após a realização de todos os cálculos, a biblioteca devolve a seguinte rota otimizada:

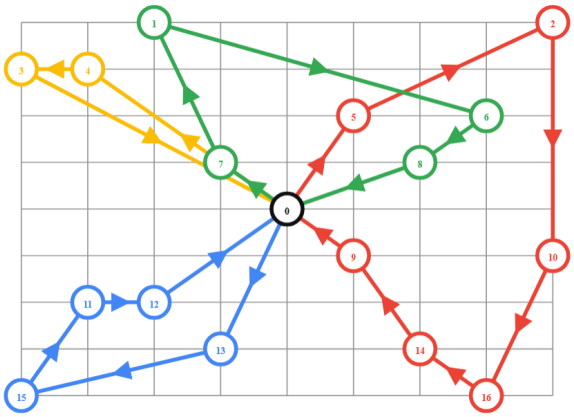

Figure 8.3: Exemplo de uma rota otimizada

Adaptação para os transportes flexíveis De modo a ultrapassar a diferença entre os transportes flexíveis e as ferramentas disponibilizadas pelo OR-Tools 8.3.3.2, foi criado um ponto imaginário (mas real), tendo como coordenadas o centro do mapa do distrito de Castelo Branco, e posteriormente utilizado como ponto de partida. Deste modo, o algoritmo calcula as rotas otimizadas com origem e destino neste ponto imaginário. Após a obtenção de uma rota, são ignorados estes dois pontos e apenas considerados os pontos intermédios, conseguindo assim rotas entre várias origens e destinos sem a necessidade de um motorista ter de começar ou parar num certo ponto.

8.3.4 Integração com Sistemas Externos

Para complementar as funcionalidades do sistema, foi necessária a integração com serviços externos que apoiam a recolha e o processamento de dados. Para além do *frontend* e do *backend*, a aplicação conta também com a integração de serviços externos essenciais ao seu funcionamento. Estes serviços foram disponibilizados através de contentores Docker, que

correm no servidor paralelamente.
No final do projeto, foram também criados contentores para o *frontend* e para o *backend*. Isto facilitou a entrega do projeto final ao cliente.
As próximas subsecções explicam em detalhe cada um destes sistemas.

8.3.4.1 OSRM

O OSRM consiste numa ferramenta *open-source* altamente eficiente para a obtenção de informação geográfica. É usado em conjunto com dados do OpenStreetMaps. No contexto da aplicação, este serviço foi usado no cálculos das rotas.

Este serviço recebe um pedido com pontos geográficos e com parâmetros e devolve a informação pedida, em relação aos parâmetros fornecidos. No contexto da aplicação, apenas foi usada a funcionalidade de obtenção de matriz de tempos entre pontos, pelo que, neste caso específico, o serviço devolve uma matriz em forma de texto.

O OSRM têm uma API pública, porém esta exerce limites de total de pedidos. Este fator, juntamente com o risco de o serviço ficar em baixo, levou à decisão de alojar localmente esta ferramenta. Felizmente o OSRM fornece uma imagem docker pré-preparada para este fim, facilitando a implementação.
Por fim, por motivos de otimização, a imagem foi alterada de modo a que apenas usasse os mapas necessários, sendo neste caso o de Portugal continental (o mapa mais pequeno disponibilizado pelo OpenStreetMaps).

8.3.4.2 Base de dados

De modo a facilitar a configuração das ligações à base de dados, foi criado desde o inicio uma imagem com uma base de dados PostgreSQL. Para a criação foi usada uma imagem disponibilizada pelo docker, e depois alterada de modo a permitir a persistência dos dados localmente. Isto permitiu re-construir as imagens as vezes necessárias sem haver perda de dados da base de dados.

8.3.4.3 *Frontend* e *Backend*

Por fim, no final do projeto, houve a necessidade de instalar a aplicação no servidor do cliente. Como a equipa apenas tinha acesso ao terminal remoto desta máquina, e de modo a garantir a consistência do sistema, criou-se duas imagens docker, uma para o *frontend* e outra para o *backend*.

8.3.5 Camada de Acesso aos Dados e Mapeamento dos Modelos

Camada de Acesso aos Dados e Mapeamento dos Modelos refere-se à arquitetura de software que isola como os dados são lidos e gravados. Esta camada abstrai o armazenamento persistente, fornecendo as operações básicas, tal como criar, ler, atualizar e eliminar, para além de

gerir as conexões e transações. Isto permite que a lógica seja feita a partir de objetos em vez de comandos SQL diretos.

8.3.5.1 Comunicação com a base de dados

A persistência de dados é feita através do *Entity Framework Core*, que consiste num ORM para .NET que permite mapear classes do domínio para tabelas da base de dados. Esta ferramenta facilita a criação, leitura, atualização e eliminação de dados através do uso do *DbContext*, uma classe central que representa a sessão com a base de dados e permite interagir com os conjuntos de entidades definidas.
Para além disso, o *Entity Framework Core* foi também usado na criação de migrações, que permitem gerir as alterações no esquema da base de dados de forma controlada e incremental, sem ter de modificar manualmente a base de dados.

8.3.5.2 Mapeamento de modelos

De modo a garantir uma separação entre dados de uso internos da aplicação e dados destinados aos utilizadores foi adotado o use de *Data Transfer Objects* (DTO). DTO são objetos simples que permitem a transferência de dados entre as diferentes camadas da aplicação. O mapeamento entre as entidades e os DTO foi feito de forma automática, através do *AutoMapper*, uma ferramenta disponibilizada pelo ASP.NET Core. Um exemplo é os dados do utilizador, em que a entidade *User* contém diversos campos, incluindo a *Hash* da sua palavra-passe. Com o uso de DTO, apenas os dados necessários são expostos, neste caso sendo, por exemplo, o nome, id, e email.

