Politécnico
Castelo Branco
Escola Superior
de Tecnologia

# Sistema para otimização dinâmica de transporte de utentes com necessidades especiais

## Projeto II

José Pedro Mateus Lourenço Jorge 20221507

Rafael Alexandre Pereira Adónis 20220395

**Orientadores**

Osvaldo Santos

Trabalho de Projeto apresentado à Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco, realizada sob a orientação científica do Doutor Osvaldo Arede dos Santos, do Instituto Politécnico de Castelo Branco.

**Setembro de 2025**

## Composição do júri

Presidente do júri

Doutor, Fernando Ribeiro

Vogais

Doutor, Osvaldo Santos

Professor Adjunto do Instituto Politécnico de Castelo Branco

Doutor, João Caldeira

Professor Adjunto do Instituto Politécnico de Castelo Branco

Doutor, Fernando Ribeiro

Professor Adjunto do Instituto Politécnico de Castelo Branco

# Resumo

Nas grandes cidades, a logística associada ao transporte de pessoas com necessidades especiais tem sido, frequentemente, um processo manual, demorado e ineficiente. Associações que oferecem suporte a este público enfrentam grandes desafios ao tentar organizar diariamente rotas, veículos, motoristas e os requisitos específicos de cada utente. Esses problemas operacionais acabam por impactar negativamente a qualidade dos serviços prestados, além de sobrecarregar os responsáveis pela gestão. Nesse contexto, surge a necessidade de uma solução tecnológica inovadora que automatize e otimize esses processos, tornando-os mais ágeis, eficazes e personalizados.

Este projeto procura responder a essa necessidade através do desenvolvimento de uma aplicação web destinada à gestão interna dos serviços de transporte de pessoas com limitações de mobilidade. A aplicação foi concebida para ser intuitiva e funcional, permitindo às associações gerir de forma eficiente motoristas, veículos, agendamentos, paragens e as necessidades específicas dos utentes. Assente num modelo de dados cuidadosamente estruturado, o sistema integra múltiplas tabelas relacionadas como Motorista, Veículo, Utente, Agenda e Caraterísticas de forma a garantir que a informação se encontra organizada de forma lógica e acessível.

A aplicação também inclui módulos para monitorizar a localização dos motoristas, garantindo que os recursos estejam sempre disponíveis e em condições para cumprir as rotas. Outro destaque é a possibilidade de visualizar as paragens realizadas em cada viagem, ajudando a auditar e melhorar continuamente os serviços prestados.

Para complementar a plataforma web, será também desenvolvida uma aplicação móvel destinada aos motoristas, com o objetivo de facilitar o seu trabalho diário. Esta aplicação fornecerá informações cruciais como o percurso a seguir, os pontos de paragem e a identificação dos utentes a recolher ou a deixar. Adicionalmente, a aplicação permitirá a comunicação com os clientes através do envio automatizado de SMS com estimativas da hora de chegada, contribuindo para uma experiência mais informada, transparente e eficiente.

Ao automatizar as tarefas anteriormente feitas manualmente, como a alocação de motoristas ou a definição de rotas otimizadas, o sistema reduz o esforço administrativo. Isso resulta em maior eficiência operacional, redução de custos e, sobretudo, na melhoria da experiência para os utentes que dependem do serviço.

Com esta solução tecnológica, espera-se transformar a maneira como as associações gerem o transporte de pessoas com necessidades especiais, promovendo inclusão, acessibilidade e qualidade de vida para os utentes, ao mesmo tempo em que alivia a carga logística das organizações envolvidas. Este projeto representa um avanço significativo rumo à modernização dos serviços de transporte assistido.

## Palavras-chave

Transporte, Otimização, Rotas, Utentes, Acessibilidade

# Abstract

In large cities, the logistics associated with transporting people with special needs has often been a manual, time-consuming, and inefficient process. Associations providing support to this audience face significant challenges when attempting to organize routes, vehicles, drivers, and the specific requirements of each user on a daily basis. These operational issues negatively impact the quality of services provided and overload those responsible for management. In this context, there arises the need for an innovative technological solution that automates and optimizes these processes, making them faster, more efficient, and tailored.

Our project aims to address this need by developing a web application for the internal management of transportation services for people with mobility limitations. The application is designed to be intuitive and functional, allowing associations to efficiently manage drivers, vehicles, schedules, stops, and the specific needs of users. Based on a carefully designed data model, the system integrates multiple related tables, such as Driver, Vehicle, User, Schedule, and Characteristics, to ensure that information is organized in a logical and accessible manner.

The application also includes modules to monitor the status of vehicles and drivers, ensuring that resources are always available and in suitable condition to carry out routes. Another highlight is the ability to visualize the stops made during each trip, helping to audit and continuously improve the services provided.

To complement the web platform, a mobile application aimed at drivers will also be developed, with the aim of facilitating their daily work. This application will provide crucial information such as the route to follow, stopping points and the identification of users to be picked up or dropped off. In addition, the application will allow communication with customers through the automated sending of SMS with estimated arrival times, contributing to a more informed, transparent and efficient experience.

By automating tasks previously performed manually, such as assigning drivers or defining optimized routes, the system reduces administrative effort. This results in greater operational efficiency, cost reduction, and, above all, improved experience for the users who depend on the service.

With this technological solution, we aim to transform how associations manage the transportation of people with special needs, promoting inclusion, accessibility, and quality of life for the users, while also alleviating the logistical burden on the organizations involved. This project represents a significant step toward the modernization of assisted transportation services.

## Keywords

Transport, Optimization, Routes, Users, Accessibility

# Índice geral

# Índice de figuras

# Lista de tabelas

## Lista de abreviaturas, siglas e acrónimos

API: *Application Programing Interface*

CRUD: *Create Read Update Delete*

DOM: *Document Object Model*

BD: Base de dados

SDK: *Software development kit*

*JSON: JavaScript Object Notation*

# 1. Introdução

Esta secção apresenta o problema em estudo, os objetivos do projeto, a planificação do trabalho realizado e a estrutura do presente documento.

## 1.1. Contexto

Nas grandes cidades, a gestão diária do transporte de utentes de associações de apoio a pessoas com necessidades especiais assenta, tipicamente, num processo logístico manual, moroso e ineficiente.

Neste contexto, torna-se útil uma solução tecnológica inovadora que permita automatizar a gestão interna dos serviços de transporte de pessoas com mobilidade reduzida. Para tal, é essencial a criação de rotas eficientes para o transporte dos utentes. Este desafio surge, por exemplo, quando uma entidade de transporte tem de levar múltiplos utentes a diferentes locais de tratamento. Embora seja possível criar uma rota manualmente, esse processo torna-se progressivamente mais complexo à medida que se adicionam utentes com diferentes origens e destinos.

Assim, propôs-se o desenvolvimento de um sistema capaz de criar estas rotas de forma eficiente e dinâmica. O sistema deve ser capaz de identificar a melhor rota com base nos utentes que necessitam de transporte, desenhando um percurso otimizado para os recolher e entregar nos respetivos destinos. Importa referir que nem todos os utentes têm o mesmo destino e que, normalmente, existem vários veículos disponíveis para realizar os transportes, fatores que também devem ser considerados no cálculo da rota.

Adicionalmente, alguns utentes apresentam necessidades especiais ao nível do transporte, como a utilização de maca, cadeiras de rodas manuais ou elétricas, ou mesmo equipamento elétrico específico (por exemplo, um respirador artificial). O sistema deve, portanto, ser capaz de verificar se determinado veículo é compatível com os requisitos do utente.

Neste segundo semestre, surgiu o desafio de adaptar o projeto às necessidades reais de uma associação, a APPC[1]. A partir desse momento, foi necessário introduzir diversas alterações para satisfazer os requisitos da instituição. Entre essas alterações, destacam-se as mudanças significativas na base de dados e na interface web, de forma a acomodar as novas funcionalidades exigidas.

## 1.2. Âmbito e definição do problema

O objetivo fundamental deste projeto é criar uma solução informática que otimize a gestão diária do transporte de utentes de associações de apoio a pessoas com necessidades especiais.

A solução proposta consiste no desenvolvimento de um sistema de gestão de transporte de utentes com necessidades específicas, ou seja, um sistema capaz de planear rotas eficientes entre os locais de origem e os destinos dos utentes (por exemplo, hospitais). O objetivo é que essas rotas sejam otimizadas de forma a permitir o transporte simultâneo de vários utentes, mesmo que tenham destinos diferentes. Cabe ao sistema determinar a melhor rota para entregar todos os utentes nos respetivos locais.

Para além da otimização da rota, o sistema deve ter em consideração as diferentes necessidades de cada utente. Alguns podem viajar sentados, outros necessitam de maca, cadeiras de rodas (manuais ou elétricas) ou até de equipamentos especializados que requerem, por exemplo, alimentação elétrica. Com base nestas necessidades, o sistema deverá ser capaz de selecionar o veículo mais adequado para transportar os utentes de forma segura e eficiente.

É igualmente necessário transmitir esta informação ao condutor, permitindo-lhe saber que veículo utilizar, qual a rota a seguir, onde deve parar e se a paragem corresponde a uma recolha ou entrega, bem como os horários de cada operação. Adicionalmente, é importante fornecer informação aos utentes sobre o estado do seu transporte, como a hora prevista ou se o veículo já se encontra em viagem.

## 1.3. Objetivos de Projeto II

Para este segundo semestre, os principais objetivos consistiram na continuação do desenvolvimento do projeto, bem como na melhoria das funcionalidades implementadas anteriormente. Com o envolvimento direto da associação interessada, tornou-se necessário rever muitos dos elementos já desenvolvidos, em especial a interface web, o que consumiu uma parte significativa do tempo de desenvolvimento inicial desta fase.

O sistema foi concebido com uma arquitetura modular, dividida em várias partes. A base de dados constitui o núcleo central da solução, onde são armazenadas todas as informações relevantes, incluindo dados sobre utentes, condutores, locais de destino, veículos e respetivos equipamentos, agendamentos e viagens. Para permitir a comunicação entre esta base de dados e as aplicações desenvolvidas, foi criada uma API, responsável por executar operações CRUD, autenticação e outras funcionalidades essenciais, como a gestão de rotas.