8.4 Conclusão

Neste capítulo podemos ver que o backend foi estruturado de forma a garantir uma arquitetura limpa, modular e de fácil manutenção. A divisão em camadas, seguindo o formato MVC, permitiu uma clara distinção entre os endpoints e a comunicação com a base de dados. O uso de DTO garantiu o encapsulamento adequado dos dados expostos, promovendo assim uma maior segurança.
Podemos também compreender em detalhe como funciona e como foi implementado o algoritmo de cálculo de rotas, uma das partes mais fundamentais deste projeto. Este algoritmo integra dados geográficos das zonas e paragens, processa restrições e otimizações, e apresenta o resultado tanto no backend como no frontend, permitindo ao utilizador visualizar a rota prevista diretamente sobre o mapa interativo. No frontend, o uso do Leaflet combinado com OpenStreetMap proporciona uma representação visual precisa, incluindo coordenadas das paragens e zonas, e atualizações dinâmicas conforme a previsão de novas rotas.
Também podemos ver a forma como o frontend foi concebido para oferecer uma interface intuitiva e responsiva. As funcionalidades incluem sistemas de notificações desenvolvidos com PrimeNG, gestão de estados de leitura, histórico de eventos e apresentação de elementos de forma clara e adaptada ao perfil do utilizador no backoffice. O uso de componentes

modulares e reutilizáveis garante a coerência visual e facilita a manutenção futura.
Por último, é descrita a implementação de serviços externos e a sua integração no sistema através de Docker, permitindo isolar cada serviço em containers independentes, simplificando a configuração, distribuição e escalabilidade. Estes serviços suportam funcionalidades como cálculo avançado de percursos, geocodificação e sincronização de dados com sistemas de terceiros, assegurando que a solução final é robusta e flexível para evolução futura.

Chapter 9

Demonstração e Validação da Plataforma Web

9.1 Introdução

Neste capítulo irá ser demonstrada a estrutura da autenticação do site, tal como o sistema de entrada (*login*), de registo e de recuperar a palavra-passe.
Vão ser também explicados os motivos das abordagens tomadas, as possíveis alternativas consideradas, e os desafios enfrentados.

9.2 Módulos Públicos - Frontoffice

Nesta secção irá ser demonstrada a estrutura da parte pública do website, isto é, a parte acessível a todos os utilizadores.

9.2.1 Funcionalidades do *Frontoffice*

Esta parte da aplicação é focada principalmente na utilização por parte dos utilizadores, tendo sido prestada especial atenção à facilidade de uso e à responsividade para os diferentes dispositivos (por exemplo, tablets).

As funcionalidades incluem funções fundamentais para o utilizador normal, tal como a efetuação de pedidos, a visualização das paragens no mapa, o registo, recuperação de conta, etc. Basicamente corresponde a todo o tipo de acessos que o cliente possa precisar.

9.2.2 Autenticação e Registo

O módulo de autenticação garante a segurança do acesso ao sistema, permitindo o registo, *login* e recuperação de palavra-passe. Nas subsecções seguintes são descritos os mecanismos que suportam estas funcionalidades.

9.2.2.1 Diagrama da autenticação

Seguidamente apresenta-se um diagrama, representando a autenticação. 9.1

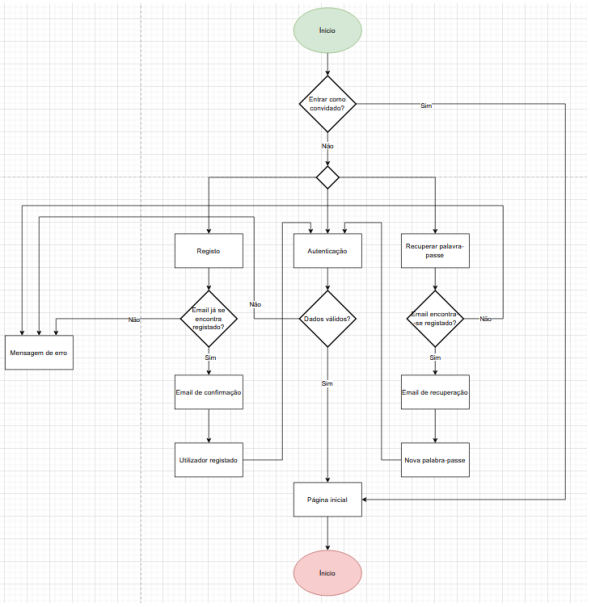

Figure 9.1: Diagrama de Autenticação

9.2.2.2 Login

O sistema de login baseia-se na validação de credenciais fornecidas pelo utilizador. A figura abaixo ilustra a interface da página de login. 9.2

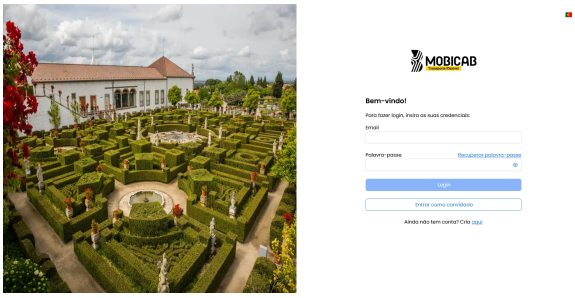

Figure 9.2: Página de Login

O sistema de *login* funciona a partir das credenciais de email e password. Quando o utilizador submete ambos, estes são mandados para o *backend*, onde a password é convertida em uma *hash* para poder ser comparada com a *hash* presente na base de dados, prevenindo assim acesso direto às credenciais secretas do utilizador. Caso as *hashes* correspondam, é enviado um *token* de sessão para o utilizador, e este recebe uma notificação de sucesso 9.3 e é reencaminhado para a página inicial da aplicação. 9.3

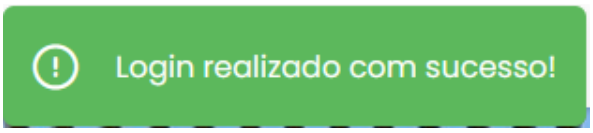

Figure 9.3: Mensagem de sucesso no login

Pelo contrário, se a password for errada, ou se o email não existir na base de dados, o *backend* envia uma mensagem de erro genérica ao *frontend*. A razão de o erro ser genérico ao invés

de especificar o que correu mal deve-se a uma tentativa de segurança, visto que utilizadores com más intenções poderiam utilizar este método para descobrir emails que estivessem registados. Por fim, o *frontend* mostra uma mensagem de erro 9.4, e o utilizador pode voltar a tentar iniciar a sessão.9.4

Figure 9.4: Mensagem de erro no login

9.2.2.3 Registo

A criação de novas contas é realizada através da página de registo. Na Figura seguinte encontra-se o formulário correspondente.9.5

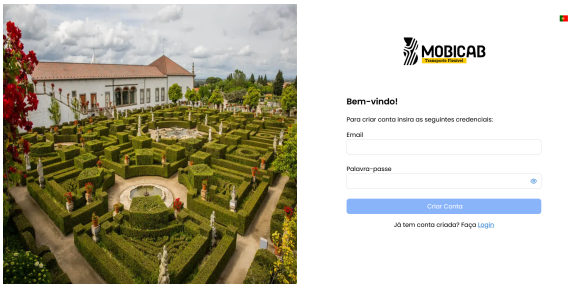

Figure 9.5: Página de Registo

O sistema de registo permite a criação de uma conta por parte do utilizador. Para tal, é necessário o preenchimento dos campos de email e da palavra-passe. Quando o formulário é submetido, os dados são enviados para o *backend*, onde ocorre uma validação. Se a conta já existir, é mostrada uma mensagem de erro 9.6.

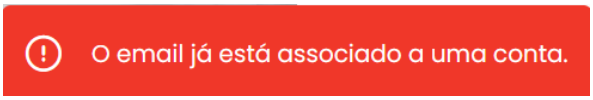

Figure 9.6: Mensagem de erro no registo

Por outro lado, se a palavra-passe não cumprir os requisitos de segurança, o botão de ”Criar Conta” fica desabilitado, e é mostrada uma mensagem a indicar os requisitos mínimos desta. 9.7

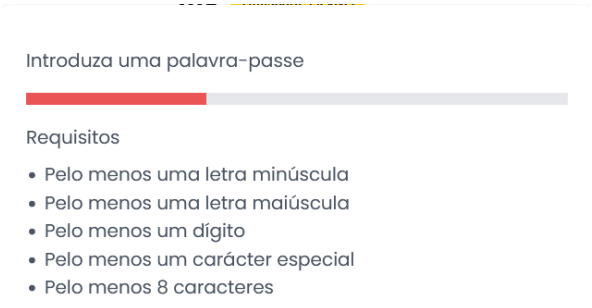

Figure 9.7: Requisitos mínimos da palavra-passe

Se a validação for bem-sucedida, a password é convertida para hash, e só então armazenada na base de dados, garantindo assim a confidencialidade dos dados sensíveis, e por fim é enviado um código para o email registado 9.8, e o utilizador é redirecionado para a página de validação de conta. 9.9

Figure 9.8: Email com o código de validação da conta

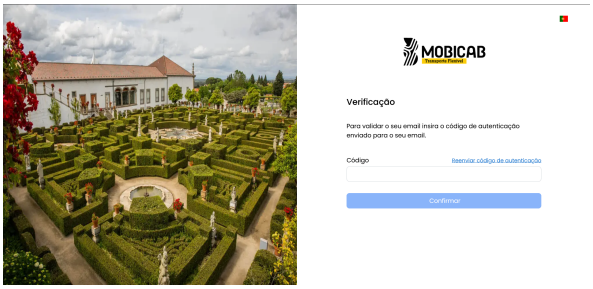

Figure 9.9: Página de validação do código do registo

Se o código introduzido for incorreto, é mostrada uma toast de erro genérica 9.10.

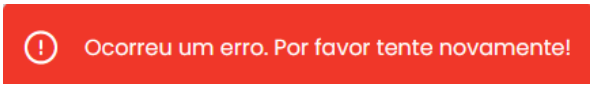

Figure 9.10: Toast de erro genérico

Por outro lado, se o código for válido, é mostrada uma mensagem de sucesso 9.11 e o utilizador é redirecionado para a página de *login* 9.2.

Figure 9.11: Toast de sucesso

Por fim, caso o código passe o tempo de expiração, fica inválido, e o utilizador necessita de requisitar outro, clicando no link ”Reenviar código de autenticação”.

9.2.2.4 Recuperar Palavra-Passe

O processo de recuperação de palavra-passe é realizado em várias etapas, conforme ilustrado nas figuras seguintes.

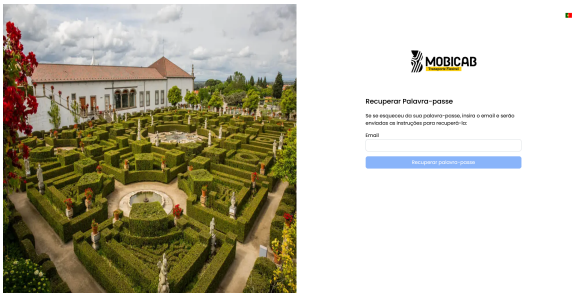

Figure 9.12: Página de Recuperar Palavra-Passe

Caso o utilizador se esqueça da palavra-passe que utilizou para criar a conta, é possível a este criar uma nova, através da página de recuperar palavra-passe 9.12.

Aqui, após o utilizador inserir o email que utilizou no registo, irá receber um email com instruções para repor a palavra-passe 9.13

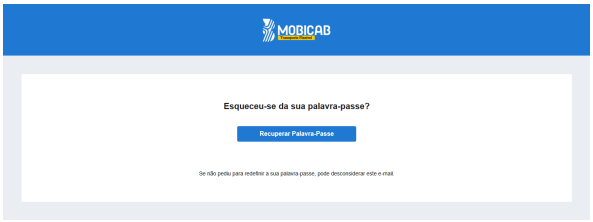

Figure 9.13: Email de reset da palavra-passe

Quando o utilizador clica no link irá ser redirecionado para a página de reposição da palavra passe 9.14. No link desta página é incluído o email do utilizador, bem como um token temporário gerado no momento em que foi feito o requisito ao *backend*.