Sobre esta infraestrutura, foram construídas duas interfaces principais:

Uma **dashboard web de administração**, acessível a partir de qualquer navegador, que permite a criação e gestão de utentes, veículos, agendamentos e outras entidades do sistema;

Uma **aplicação móvel destinada aos motoristas**, que disponibiliza os pontos de paragem da viagem, permite iniciar a navegação e oferece funcionalidades como envio de mensagens SMS para utentes, bem como o registo da localização em tempo real.

Para atingir os objetivos propostos para o Projeto II, foi necessário concluir ou implementar os seguintes pontos:

- Reformulação da interface Front-end com base nos requisitos da associação;
- Reestruturação da base de dados para refletir os novos requisitos e suportar funcionalidades futuras;
- Conclusão das páginas em falta no Front-end (ex.: viagens, agendamentos, mapas);
- Desenvolvimento das funcionalidades de agendamento na API;
- Desenvolvimento das funcionalidades de viagens na API;
- Criação da aplicação móvel para condutores;

Para a definição dos objetivos acima identificados, o plano de trabalho apresentado na Figura 1, é dividida pelas seguintes etapas abaixo descritas:

**Etapa 1 – Levantamento e adaptação às necessidades do cliente**
Com a entrada da associação interessada, foi necessário rever os requisitos do projeto, investigando as melhores práticas e tecnologias.

**Etapa 2 – Reestruturação da base de dados**

Foi realizada uma atualização do Diagrama Entidade-Relacionamento (ER), de modo a ajustar a base de dados aos novos requisitos.

**Etapa 3 – Melhoria e expansão do Front-end**

Foram feitas alterações visuais e funcionais à interface de administração web, tanto em componentes existentes como na criação de novas páginas. O desenvolvimento foi feito com React, privilegiando a acessibilidade e usabilidade.

**Etapa 4 – Desenvolvimento e integração das funcionalidades na API**

Foi desenvolvida a lógica necessária para suportar as funcionalidades de agendamentos e viagens na API, garantindo que os dados fossem corretamente tratados entre o front-end, a API e a base de dados.

**Etapa 5 – Criação da aplicação móvel para motoristas**

Com foco em Android, esta etapa envolveu a implementação de funcionalidades como login, introdução de ID da viagem, visualização das paragens, navegação turn-by-turn e envio de mensagens para os utentes.


**Figura 1 -** Diagrama de Gantt[2]

## 1.4. Organização do documento

Este documento diz respeito exclusivamente ao trabalho desenvolvido no segundo semestre, no âmbito do Projeto II, ainda que sejam feitas algumas referências ao Projeto I, uma vez que este teve de ser alterado e/ou melhorado.

O relatório está organizado em capítulos de forma a facilitar a compreensão do trabalho realizado. No primeiro capítulo, é apresentado o enquadramento do projeto, o seu âmbito e os objetivos definidos para o segundo semestre.

O segundo capítulo descreve as alterações efetuadas ao trabalho previamente desenvolvido. Este capítulo inicia-se com as modificações introduzidas na arquitetura do sistema, segue-se a descrição das alterações realizadas na base de dados e, por fim, as melhorias aplicadas ao front-end.

O terceiro capítulo aborda o desenvolvimento relacionado com o agendamento dos transportes, explicando como os dados são armazenados na base de dados e de que forma o front-end permite a criação desses agendamentos.

O quarto capítulo descreve o processo de criação das viagens, detalhando as ações que o utilizador deve realizar no front-end, bem como toda a lógica implementada na API para gerar e guardar as viagens. Neste capítulo são ainda apresentados alguns testes realizados, com a respetiva análise de resultados e identificação de problemas encontrados.

O quinto capítulo incide sobre o desenvolvimento da aplicação móvel, apresentando as suas funcionalidades e a forma como foram implementadas. Esta conta ainda com uma demonstração das funcionalidades da mesma.

Por fim, o relatório inclui uma conclusão, onde se destaca o trabalho desenvolvido, é feita uma análise crítica e é sugerido possível trabalho futuro. O documento termina com a listagem das referências bibliográficas utilizadas.

# 2. Alterações e melhorias no trabalho de projeto I

Neste segundo semestre, foi necessário realizar várias alterações em partes do projeto já desenvolvidas. Estas mudanças deveram-se não só ao envolvimento da associação interessada no projeto, como mencionado anteriormente, mas também à necessidade de preparar o sistema para funcionalidades futuras e de melhorar trabalho previamente efetuado.

Para isso, foram feitas alterações à base de dados, ao front-end e foram corrigidos alguns bugs na API.

## 2.1. Alterações na arquitetura do sistema

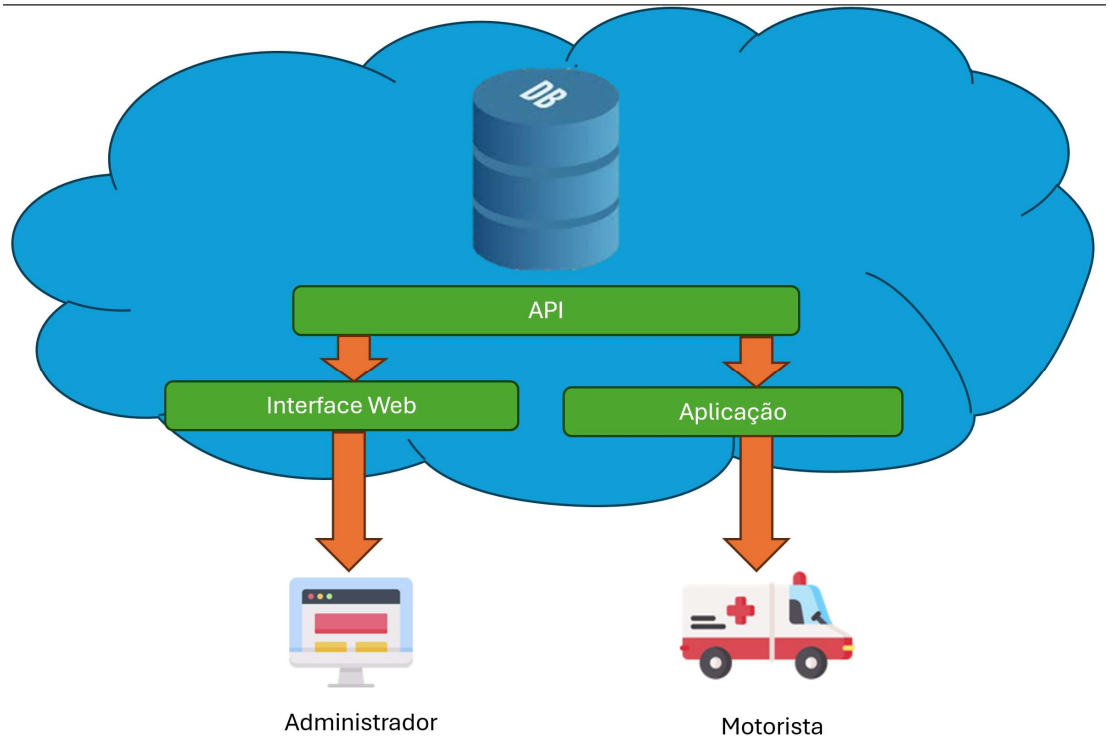


**Figura 2 -** Diagrama arquitetura da solução

Em relação à arquitetura da solução, esta manteve-se semelhante à do semestre anterior. Apesar da redução do número de aplicações (de motorista e utente para apenas motorista), esta alteração resultou, sobretudo, do feedback da APPC, que referiu que muitos dos seus utentes não conseguem utilizar smartphones ou têm grandes dificuldades em fazê-lo.

Embora a aplicação para os utentes não tenha sido desenvolvida, foi implementado um método alternativo e mais simples para manter os utentes informados sobre as suas viagens, abordagem que será detalhada na secção dedicada à aplicação do motorista.

Outro componente que sofreu alterações significativas foi a base de dados, que foi praticamente reconstruída, com a adição de novos campos e tabelas, de forma a

adaptar-se melhor às necessidades do projeto. Algumas dessas tabelas ainda não estão a ser utilizadas, mas foram criadas com vista a suportar funcionalidades futuras.

A interface web também foi alvo de várias melhorias, com o objetivo de se tornar mais intuitiva e fácil de utilizar. Foram adicionadas novas funcionalidades, como a integração de mapas, e concluídas páginas que não tinham sido implementadas no primeiro semestre, nomeadamente as secções de agendamentos e viagens.

Nos subcapítulos seguintes, serão detalhadas as alterações realizadas e as novas funcionalidades implementadas.

## 2.2. Base de dados

Como referido anteriormente, a base de dados sofreu alterações significativas, nomeadamente ao nível da gestão de utilizadores. Uma das principais mudanças foi a introdução da tabela “Role”, responsável por armazenar o tipo de cada utilizador (condutor, paciente ou cuidador). Foi também criada a tabela “User”, onde são guardadas informações gerais como o email e a palavra-passe.

Posteriormente, a tabela “User” estabelece ligações com tabelas específicas consoante o tipo de utilizador, onde são armazenadas informações mais detalhadas. Por exemplo, no caso dos pacientes, dados como a morada e a data de nascimento são guardados na tabela “Patient”.

Para além destas, foram realizadas várias outras alterações, que serão abordadas nos pontos seguintes.