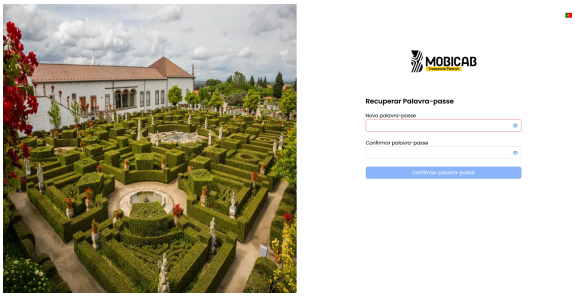

Figure 9.14: Página de reposição da palavra-passe

Por fim, após o utilizador escolher a nova palavra-passe, este é redirecionado para a página de *login*.

9.2.3 Gestão de Viagens

O pedido de viagem é realizado através de um formulário interativo. De seguida apresentam-se as principais interfaces utilizadas para recolher os dados do utilizador.

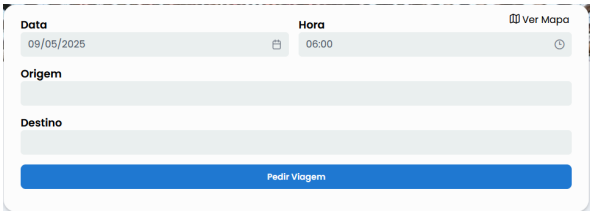

Figure 9.15: Menu de pedido de viagem

No centro da página encontra-se o menu de pedido de viagem. Através deste menu, o utilizador pode selecionar a origem e o destino. O destino é filtrado de acordo com a origem selecionada, visto que nem todas as paragens partilham a mesma rota/horários.

Após o utilizador selecionar ”Pedir Viagem”, este é levado para a página de Novo Pedido. 9.16

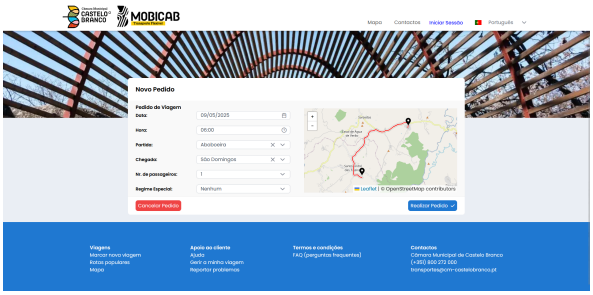

Figure 9.16: Janela de preenchimento de dados da viagem

Nesta página é possível escolher a data e hora da viagem, o número de passageiros, o regime especial (se for aplicado), e ainda alterar a origem e destino. Por fim o utilizador pode visualizar no mapa o trajeto. A data vai ser a escolhida, porém a hora serve apenas para referência de chegada. Isto é, o algoritmo após calcular a rota vai garantir que a pessoa chega ao destino antes da hora escolhida, mas não garante que seja próxima da hora.

Após o utilizador submeter as informações, a aplicação passa para a última etapa. 9.17

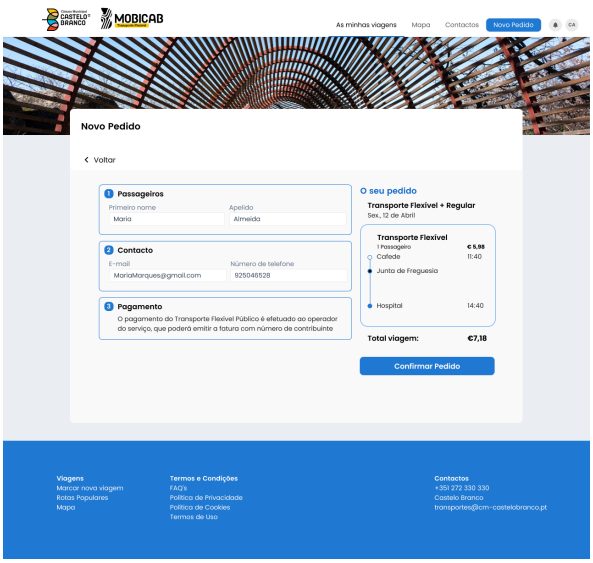

Figure 9.17: Janela de informações do cliente

Aqui o utilizador deve inserir o primeiro e último nome, o email e o número de telemóvel. É importante referir que, no caso de o utilizador já se encontrar registado na aplicação, e se tiver preenchido estes dados anteriormente, estes vêm já pré-preenchidos. Por fim, o utilizador pode visualizar os custos da viagem e a hora de partida.9.18

Figure 9.18: Mensagem de confirmação

Após a última submissão, é mostrada uma mensagem de sucesso e enviado um email ao utilizador.9.19

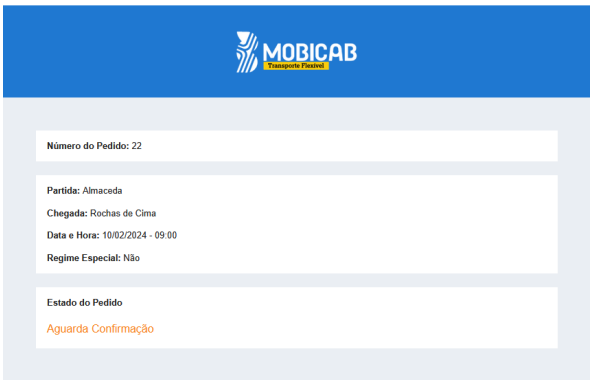

Figure 9.19: Email recebido pelo cliente

Neste momento, o pedido encontra-se no estado de ”Aguarda Confirmação”. Neste estado, é necessário que um funcionário no backoffice confirme o pedido. Só a partir daí é que fica no estado pendente, e é considerado pelo algoritmo no planeamento da rota.

9.2.4 Mapa Interativo de Transporte

A aplicação inclui um mapa interativo que disponibiliza informações sobre paragens, zonas e agregados. A figura seguinte apresenta uma vista geral.9.20

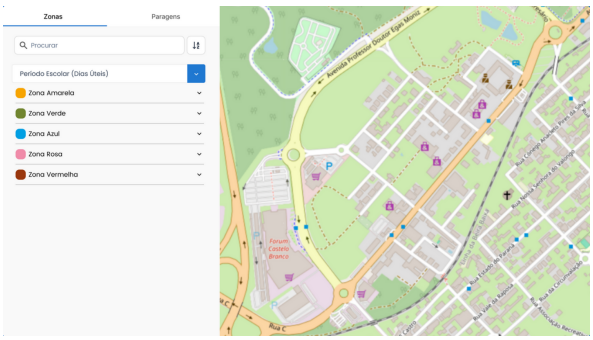

Figure 9.20: Mapa com todas as paragens, agregados e zonas

Na barra de navegação é possível encontrar a aba ”Mapa”. Ao clicar no mesmo, o utilizador é redirecionado para a página do mapa.
Nesta página, é possível ver todas as paragens operacionais, bem como as zonas e os agregados. É possível, também, filtrar estas informações conforme a preferência do utilizador.
Ao selecionar o marcador de uma paragem no mapa, é possível obter o nome desta para uma melhor consulta.

9.2.5 Gestão de Viagens do Utente

Na área pessoal do utilizador, existe uma secção dedicada ao histórico de viagens. A figura seguinte ilustra esta página.9.21

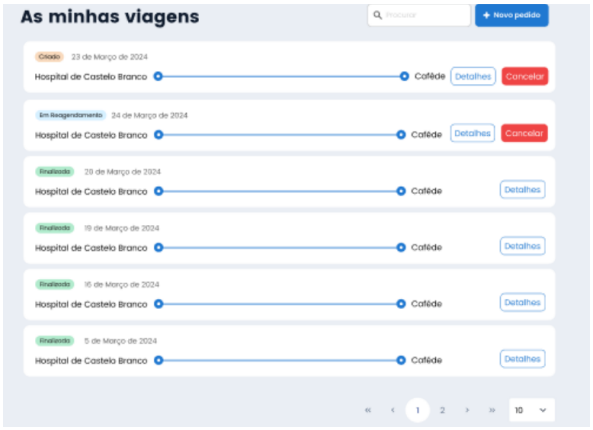

Figure 9.21: Página de histórico de pedidos de viagem do cliente

Através da aba Viagens (no caso de o utilizador estar autenticado) é possível aceder à página de histórico dos pedidos. Nesta página é possível ver todos os pedidos efetuados pelo utilizador. Os pedidos, por defeito, são apresentados por ordem de estado, sendo que os primeiros são os que estão pendentes (estados Criado e Em Reagendamento). Os restantes são apresentados por data descendente, ou seja, do mais recente para o mais antigo. É ainda possível cancelar ou editar um pedido que esteja com o estado Criado ou Em Reagendamento.

9.3 Sistema de Notificações

O sistema inclui uma secção dedicada às notificações, permitindo ao utilizador acompanhar alterações relevantes no estado dos seus pedidos e outras informações importantes. A figura seguinte apresenta a interface onde estas notificações são listadas e geridas.9.22

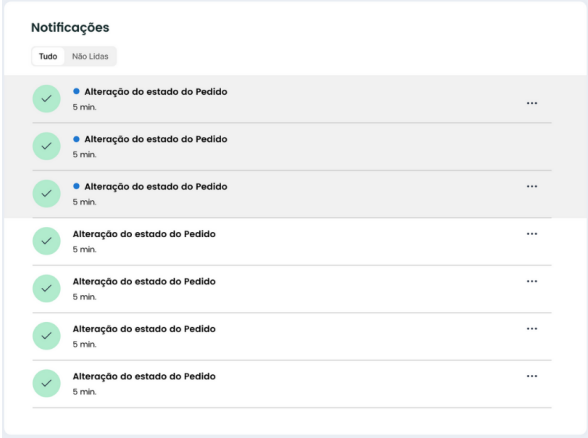

Figure 9.22: Página de notificações do cliente

Na barra de navegação (caso o utilizador tenha iniciado sessão na plataforma) é possível encontrar um botão com um ícone de um sino. Este botão redireciona o utilizador para a página de Notificações. Nesta página é possível visualizar todas as atualizações referentes aos pedidos do utilizador, e à sua conta.

O utilizador irá receber notificações sempre que o seu pedido mudar de estado ou quando este for cancelado, e sempre que este realizar alterações às informações da sua conta. Para marcar uma notificação como lida, basta aceder às opções da mesma através do símbolo "..." e selecionar a opção "Marcar como lida". No caso de estar a usar a aplicação num dispositivo móvel, basta carregar na própria notificação.

9.4 Área Pessoal do Utente

A área pessoal permite ao utilizador gerir os seus dados e configurações. As figuras seguintes apresentam as principais interfaces desta funcionalidade.

Figure 9.23: Pop-up de opções do utilizador

Na barra de navegação (caso o utilizador tenha iniciado sessão na plataforma) é possível encontrar um botão com as suas iniciais. Ao clicar neste botão, é possível visualizar a opção de Definições. Ao escolher essa opção, será redirecionado para a página Área Pessoal.9.24

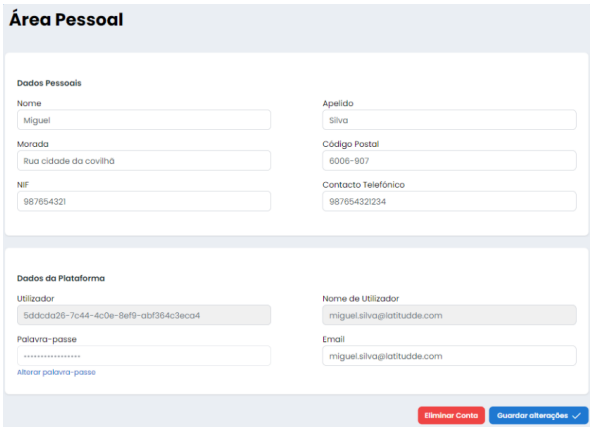

Figure 9.24: Página pessoal do utilizador

Nesta página é possível alterar as informações da sua conta, tal como o nome, morada, Número de Identificação Fiscal (NIF), entre outras. É também possível alterar as credenciais associadas à conta, tal como a palavra-passe e o email.