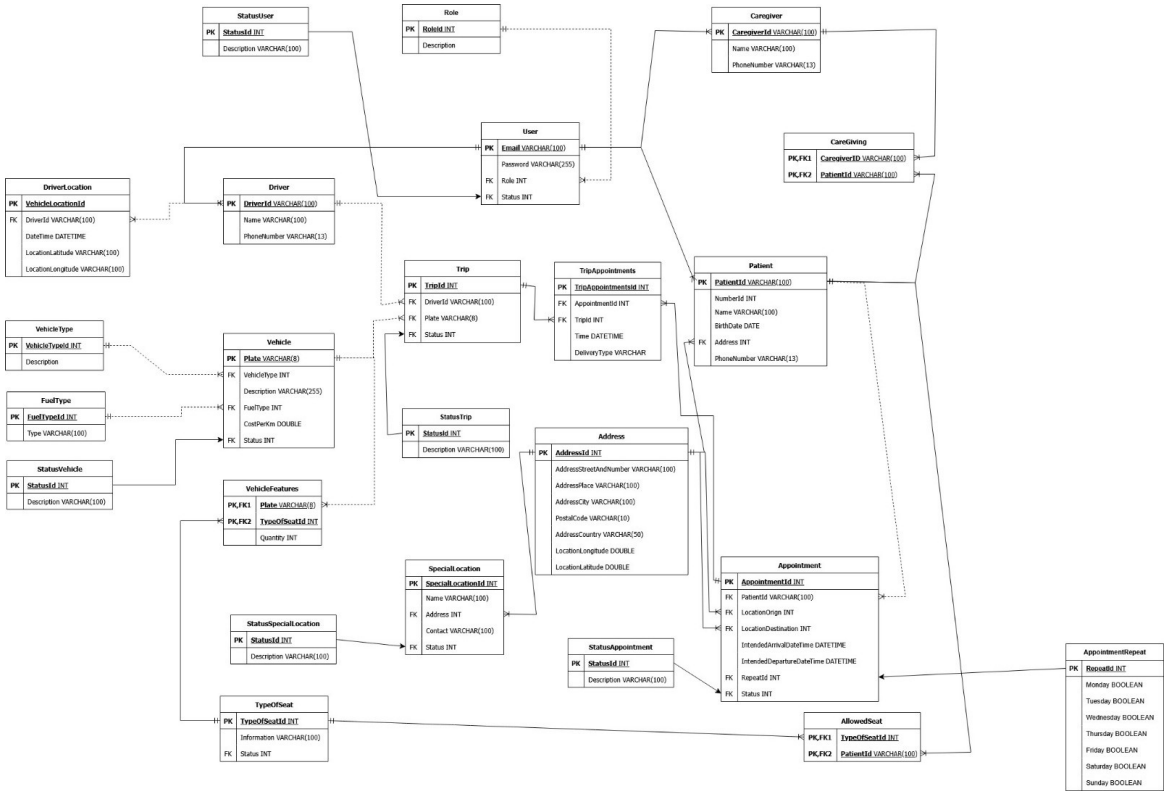


**Figura 3 -** Diagrama de classes

## 2.2.1. Tabela “Driver”

Nesta tabela apenas se encontra o nome e número de telefone do motorista, esta encontra-se ligada à tabela “DriverLocation” que tem como objetivo guardar a localização do motorista ao longo do tempo. Isto pode ser útil para no futuro se criar uma página onde se pode ver a localização de todos os veículos, pode também ser utilizado para ter a certeza de que as rotas estão a ser realizadas corretamente.

### 2.2.2. Tabela “Trip”

A tabela viagem foi bastante modificada tendo agora apenas uma ligação ao motorista, ao veículo e um status. Uma das principais diferenças é como as viagens são guardadas, isto é, em vez de se guardar os pontos de paragem da viagem como se tinha na base de dados do semestre passado agora guarda-se os agendamentos por ordem na tabela “TripAppointments” facilitando não só a busca das localizações, mas também saber a que utente esse agendamento está associado.

### 2.2.3. Tabela “Address”

Todas as localizações foram guardadas numa tabela única que guarda informações normais como a rua, número, cidade, código postal e país, mas também guarda coordenadas GPS para serem depois utilizadas tanto no front-end como na aplicação.

### 2.2.4. Tabela “SpecialLocation”

Esta tabela serve para guardar localizações de interesse como por exemplo hospitais, centros de tratamento a localização do centro da APPC entre outros, esta tabela conta com um campo de contacto, um status e uma ligação à tabela “Address”.

### 2.2.5. Tabela “Vehicle”

A tabela Veículo sofreu várias alterações, não só foram adicionadas várias ligações a outras tabelas para ajudar a categorizar os veículos com a tabela “FuelType” e “VehicleType” para guardar informações como o tipo de combustível que o veículo utiliza e que tipo de veículo é (pesado, ligeiro, etc.). A parte das características do veículo continua similar ao primeiro semestre, alterando-se apenas o nome das tabelas.

### 2.2.6. Tabela Agenda

Na tabela Agenda, estão contidas informações como o seu identificador único (IdAgendamento) juntamente com identificador único da tabela Utente e Paragem (NumCC e IdParagem) pois nesta tabela serão apresentadas as informações que o adminstrador registar no front-end. Além disso, também é necessário o dia com a data e hora (DataHora) e um campo para identificar o estado da viagem caso esteja pendente, a realizar ou realizada (Estado).

### 2.2.7. Tabela “Caregiver” e “CareGiving”

Estas tabelas são as tabelas que não estão a ser utilizadas como mencionado anteriormente, o objetivo destas tabelas é associar cuidadores a um ou mais utentes, por exemplo adicionar uma mãe que esteja a cuidar do seu filho.

### 2.2.8. Tabela “Appointment”

Na tabela “Appointment” é onde ficam guardados os agendamentos feitos, aqui fica o utente, o local de origem e destino, o dia e hora de Partida/Chegada, um Status e um campo Until, este campo serve para dizer se o agendamento se repete e até quando. Na tabela também existe uma ligação à tabela “AppointmentRepeat”, esta

funciona em conjunto com o campo Until permitindo assim marcar o agendamento apenas em dias específicos da semana.

## 2.3. Front-end

A parte do front-end teve bastante importância nesta segunda fase do projeto, uma vez que é através dela que o administrador consegue gerir toda a informação e garantir o bom funcionamento do sistema. Nesta etapa, grande parte da interface foi refeita, tanto a nível visual como funcional, com o objetivo de a tornar mais simples de utilizar e com uma navegação mais intuitiva.

Foi utilizada a linguagem JavaScript, com o apoio da biblioteca React, por ser leve, rápida e facilitar a criação de componentes reutilizáveis. Esta escolha permitiu um desenvolvimento mais organizado e a adição de novas funcionalidades de forma eficiente.

Foram concluídas páginas que tinham ficado por implementar no primeiro semestre, como as de agendamentos e viagens, e também foram adicionadas novas páginas, como a dos mapas. O sistema continuou a suportar todas as operações CRUD nas várias listas de dados.

Desde o início, procurou-se garantir que qualquer pessoa pudesse utilizar o sistema sem necessidade de explicações complexas. Para isso, foi adotada uma navegação simples, com uma barra lateral (sidebar) bem organizada, com ícones e menus claros. O objetivo foi sempre criar uma solução prática, funcional e de fácil aprendizagem.

Apresenta-se, de seguida, a interface desenvolvida, iniciando com a primeira interação no website: o sistema de login, ilustrado na Figura 4.

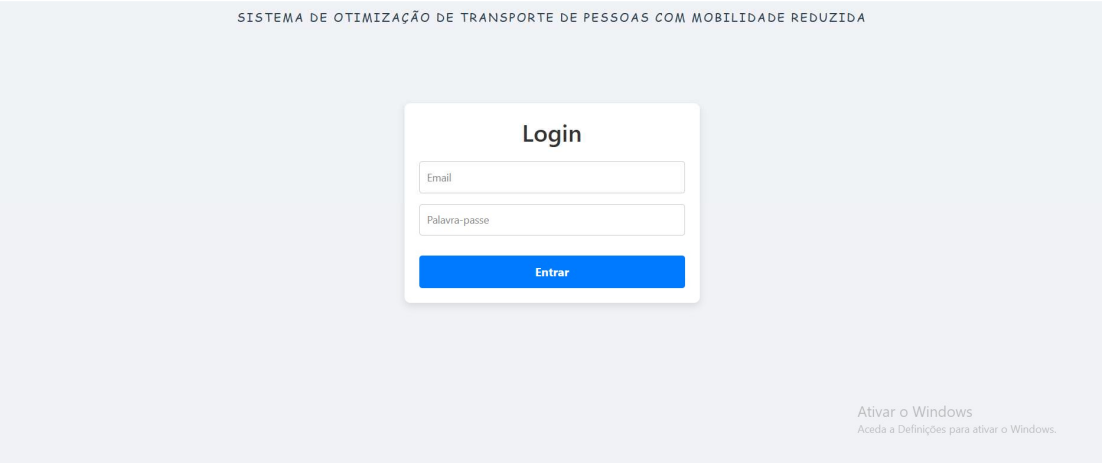


**Figura 4 -** Sistema de login

Após a inserção das credenciais por parte do administrador a página inicial será semelhante à da Figura 5, em que será possível ter acesso às diversas funcionalidades do sistema:

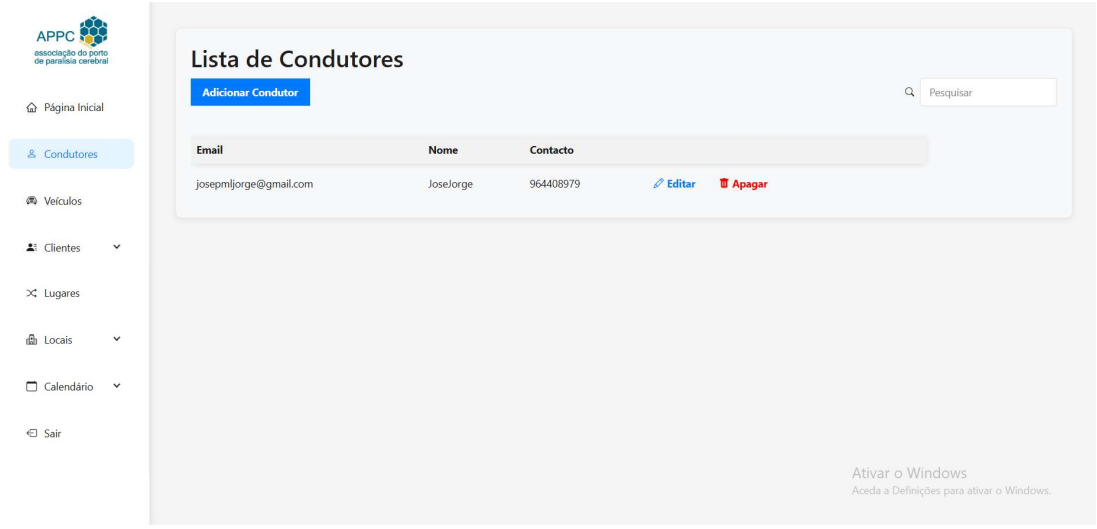


**Figura 5 -** Página Inicial

Com o objetivo de desenvolver uma interface web minimalista, mas simultaneamente intuitiva e de fácil aprendizagem, foi adotada uma estrutura baseada numa sidebar, conforme ilustrado na Figura 6. Esta abordagem permite uma visualização clara e acesso direto às funcionalidades principais do sistema. A barra lateral revela-se especialmente útil para o administrador, uma vez que facilita a navegação entre as diversas secções, através de ícones com menus e submenus organizados de forma lógica.

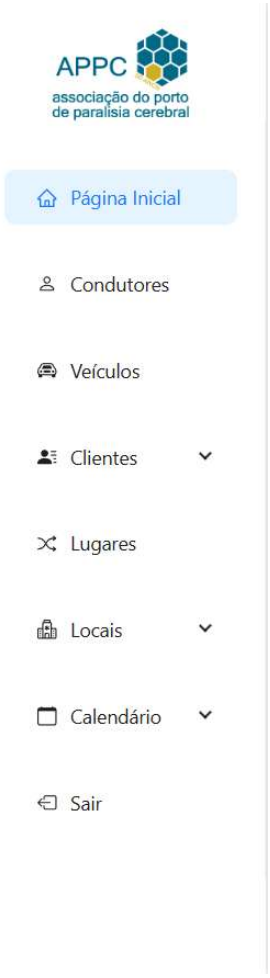


**Figura 6 -** Sidebar

No contexto de listagem dos dados, através da *sidebar* mencionada anteriormente o administrador tem acesso a todas as páginas que vão ser mostradas abaixo em que poderá encontrar as informações todas e modificá-las facilmente.

Podemos observar na Figura 7 a listagem dos condutores de forma organizada registados no sistema, incluindo o nome, email e contacto de cada um. Esta área permite ao administrador adicionar novos condutores, assim como editar ou remover os existentes, garantindo que a informação se mantém atualizada e completa.

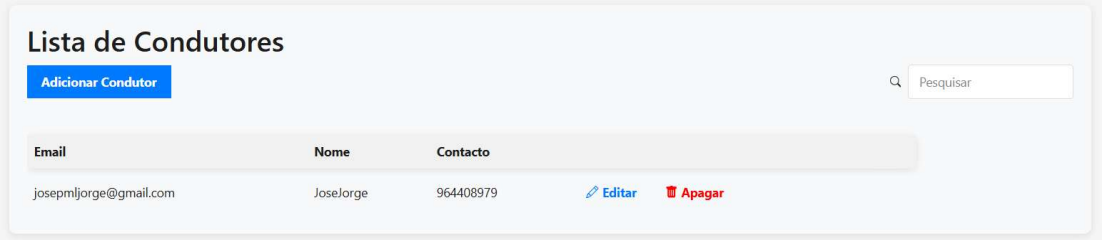


**Figura 7 -** Lista de Condutores

A página de lista de características, vista na Figura 8, permite ao administrador visualizar, editar ou remover as informações relativas às necessidades especiais dos utentes, como cadeiras de rodas ou macas. A interface apresenta uma tabela simples, com opções claras de edição e eliminação de registos, além de um botão de adição de novas características no topo.


**Figura 8 -** Lista de Características

Nesta página (Figura 9), estão listados todos os clientes registados no sistema, incluindo os respetivos contactos e as necessidades específicas de transporte. Através desta interface, é possível pesquisar rapidamente um cliente, bem como realizar operações de edição ou remoção de forma intuitiva.

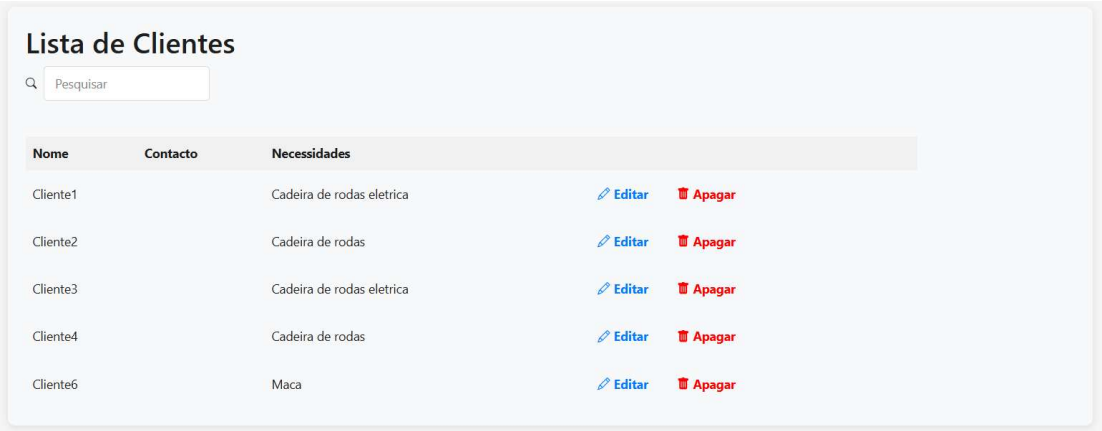


**Figura 9 -** Lista de Clientes

A lista de locais disponibiliza ao administrador a gestão de destinos relevantes, como hospitais ou centros de tratamento. Cada local apresenta informação de contacto associada e oferece funcionalidades para modificar ou eliminar os dados sempre que necessário, como pode ser visto na Figura 10.


**Figura 10 -** Lista de Locais

Na lista de veículos (Figura 11) é possível consultar todos os veículos disponíveis para transporte, incluindo detalhes relevantes como o tipo de viatura, matrícula e outras informações essenciais. Estes dados são fundamentais para que a API consiga posteriormente corresponder as características requeridas por cada viagem.

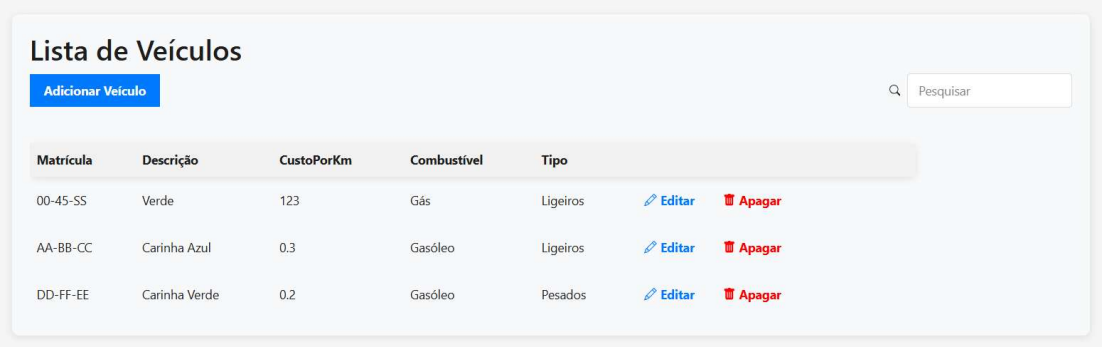


**Figura 11 -** Lista de Veículos

Relativamente ao sistema de mensagens, foi implementada uma dinâmica que exibe notificações textuais no topo da página sempre que ocorre um erro, um campo obrigatório não é preenchido ou uma operação (como edição, remoção ou adição) é concluída com sucesso. Estas mensagens são apresentadas com cores distintas consoante o tipo de feedback (positivo ou negativo) e permanecem visíveis durante três segundos, conforme demonstrado na Figura 12

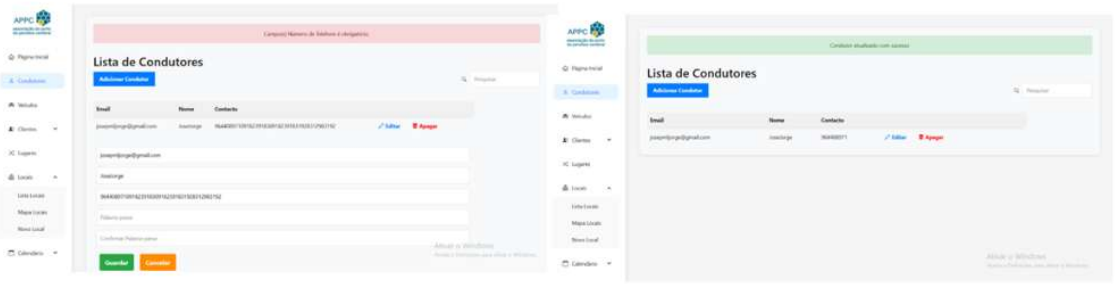


**Figura 12 -** Mensagem com Erro / Mensagem com Sucesso

### 2.3.1. Testes de Usabilidade

Tendo como objetivo garantir que a interface desenvolvida é de fácil utilização e que qualquer utilizador consiga navegar sem formação prévia, foram realizados testes de usabilidade com um pequeno grupo de utilizadores. Estes testes visaram identificar possíveis dificuldades na interação com o sistema, validar a organização da interface e recolher feedback direto sobre a experiência de uso.

#### 2.3.1.1. Metodologia

Os testes foram realizados com 5 participantes, selecionados por representarem o perfil de utilizadores administrativos da instituição. Nenhum dos participantes recebeu formação prévia sobre o sistema, de modo a simular um primeiro contacto realista com a aplicação.

Os participantes possuem as seguintes caracterizações:

**P1**: Mulher, 29 anos, licenciada em Administração, experiência básica em informática (uso de Word e e-mail).

**P2**: Homem, 45 anos, ensino secundário completo, experiência moderada com aplicações de gestão.

**P3**: Mulher, 38 anos, licenciada em Gestão, utiliza diariamente software administrativo e folha de cálculo.

**P4**: Homem, 52 anos, 9.º ano de escolaridade, experiência limitada com computadores.

**P5**: Mulher, 33 anos, mestrado em Ciências Sociais, experiência regular em plataformas digitais.

**P6**: Mulher, 24 anos, licenciada em Serviço Social, utiliza diariamente software e Word.

Cada participante foi convidado a realizar as seguintes tarefas no sistema:

- Criar um novo utente;
- Criar um agendamento;
- Consultar a lista de agendamentos;
- Consultar o mapa de viagens.