9.4.1 Gestão de Problemas

O sistema disponibiliza uma funcionalidade para reportar problemas diretamente à equipa de suporte. As figuras seguintes ilustram o acesso e o formulário de reporte.

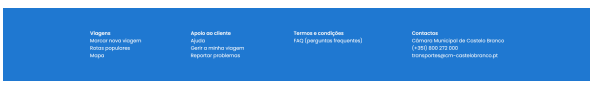

Figure 9.25: Rodapé da aplicação

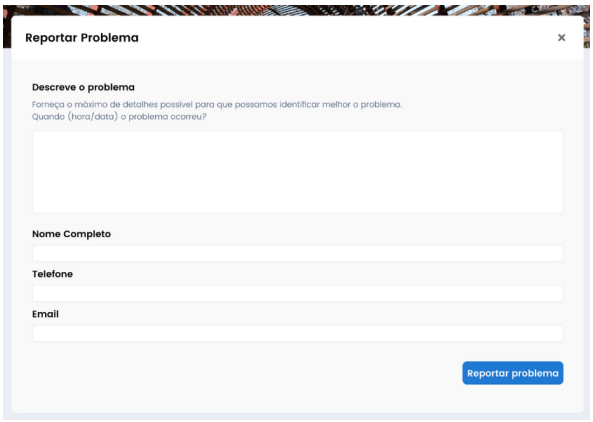

Figure 9.26: Página de reportar problemas

No rodapé da plataforma, é possível encontrar uma ligação para a página de Reportar Problema. Ao clicar nesta, o utilizador será redirecionado para uma página onde poderá descrever detalhadamente qualquer problema que poderá encontrar. Nesta página, encontra-se um formulário no qual é possível fornecer uma descrição clara e completa do problema, permitindo à equipa de suporte entender melhor a situação. Após submeter o relatório, a equipa de suporte entrará em contacto com o utilizador assim que possível.

9.4.2 Gestão de Contactos

Para facilitar a comunicação entre utilizadores e a equipa de suporte, a aplicação inclui uma página de contactos, representada na figura seguinte.9.27

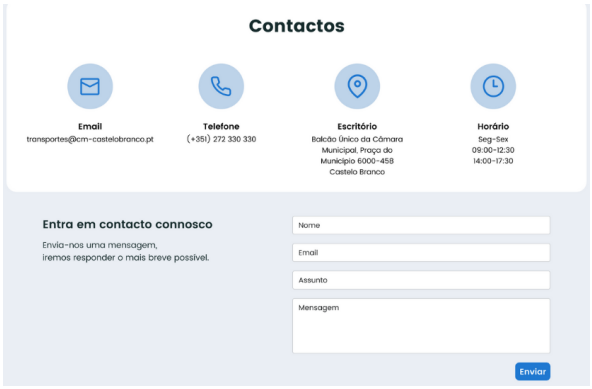

Figure 9.27: Página de contactos

No rodapé da plataforma, é possível encontrar uma ligação para a página de Contactos. Ao clicar nesta, o utilizador será redirecionado para uma página onde poderá encontrar os contactos e o horário de funcionamento. Nesta página, existe também um formulário simples, igual ao formulário da página de Reportar Problemas.

9.5 Módulos Privados - *Backoffice*

O *backend* do backoffice fornece as ferramentas necessárias para a gestão e manutenção da plataforma. As subsecções seguintes detalham as funcionalidades e páginas disponíveis para os utilizadores autorizados.
Nesta secção irá ser dada uma breve descrição do *backend*, e mostradas algumas páginas. Devido a certas partes serem sensíveis no que toca á segurança do sistema, apenas porções do backoffice serão exploradas e explicadas.

9.5.1 Funcionalidades do Backoffice

O backoffice apenas permite o acesso a funcionários específicos. Isto é assegurado a partir de permissões e de funções (*roles*) dos utilizadores. Por defeito a aplicação vem com um utilizador administrador (isto é, um utilizador com todas as permissões). Este pode criar ou alterar contas com determinadas permissões. Cada página do backoffice tem uma *role* associada, e cada ação possível dentro dessa página uma permissão. Se o utilizador não tiver uma certa permissão o link da página/ação correspondente não vai aparecer no ecrã deste e, caso ele tente entrar diretamente através do *url*, é redirecionado para uma página padrão a informar que não tem permissões suficientes.

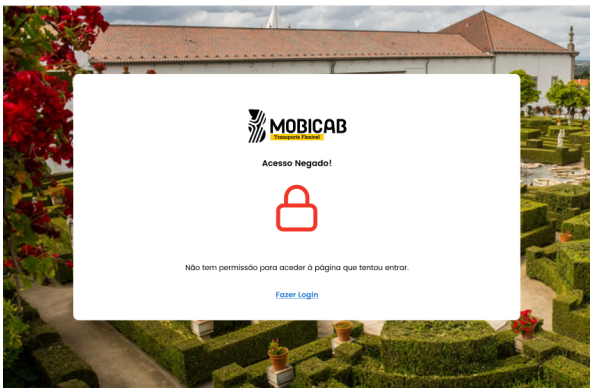

Figure 9.28: Página de acesso proibido

9.5.2 Gestão dos Utilizadores e Perfis

Uma das páginas disponíveis no backoffice refere-se aos utilizadores registados na plataforma.