Durante a realização das tarefas, foram observados os seguintes aspetos:

- Tempo necessário para completar cada tarefa;
- Número de erros cometidos;
- Necessidade de ajuda ou esclarecimentos;
- Comentários e sugestões.

### 2.3.1.2. Resultados

A tabela seguinte representa os resultados individuais por participante para cada métrica avaliada:

**Tabela 1 -** Resultados dos participantes

| Participante | Tarefa 1 (Tempo / Erros / Ajuda) | Tarefa 2 (Tempo / Erros / Ajuda) | Tarefa 3 (Tempo / Erros / Ajuda) | Tarefa 4 (Tempo / Erros / Ajuda) | Comentários / Sugestões |
|---|---|---|---|---|---|
| **P1** | 1m45s / 0 / Não | 1m30s / 0 / Não | 1m15s / 0 / Não | 1m40s / 0 / Não | Sugeriu destacar mais o botão de confirmação |
| **P2** | 2m05s / 1 / Sim | 2m20s / 1 / Sim | 1m50s / 0 / Não | 2m10s / 1 / Sim | Indicou que os ícones podiam ser maiores |
| **P3** | 1m20s / 0 / Não | 1m25s / 0 / Não | 1m10s / 0 / Não | 1m30s / 0 / Não | Considerou a navegação muito clara e intuitiva |
| **P4** | 2m30s / 2 / Sim | 2m45s / 1 / Sim | 2m00s / 1 / Sim | 2m35s / 1 / Sim | Sentiu dificuldades em localizar elementos da interface em diferentes ecrãs |
| **P5** | 1m35s / 0 / Não | 1m40s / 0 / Não | 1m20s / 0 / Não | 1m25s / 0 / Não | Comentou que as mensagens de erro são claras e ajudam a perceber se está no caminho certo |
| **P6** | 1m25s / 0 / Não | 1m25s / 0 / Não | 1m15s / 0 / Não | 1m40s / 0 / Não | Destacou o design agradável e organizado |

Após os resultados da primeira tabela foi construída uma nova para perceber as médias por tarefa dos participantes:

**Tabela 2 -** Medias por tarefa

| Tarefa | Tempo Médio | Erros Médios | Necessidade de Ajuda |
|---|---|---|---|
| **Criar novo utente** | 1m55s | 0,6 | 2/6 |
| **Criar agendamento** | 2m | 0,4 | 2/6 |
| **Consultar lista de agendamentos** | 1m35s | 0,2 | 1/6 |
| **Consultar mapa de viagens** | 1m48s | 0,4 | 2/6 |

.

### 2.3.1.3. Conclusão dos Testes

Os resultados demonstram que todos os participantes foram capazes de concluir com sucesso as tarefas propostas, ainda que alguns necessitassem de apoio. O tempo médio de execução foi igual ou inferior a 2 minutos por tarefa, o que indica boa eficiência da interface.

A barra lateral foi considerada intuitiva pela maioria, embora um dos participantes (P4) tenha registado dificuldades consistentes em localizar elementos da interface em diferentes ecrãs, sugerindo maior destaque visual para botões, ícones e opções de navegação. As mensagens de erro e de sucesso mostraram-se eficazes na orientação das ações.

De forma geral, os testes confirmam que a interface é clara, funcional e acessível mesmo a utilizadores com menor experiência digital. As sugestões recolhidas, como maior destaque em botões, ícones e elementos de navegação foram integradas no desenvolvimento final, contribuindo para uma experiência de utilização mais simples e orientada.

# 3. Agendamento de transportes

A partir deste capítulo, será descrito o trabalho desenvolvido no âmbito de Projeto II. Uma das primeiras funcionalidades a ser implementada foi o agendamento de viagens. Para tal, foi necessário efetuar alterações tanto na API como no front-end.

No contexto dos agendamentos, como se pode observar na Figura 13, existe um formulário para a criação de um novo agendamento, no qual o administrador seleciona o local de origem e o destino de um utente, bem como a hora de chegada ou de partida. Apenas uma das opções deve ser selecionada, pois o sistema interpreta o valor inserido de acordo com o seu tipo: ao escolher uma hora de partida, o sistema assume que o utente deverá sair do local indicado exatamente àquela hora; ao selecionar uma hora de chegada, entende-se que o utente deverá estar no destino à hora especificada.

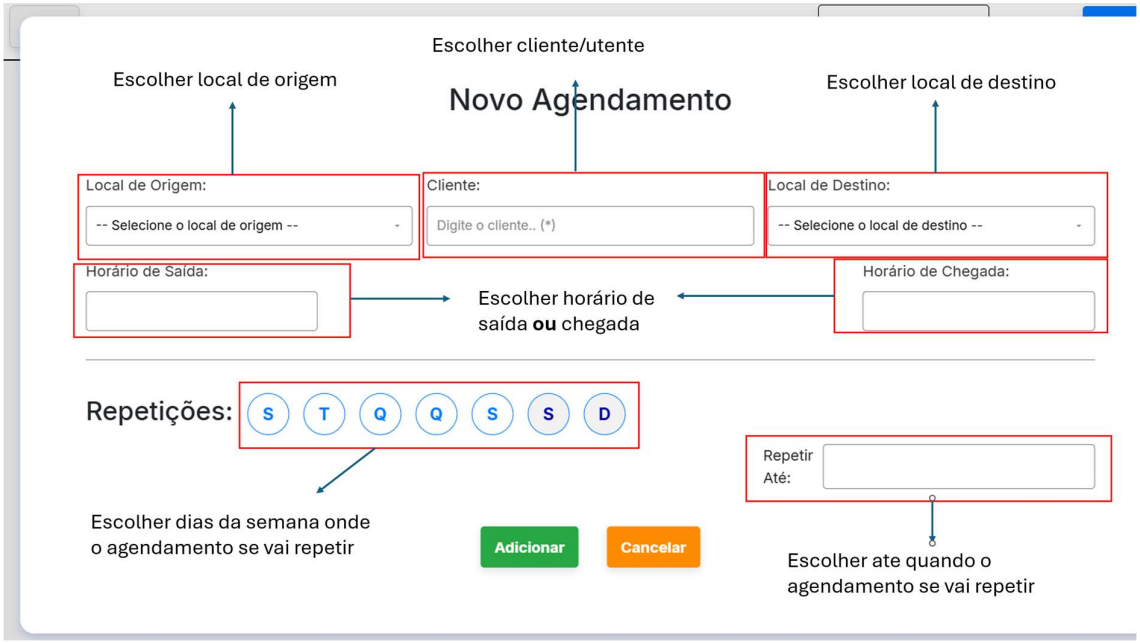


**Figura 13 -** Formulário de criação de agendamento

Um exemplo que ajuda a compreender este funcionamento consiste em imaginar que todos os utentes devem estar na APPC às 9:00. Nesse caso, o administrador insere esse valor como hora de chegada, e o sistema calcula automaticamente a hora de saída individual de cada utente, com base na rota que será posteriormente gerada.

O algoritmo responsável pela criação dos agendamentos é particularmente complexo nos casos em que está envolvida a repetição recorrente de agendamentos. A sua execução é realizada dentro de uma transação de base de dados, de forma a garantir a integridade e consistência dos dados em caso de falha.

O primeiro passo do algoritmo consiste na validação dos dados recebidos. É verificado se o corpo do pedido contém informação válida e se a data final do agendamento (“Until”) não ultrapassa o limite de um ano a partir da data atual, de forma a garantir o controlo da quantidade de dados armazenados.

Segue-se a determinação dos locais de origem e destino. Caso o identificador do local seja “-1”, tal indica que o local corresponde à morada do utente, sendo necessário consultar a base de dados para obter o ID da morada associada a esse utente.

Se o agendamento for recorrente, é efetuada a inserção de uma nova entrada na tabela “AppointmentRepeat”, onde são armazenados os dias da semana selecionados para repetição (por exemplo, segunda, quarta e sexta-feira). Com base nesses dados, o algoritmo itera sobre os dias entre a data inicial e a data final, verificando, em cada iteração, se o dia atual coincide com um dos dias da semana definidos. Sempre que essa condição se verifica, é criado um novo registo na tabela “Appointment”.

Durante esse processo, a informação relativa ao utente, ao local de origem e ao destino permanece constante em todas as instâncias geradas. No entanto, os campos relativos à hora de chegada ou partida necessitam de ser ajustados, uma vez que são armazenados em conjunto com a data (no formato “DATETIME”). Assim, é necessário reconstruir o timestamp correspondente a cada dia de repetição, mantendo a hora original inalterada.

Por fim, nos casos em que o agendamento não é recorrente, a inserção é feita diretamente na tabela “Appointment”, sem necessidade de ciclos ou de referências à tabela “AppointmentRepeat”.

Toda a informação é enviada pelo front-end em formato JSON. Um exemplo de pedido para a criação de um novo agendamento pode ser visualizado na figura 14, onde são enviados os seguintes dados: o ID do utente, o ID do local de origem e o ID do local de destino, bem como a data de partida ou de chegada (ambas são enviadas, mas uma delas permanece a null, dependendo do tipo de agendamento).

São também enviados os nomes dos locais e do utente, embora estes não sejam utilizados pelo algoritmo.

Adicionalmente, são incluídas as informações relativas à repetição, nomeadamente o campo "until", que indica até quando o agendamento se deve repetir, e os dias da semana em que a repetição deve ocorrer.


**Figura 14 -** Exemplo documento de criação de agendamento

Foi ainda realizado um pequeno teste para demonstrar a maior rapidez ao utilizar a funcionalidade de repetição, em comparação com a inserção manual de cada agendamento individualmente.

Para o teste, foi cronometrado o tempo necessário para criar agendamentos para todas as segundas e quartas-feiras do mês de junho de 2025.

Os resultados obtidos podem ser consultados na tabela 3.