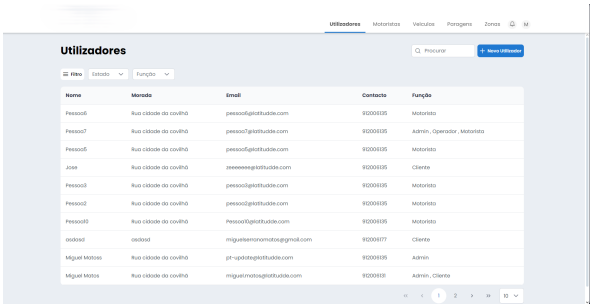

Figure 9.29: Página dos utilizadores

Neste espaço, os utilizadores têm a oportunidade de criar novos perfis ou modificar aqueles que já estão registados no sistema. Além disso, é aqui que se pode atribuir ou revogar permissões específicas a cada utilizador, garantindo assim um controlo adequado sobre as suas funcionalidades e acessos.

9.5.3 Gestão dos Motoristas

Uma das páginas disponíveis no backoffice refere-se aos motoristas que estão registados na plataforma, oferecendo informações detalhadas sobre suas atividades e histórico de viagens.

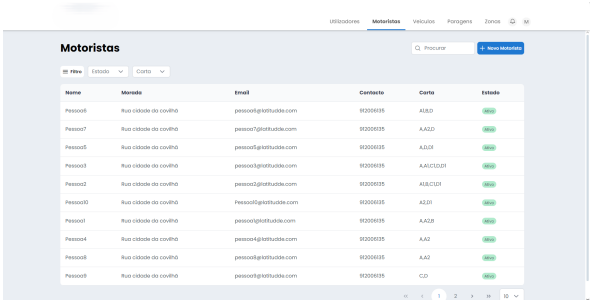

Figure 9.30: Página dos motoristas

Aqui é possível criar um novo motoristas ou verificar/editar motoristas já registados.

9.6 Conclusão

Neste capítulo podemos ver a demonstração e validação dos principais módulos da plataforma, cobrindo a autenticação, a área pública e o *Backoffice*, incluindo os respetivos fluxos essenciais de utilização. Verificou-se também o correto funcionamento das operações críticas, com mensagens claras de sucesso e erro e respeito pelas permissões definidas nos acessos. Concluindo, os resultados apresentados confirmam o cumprimento dos requisitos anteriormente declarados.

Chapter 10

Conclusão e Trabalho Futuro

10.1 Conclusão

Este relatório apresenta a experiência de estágio curricular realizada na empresa Latitudde – Digital Enablers, LDA, com uma duração de nove meses. O principal objetivo foi o desenvolvimento de um sistema de transportes flexíveis para o município de Castelo Branco, visando facilitar a mobilidade dos residentes em áreas rurais.
Durante este período foram abordadas todas as etapas do ciclo de vida de desenvolvimento de software: o levantamento de requisitos, a esquematização e implementação da base de dados, o planeamento e organização do trabalho nas *dailies/sprints*, desenvolvimento do *frontend* e *backend* e por fim os testes e suporte à aplicação. O sistema resultante possibilitou à autarquia gerir os pedidos de transporte de maneira mais eficiente e acessível, contribuindo para a inclusão digital e social da população.
Apesar do sucesso do projeto, existiram diversas dificuldades, nomeadamente a limitação inicial de conhecimentos nas tecnologias utilizadas e a falta de clareza em certos requisitos. Devido a estes problemas, algumas implementações deveriam ter sido diferentes, tal como a modularização no *frontend* ou o uso de testes automatizados. Porém, o reconhecimento destas falhas ajudará o estagiário em futuros projetos.
O estágio representou uma etapa fundamental na minha formação enquanto engenheiro informático. A experiência permitiu-me aplicar os conhecimentos adquiridos no percurso académico a um projeto real, com impacto direto na sociedade.
Um especial agradecimento à Latitudde, à Câmara Municipal de Castelo Branco, e aos orientadores pela orientação, apoio e confiança depositada no estagiário. Esta experiência foi decisiva para o crescimento profissional e consolidação do percurso na área de desenvolvimento de software.

10.2 Contributo Pessoal

Ao longo deste percurso, o estagiário conseguiu adquirir diversas competências técnicas, com destaque para o uso do Angular no *frontend* e ASP.NET Core no *backend*. Também a utilização de PostgreSQL para modelação de uma base de dados relacional e a utilização de Docker para ambiente de produção permitiram adquirir uma visão prática e robusta da engenharia de software moderna.
Além disso, foram enfrentados desafios que exigiram o desenvolvimento de soft skills, como a comunicação eficaz com colegas e orientadores, a gestão do tempo em ambiente ágil e a colaboração em equipa. A constante adaptação aos requisitos e a aprendizagem autónoma de tecnologias reforçaram a maturidade profissional do estagiário.

10.3 Trabalho Futuro

Apesar do esforço e dedicação na realização deste projeto, existem diversos pontos que poderiam ter sido abordados e implementados de uma maneira melhor.
Na parte do *frontend* repara-se que o site demora alguns segundos a ser carregado, pelo que se deveria ter usado diferentes técnicas de otimização, tal como a redução do tamanho das imagens, o pré-carregamento das mesmas e a adoção de sistemas de cache. Existem também muitas páginas idênticas que poderiam ter sido feitas usando o principio de componentes reutilizáveis, facilitando o desenvolvimento e uma futura evolução ou mudança na estética e funcionamento da aplicação.
Deveria ter sido também investido algum tempo na aprendizagem e construção de sistemas de testes automáticos, de modo a garantir a correta funcionalidade de todas as partes da aplicação ao longo do desenvolvimento.
No contexto do algoritmo, pode ser melhorado de modo a considerar variáveis, tais como o trânsito, estradas possivelmente cortadas, etc. É também possível que o cálculo das rotas não esteja bem otimizado, pelo que se deve experimentar variantes do TSP de modo a encontrar o melhor algoritmo.
No âmbito da conteinerização da aplicação seria bom implementar sistemas avançados de suporte, tal como um serviço de reinicio do sistema em caso de falha, limpeza regular da cache, sistema de registo de falhas, entre outros.
Por fim, apesar do esforço pela parte da equipa na implementação de um sistema seguro, não foram feitos testes de âmbito de segurança informática, pelo que é possível que existam falhas que possam comprometer o sistema.

Bibliography

[1] Freeman, A., *Pro Angular 13*, Apress, 2022.