**Tabela 3 -** Teste de utilização das repetições

| Tempo utilizando repetições | Tempo sem repetições |
|---|---|
| 00:30.70 segundos | 02:07.30 segundos |

Como pode ser observado este mecanismo permite criar vários agendamentos de uma só vez. Por essa razão, além do limite de um ano imposto, é exigido que, aquando e feito um pedido GET das viagens para exibição no front-end, seja enviado o dia específico que se pretende visualizar. Desta forma, evita-se que toda a informação da base de dados seja carregada de uma só vez para o front-end.

Na figura abaixo, pode observar-se a página que lista os agendamentos. Esta apresenta o local de origem, o cliente associado e o local de destino. É também exibida a hora correspondente ao agendamento: caso tenha sido selecionado o horário de chegada, este é apresentado por baixo do local de destino; se for o horário de partida, surge por baixo do local de origem.

Foram ainda utilizadas cores para indicar o estado do agendamento, por exemplo, os agendamentos já associados a uma viagem são destacados a azul-ciano.

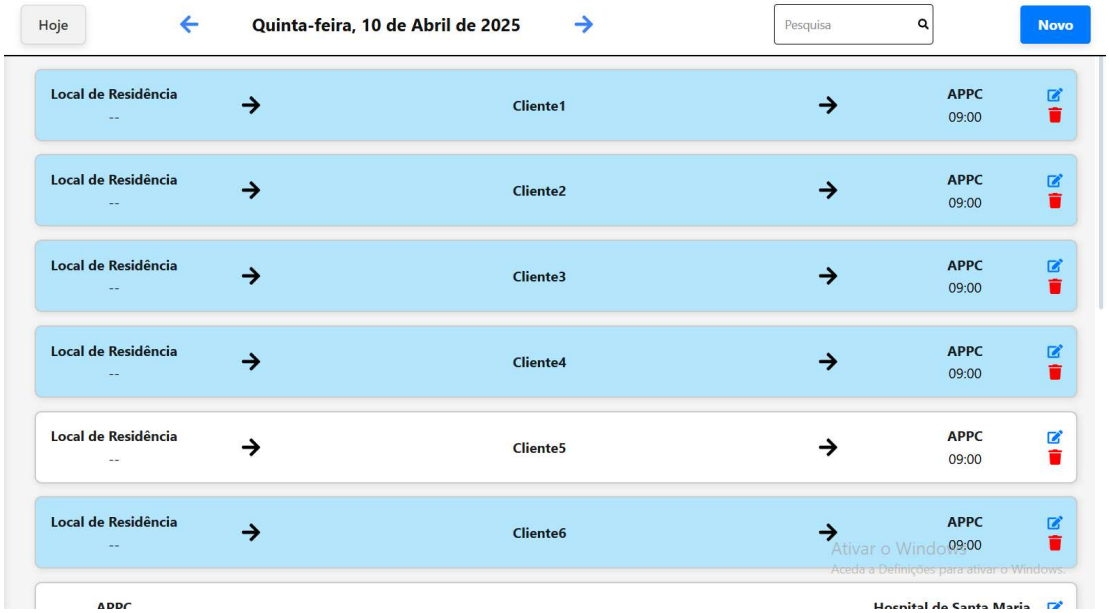


**Figura 15 -** Lista de Agendamentos

No caso de ser necessário eliminar ou editar um agendamento, o utilizador pode clicar no ícone do lápis para editar ou no ícone do caixote do lixo para eliminar, ambos localizados à direita de cada agendamento.

Ao clicar num destes botões, surge um pop-up a questionar se pretende editar/eliminar apenas esse agendamento ou todos os agendamentos recorrentes a partir do selecionado. Caso o agendamento não seja recorrente, a opção para eliminar os próximos agendamentos fica desativada.

Se o utilizador optar por editar ou eliminar apenas o agendamento selecionado, a API executa o processo normalmente, utilizando o respetivo ID do agendamento.

No caso de o utilizador querer eliminar todos os agendamentos recorrentes, a API realiza uma pesquisa por todos os agendamentos que tenham o mesmo ID de repetição e elimina-os.

Por outro lado, se o utilizador pretender editar todos os agendamentos recorrentes, é utilizado um algoritmo semelhante ao da inserção, tendo apenas o cuidado de aplicar as alterações apenas aos agendamentos cujo ID de repetição seja igual.

# 4. Criação das viagens

Havendo uma série de agendamentos de transporte para um determinado dia, o sistema deve ser capaz de calcular quais os veículos a utilizar, que utentes devem ser transportados em cada veículo, bem como a rota e os horários de recolha e entrega de cada utente.

Para dar resposta a esta necessidade, foi desenvolvido o código responsável pela criação das viagens, assim como toda a componente visual associada. No contexto da gestão de viagens, foi também adicionado um botão na interface de listagem de viagens (visível na Figura 16) com a designação "Criar Viagem".


**Figura 16 -** Lista de Viagens

Após clicar no botão 'Criar Viagem', é apresentada uma lista com todos os veículos disponíveis para realizar o transporte. Por defeito, todos os veículos surgem selecionados. Depois de efetuada a seleção, o utilizador deve clicar no botão “Confirmar”, o qual envia o pedido à API.

Devido ao facto de o processo de criação da viagem poder demorar alguns segundos, o formulário poderá demorar um pouco a desaparecer após a submissão.


**Figura 17 -** Nova viagem

De seguida, após a API receber o pedido de criação de viagem com as informações referidas anteriormente, inicia-se o processamento dos dados. O primeiro passo consiste na obtenção dos agendamentos correspondentes ao dia especificado.

O processo de criação de uma viagem pode ser dividido em duas fases principais: o registo das informações na base de dados e a otimização da rota. Para esta

otimização foi utilizada a API de otimização do Mapbox[3], uma vez que já inclui diversas funcionalidades essenciais. Para além da própria otimização do percurso, a API permite associar capacidades aos veículos e necessidades aos shipments (termo utilizado pelo Mapbox para designar os elementos a transportar), o que possibilita resolver situações em que determinados utentes necessitam de equipamento específico no veículo.

Atualmente, o Mapbox disponibiliza duas APIs de otimização de rotas: a V1 e a V2. No presente projeto foi utilizada a V2, uma vez que esta inclui as funcionalidades previamente mencionadas. Ainda assim, a V1 também é capaz de gerar rotas com até 12 coordenadas. Apesar de a otimização realizada por esta versão ser mais simples, inclui algumas funcionalidades que não estão presentes na versão mais recente. Entre estas, destacam-se a possibilidade de especificar o lado da rua em que o ponto deve ser atingido ou, definir que a rota siga um determinado bearing (por exemplo, 45º), forçando o sistema a procurar trajetos que obedeçam a esse ângulo.

Por outro lado, a V2 oferece um conjunto mais alargado de funcionalidades, especialmente direcionadas para a gestão de transportes. Esta versão permite, por exemplo, definir pontos de origem e/ou destino, indicar as horas de início e fim de operação de cada veículo, introduzir paragens obrigatórias para descanso e configurar a política de carregamento ("loading_policy"), cujas opções disponíveis são "LIFO", "FIFO" e "ANY".

Além disso, a V2 possibilita escolher se se pretende otimizar entregas ("shipments") ou serviços("services"). No caso dos serviços, é possível especificar a duração de cada operação ou até o intervalo horário em que esta pode ser realizada. No caso das entregas, pode-se definir não apenas a origem e o destino, mas também características adicionais, como o tamanho do objeto, o tempo necessário para carga e descarga, ou até os horários em que o transporte pode ser efetuado. Para além disso, é possível associar requisitos específicos a cada entrega ou serviço.

Ambas as versões têm um limite de 300 pedidos por minuto, sendo que a V2 permite ainda até 1000 localizações por pedido. Em ambos os casos, estão incluídos 100 000 pedidos gratuitos por mês.

Ambas as APIs contam com poucos endpoints. A V1 disponibiliza apenas o seguinte pedido:

- GET https://api.mapbox.com/optimized-trips/v1/{profile}/{coordinates}.

A V2, sendo mais completa, disponibiliza três endpoints:

- um para o envio do pedido de otimização: POST https://api.mapbox.com/optimized-trips/v2,
- um para a obtenção da solução do problema: GET https://api.mapbox.com/optimized-trips/v2/{id},
- um para listar os IDs de todos os pedidos efetuados: GET https://api.mapbox.com/optimized-trips/v2.

Para que estes pedidos funcionem, é necessário incluir o token privado.

Tendo em conta estas informações, o primeiro passo para fazer um pedido consiste em formatar corretamente as informações a enviar à API.

Para que a API do Mapbox funcione corretamente, é necessário enviar os dados relativos às localizações, veículos e shipments num documento JSON estruturado. A primeira entrada deste documento é o campo “version”, que atualmente é sempre definido como '1'. Segue-se a lista de localizações, que inclui tanto os locais associados aos utentes como localizações especiais (por exemplo, hospitais). Cada localização contém um nome e coordenadas geográficas; no caso dos utentes, é utilizado o respetivo nome como identificador.

Posteriormente, são inseridas as informações relativas aos veículos, incluindo o nome (neste caso, a matrícula), o “routing_profile” (que define o tipo de trajeto a otimizar este podem ser driving, driving-traffic, cycling, walking) e as capacidades do veículo.

Por fim, são adicionados os dados dos shipments, que representam os transportes a realizar. Para cada “shipment”, é necessário indicar a origem, o destino, o horário e as necessidades do utente. O nome de cada “shipment” é construído a partir do ID do agendamento e do nome do utente, separados por um hífen (ex.: idAgendamento - Nome). Esta convenção facilita o tratamento posterior dos dados, tanto no processamento como na apresentação visual.

```
{
  "version": 1,
  "location": [
    {
      "name": "APPC Centro de Reabilitacao",
      "coordinates": [-8.573894814798138,41.16114845102496]
    },
    {
      "name": "Cliente1",
      "coordinates": [-8.661542710728668,41.155450811588366]
    }
  ],
  "vehicles": [
    {
      "name": "AA-BB-CC",
      "routing_profile": "mapbox/driving-traffic",
      "capacities": {
        "NumeroMaca": 1,
        "NumeroAssento": 4
      },
      "capabilities": [
        "Maca",
        "Assento"
      ]
    }
  ],
  "Shipments": [
    {
      "name": "Cliente1",
      "from": "Cliente1",
      "to": "APPC Centro de Reabilitacao",
      "size": {
        "NumeroMaca": 1
      },
      "requirements": [
        "Maca"
      ],
      "dropoff_times": [
        {
          "earliest": "2025-04-06T10:15:00Z",
          "latest": "2025-04-06T10:30:00Z",
          "type": "strict"
        }
      ]
    }
  ]
}
```


**Figura 18 -** Exemplo de pedido a API do Mapbox

Após o envio das informações para a API do Mapbox, é devolvida uma resposta em formato JSON, semelhante ao exemplo apresentado na Figura 19. Com base nesses dados, inicia-se o processo de registo da viagem na base de dados.

Primeiramente, é criada uma nova entrada na tabela "Trip", onde são guardadas informações como a matrícula do veículo atribuído à viagem. O ID do motorista apenas é associado mais tarde, quando este inicia a navegação na aplicação móvel.

O Mapbox pode gerar múltiplas viagens numa única resposta, o que pode ser verificado através da presença de mais do que uma rota na secção "routes" do documento JSON. De seguida, são processadas as informações relativas aos "stops". Nesta parte do documento encontra-se uma grande quantidade de dados, incluindo o "type" (que pode assumir os valores "depart", "arrival" ou "break"), a localização, o tempo estimado de chegada, bem como a distância já percorrida até esse ponto da rota.

Estas informações são então armazenadas na tabela TripAppointments, a qual estabelece a ligação entre as viagens e os agendamentos.

Em cada registo da tabela "TripAppointments" são guardados dados como a hora prevista, o tipo de operação (recolha ou entrega), o ID da viagem e o ID do agendamento. Esta associação é possível porque o nome de cada "shipment" enviado à API contém o ID do agendamento, o que permite recuperar essa informação diretamente, sem necessidade de criar uma nova tabela para representar as paragens. Assim, reutiliza-se a estrutura já existente dos agendamentos.

```
{
  "dropped": {
    "services": ["service-3", "service-4"],
    "shipments": []
  },
  "routes": [
    {
      "vehicle": "xyz",
      "stops": [
        {
          "type": "depart",
          "location": "warehouse",
          "eta": "2022-08-02T09:00:00Z",
          "odometer": 0
        },
        {
          "type": "service",
          "location": "location-1",
          "services": ["service-1"],
          "eta": "2022-08-02T14:42:00Z",
          "odometer": 100,
          "wait": 300
        },
        { "type": "break", "eta": "2022-08-02T14:42:00Z", "wait": 3600 },
        {
          "type": "service",
          "services": ["service-2"],
          "location": "location-2",
          "eta": "2022-08-02T15:42:00Z",
          "wait": 0,
          "odometer": 200
        },
        {
          "type": "arrive",
          "location": "warehouse",
          "eta": "2022-08-02T20:24:00Z",
          "odometer": 300
        }
      ]
    }
  ]
}
```


**Figura 19 -** Exemplo de resposta da API do Mapbox

Depois de todo este processo estar concluído, é possível visualizar as viagens criadas através do front-end. Para além de uma lista com todas as viagens (semelhante à lista de agendamentos) é também possível clicar numa viagem específica para obter mais informações detalhadas.

Essas informações incluem os utentes associados, as respetivas moradas, os horários previstos e a indicação de se se trata de uma recolha ou de uma entrega. Esta funcionalidade está ilustrada na Figura 20.

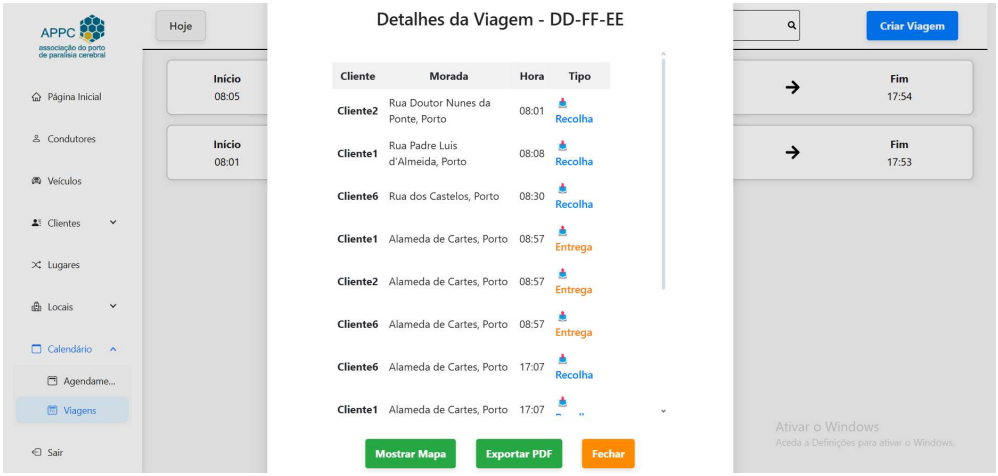


**Figura 20 -** Detalhes da Viagem

É ainda possível visualizar um mapa com os pontos da viagem, como ilustrado na Figura 21, ao clicar em "Mostrar Mapa" na página de detalhes da viagem.

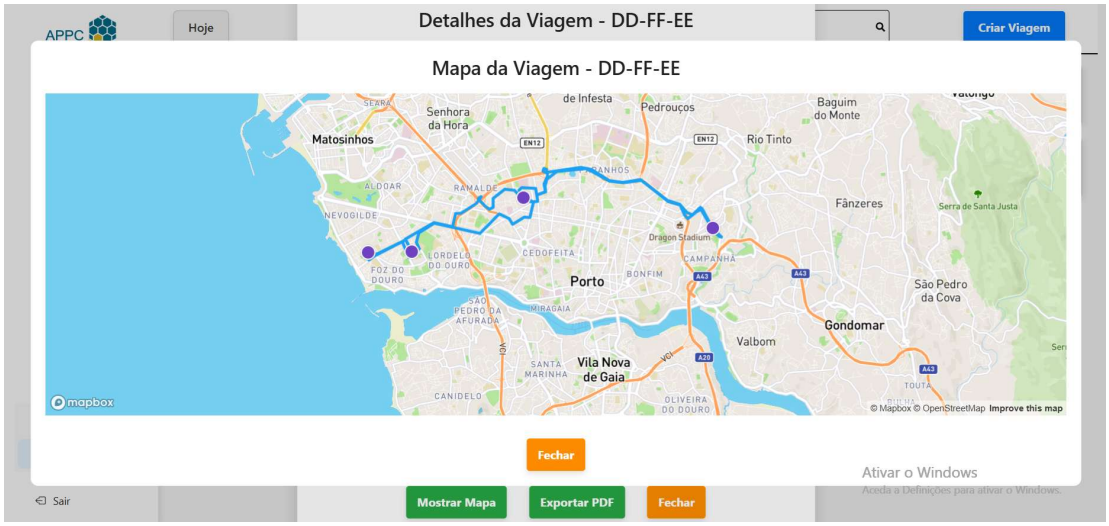


**Figura 21 -** Mapa da Viagem

Por fim, é ainda possível gerar um PDF, semelhante ao apresentado na Figura 22, através do botão "Exportar PDF". Este documento pode ser impresso e entregue ao condutor para o auxiliar no transporte, uma vez que contém todas as paragens e as respetivas informações.

O PDF inclui também o ID da viagem, que poderá ser utilizado posteriormente para localizar essa viagem na aplicação móvel.

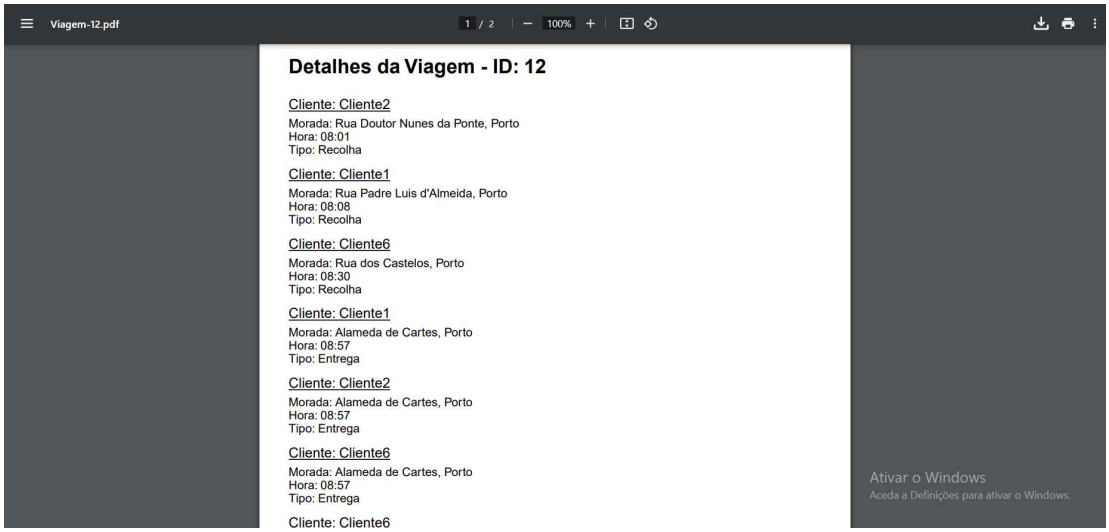


**Figura 22 -** PDF da Viagem

## 4.1. Testes de criação de viagens

Para terminar este tópico, serão discutidos alguns testes realizados, bem como alguns problemas encontrados na utilização da funcionalidade de otimização de rotas do Mapbox.

Para demonstrar o funcionamento da criação de rotas, foram criados vários utentes e adicionados pontos de interesse na zona de Castelo Branco. Na figura abaixo, pode ser visualizada uma tabela com os utentes utilizados nos testes.

**Tabela 4 -** Utentes criados para os testes

| Nome | Morada | Necessidade |
|---|---|---|
| **CB1** | Rua Maria Martins, Castelo Branco | Maca |
| **CB2** | Rua Adelino Semedo Barata, Castelo Branco | Maca |
| **CB3** | Rua Doutor Armindo Ramos, Castelo Branco | Cadeira de rodas elétrica |
| **CB4** | Rua Antônio Lourenço Barata, Alcains | Cadeira de rodas |

Na tabela seguinte, podem observar-se as localizações utilizadas.