[2] Allen, J., *Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages*, Apress, 2020.

[3] Microsoft Docs, *ASP.NET Core Documentation*, Disponível em: `https://learn.microsoft.com/aspnet/core/`, Acedido em junho de 2025.

[4] Google Developers, *OR-Tools: Operations Research Tools*, Disponível em: `https://developers.google.com/optimization/`, Acedido em junho de 2025.

[5] PostgreSQL Global Development Group, *PostgreSQL Documentation*, Disponível em: `https://www.postgresql.org/docs/`, Acedido em junho de 2025.

[6] Swagger, *Open Source Tools for APIs*, Disponível em: `https://swagger.io/tools/`, Acedido em junho de 2025.

[7] OpenStreetMap Foundation, *OpenStreetMap Wiki*, Disponível em: `https://wiki.openstreetmap.org/`, Acedido em junho de 2025.

[8] GitLab Inc., *GitLab Documentation*, Disponível em: `https://docs.gitlab.com/`, Acedido em junho de 2025.

[9] Sommerville, I., *Software Engineering* (10th ed.), Pearson, 2015.

[10] Pressman, R. S., & Maxim, B. R., *Software Engineering: A Practitioner's Approach* (8th ed.), McGraw-Hill Education, 2014.

[11] Schwaber, K., & Sutherland, J., *The Scrum Guide*, Scrum.org, 2020. Disponível em: `https://scrumguides.org/`, Acedido em junho de 2025.

[12] Boeck, P., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 2017.

[13] Albahari, J., & Albahari, B., *C# 10 in a Nutshell*, O'Reilly Media, 2021.

[14] Microsoft, *LINQ (Language Integrated Query)*, Disponível em: `https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/`, Acedido em junho de 2025.

[15] Docker Inc., *Docker Documentation*, Disponível em: `https://docs.docker.com/`, Acedido em junho de 2025.

[16] Figma Inc., *Figma: Interface Design Tool*, Disponível em: `https://www.figma.com/`, Acedido em junho de 2025.

[17] Mobatek, *MobaXterm Professional*, Disponível em: `https://mobaxterm.mobatek.net/`, Acedido em junho de 2025.

Glossário

API	Interface de Programação de Aplicações que permite a comunicação entre diferentes sistemas de software.
ASP.NET	*Framework* de desenvolvimento da Microsoft para a criação de aplicações *web* dinâmicas.
ASP.NET	Conjunto de tecnologias da Microsoft para o desenvolvimento de aplicações web dinâmicas na plataforma .NET.
Backend	Parte da aplicação responsável pela lógica de negócio, acesso a dados e regras do servidor.
C#	Linguagem de programação moderna, segura e orientada a objeto que abrange desde recursos de alto nível, como registos orientados a dados, até recursos de baixo nível, como ponteiros de função.
Container	Unidade leve e portátil que agrupa uma aplicação e todas as suas dependências, garantindo que funcione consistentemente em diferentes ambientes.
Containerização	Processo de empacotar *software* e as suas dependências em *containers*, promovendo portabilidade, consistência e escalabilidade.
CSS	Linguagem de estilo utilizada para definir a apresentação visual de documentos HTML.
CVRP	Problema de Roteamento de Veículos com Capacidade, uma extensão do problema do caixeiro viajante que considera veículos com limite de carga.
DTO	Objetos de Transferência de Dados, usados para transportar dados entre camadas da aplicação.
Entity Framework Core	ORM da Microsoft que permite mapear objetos .NET para bases de dados relacionais.

Framework	Conjunto de ferramentas, bibliotecas e boas práticas que oferece uma estrutura base para o desenvolvimento de aplicações.
Frontend	Parte visual de uma aplicação com a qual o utilizador interage diretamente.
HTML	Linguagem de marcação usada para estruturar e apresentar conteúdos na web.
HTTP	Protocolo de comunicação usado para transferir dados na *web* entre clientes (como *browsers*) e servidores.
i18n	Processo de internacionalização de software, permitindo a adaptação para diferentes idiomas e regiões.
IDE	Ambiente de Desenvolvimento Integrado que fornece ferramentas para programar, depurar e compilar código.
JavaScript	Linguagem de programação interpretada e dinâmica, amplamente utilizada no desenvolvimento web para criar comportamentos interativos em páginas. É suportada nativamente pelos navegadores e funciona principalmente no lado do cliente.
JSON	Formato leve de intercâmbio de dados, fácil de ler e escrever por humanos e máquinas.
JWT	Token Web em formato JSON utilizado para autenticação segura entre cliente e servidor.
LINQ	Linguagem de consulta integrada no .NET para manipulação de dados de forma declarativa.
MVC	Padrão de arquitetura de *software* que separa a aplicação em três componentes principais: *Model*, que gere os dados e regras de negócio, *View*, que apresenta os dados ao utilizador, e *Controller* que gere a interação entre a *view* e o *model*.

ORM	Técnica que permite manipular bases de dados relacionais por meio de objetos em linguagens de programação orientadas a objetos.
OSRM	Motor de cálculo de rotas de código aberto, baseado em OpenStreetMap.
SPA	Aplicação de Página Única que carrega uma única página HTML e atualiza dinamicamente os conteúdos sem recarregar a página inteira.
SQL	Linguagem padrão para gerenciamento e manipulação de bases de dados relacionais.
TPF	Sistema de Transportes Públicos Flexíveis, que visa melhorar a mobilidade em áreas com baixa densidade populacional.
TSP	Problema do Caixeiro Viajante, que busca encontrar o menor caminho possível para visitar um conjunto de cidades uma única vez.
TypeScript	Linguagem de programação desenvolvida pela Microsoft que é um superconjunto do JavaScript. Acrescenta tipagem estática e recursos modernos de desenvolvimento, ajudando a escrever código mais robusto e fácil de manter.
UBI	Universidade da Beira Interior, instituição de ensino superior localizada na Covilhã.
VPN	Rede Privada Virtual que permite uma ligação segura a uma rede pública através de canais encriptados.