**Tabela 5 -** Localizações criadas para os testes

| Nome | Morada |
|---|---|
| **Affidea Castelo Branco** | N112 |
| **Hospital Amato Lusitano** | Avenida Pedro Álvares Cabral, Castelo Branco |
| **EST** | Avenida do Empresário, Castelo Branco |

Foram ainda adicionados os seguintes veículos.

**Tabela 6 -** Veículos criados para os testes

| Matrícula | Capacidades |
|---|---|
| **DD-EE-FF** | Cadeira de rodas - 1, Cadeira de rodas elétrica - 1, Macas - 2 |
| **AA-BB-CC** | Cadeira de rodas - 1, Cadeira de rodas elétrica - 0, Macas - 3 |
| **GG-HH-II** | Cadeira de rodas - 4, Cadeira de rodas elétrica - 4, Macas - 0 |
| **JJ-KK-LL** | Cadeira de rodas - 1, Cadeira de rodas elétrica - 1, Macas - 1 |

Para o primeiro teste, foram criados agendamentos simulando uma possível rota diária de um motorista. Esta rota começa com a recolha de todos os utentes nas respetivas moradas, com destino à EST, onde deverão chegar às 8h30. No final do dia, todos os utentes saem da EST pelas 17h30 e regressam às suas habitações. Abaixo encontra-se uma figura que apresenta a lista de todos esses agendamentos.


**Figura 23 -** Agendamentos teste 1

De seguida, foi feita uma solicitação à API de otimização do Mapbox para otimizar esta rota utilizando todos os veículos disponíveis. O resultado foram quatro viagens, uma para cada veículo. A figura seguinte apresenta as quatro rotas geradas.

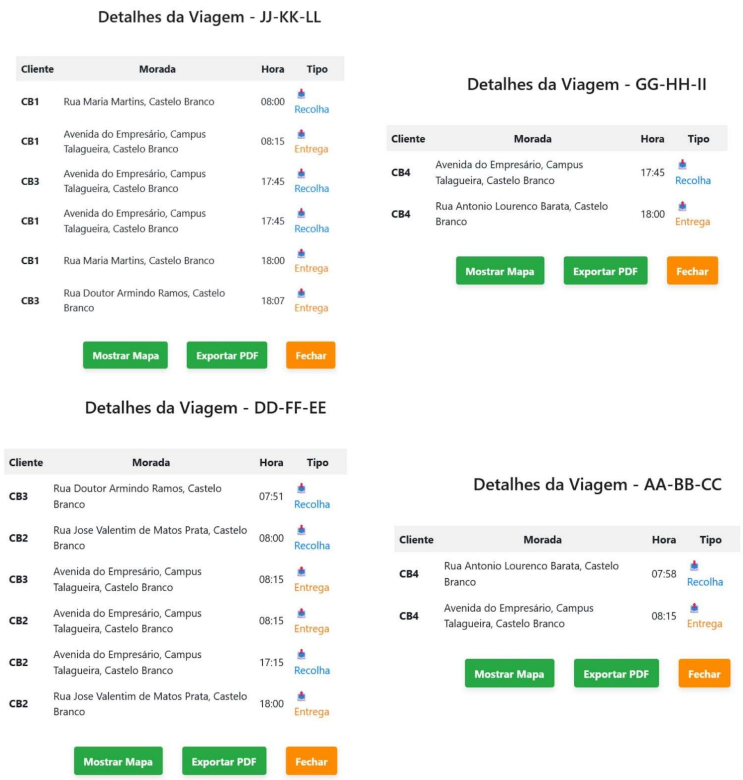


**Figura 24 -** Viagens teste 1

Na Figura 25 é também possível visualizar o mapa correspondente a cada viagem criada. No entanto, importa salientar que a rota apresentada neste momento não será necessariamente a mesma exibida na aplicação, uma vez que esta terá em conta tanto o trânsito no momento da geração da rota como a localização atual do motorista.

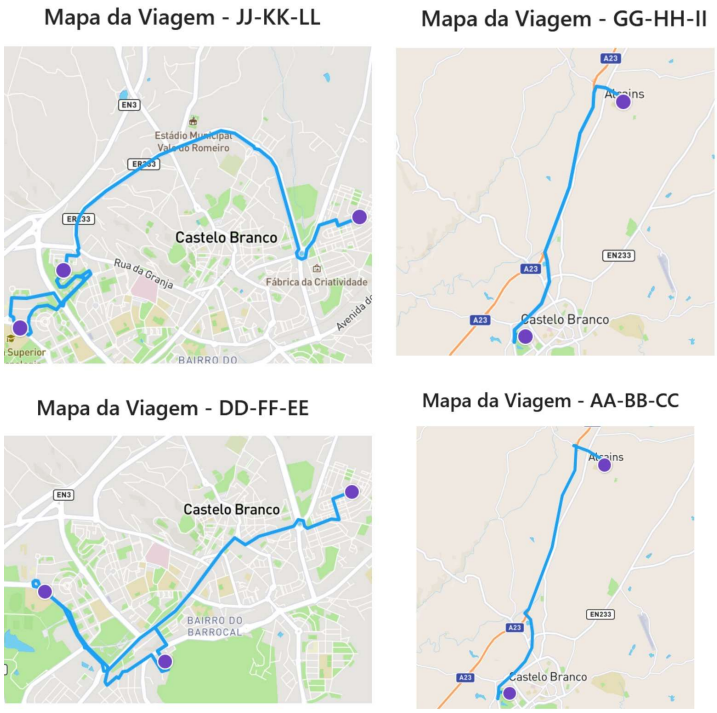


**Figura 25 -** Mapas das viagens criadas

Como se pode observar, todos os utentes foram deixados e recolhidos dentro do horário previsto (o sistema foi desenvolvido com uma tolerância de 15 minutos). Verifica-se também que os utentes residentes em Castelo Branco foram agrupados nos mesmos veículos, enquanto o utente cuja residência era mais distante realizou o trajeto sozinho.

De seguida, foi criada uma rota semelhante, mas desta vez foi removido o utente que vivia mais longe e adicionado um novo utente que, às 12h00, necessitava de se deslocar ao hospital, regressando à EST às 14h00. Os agendamentos desta nova configuração podem ser observados na imagem abaixo.


**Figura 26 -** Agendamentos teste 2

Neste caso, foram apenas criadas duas rotas, mas todos os agendamentos foram concluídos dentro do horário previsto.


**Figura 27 -** Viagem teste 2

De seguida, foi realizada a mesma rota, mas com a remoção do veículo DD-FF-EE. Neste caso, o sistema calculou que seria possível efetuar a rota utilizando apenas os veículos JJ-KK-LL e AA-BB-CC. Estas rotas podem ser observadas na figura 28.


**Figura 28 -** Viagens teste 3

Uma das funcionalidades do sistema é a capacidade de compreender que, quando um utente é entregue, o lugar que ocupava no veículo fica novamente disponível. Este comportamento pode ser observado nos agendamentos apresentados na figura 29, onde tanto o utente CB1 como o utente CB2 requerem o uso de uma maca.


**Figura 29 -** Agendamentos teste 4

Utilizando o veículo JJ-KK-LL, que dispõe apenas de uma maca, o sistema continua a conseguir gerar uma rota com sucesso.


**Figura 30 -** Viagem teste 4

No entanto, foi identificado um problema na utilização da API de otimização do Mapbox. Este problema ocorre quando o número de necessidades ultrapassa a capacidade total disponível. Por exemplo, num cenário semelhante ao anterior, em que os utentes CB1 e CB2 têm de ser transportados ao mesmo tempo e apenas está disponível o veículo JJ-KK-LL, que possui apenas uma maca. A figura seguinte apresenta um exemplo desta rota.


**Figura 31 -** Exemplo rota problemática

Segundo a documentação do Mapbox, quando não é possível encontrar uma solução viável, o sistema tenta remover um ou mais “shipments” para tentar gerar uma rota válida. Os “shipments” removidos deveriam ser listados na seção “dropped” do documento JSON devolvido. Esta informação está descrita na documentação original, que afirma o seguinte (traduzido para português):

” Esta propriedade lista shipments e services que não estão incluídos na solução. Isso pode ocorrer quando o mecanismo de otimização não consegue encontrar uma solução, mas, ao descartar uma remessa ou serviço da consideração, o mecanismo consegue encontrar uma solução. Se todas as remessas e serviços forem resolvidos, essas listas estarão vazias.”[3]

No entanto, este comportamento não foi observado nos testes realizados. Sempre que o número de necessidades era superior à capacidade disponível, o sistema devolvia a mesma resposta, como se pode verificar na Figura 32, sem incluir qualquer informação na secção “dropped”.

{"code":"unsolvable","message":"No solutions exist","ref":"437d658f-6829-49b2-b51f-d2c8847ddb8b"}

**Figura 32 -** Resposta da API do Mapbox

Outro problema foi identificado quando apenas o nome do utente era utilizado como nome do “Shipment”. Como a API de otimização do Mapbox foi concebida com a distribuição de encomendas em mente, situações como a representada na Figura 23 (em que um utente se desloca ao hospital e posteriormente regressa à EST) resultavam em comportamento incorreto: o sistema nunca “descarregava” o utente na EST, assumindo que este ainda teria outro destino final (neste caso, a sua habitação), e, por isso, não fazia sentido realizar a entrega intermédia.

Este problema foi resolvido ao passar a utilizar, como nome do “shipments”, uma combinação do ID do agendamento com o nome do utente. Esta abordagem fez com que a API do Mapbox considerasse os agendamentos como shipments distintos, devido à diferença nos nomes, permitindo assim o correto planeamento das entregas.

# 5. Aplicação móvel para os condutores

A aplicação móvel para os condutores foi um dos principais trabalhos deste segundo semestre. Esta tem como objetivo fornecer informações ao motorista sobre as viagens, como as paragens que devem ser efetuadas e o respetivo percurso. A aplicação é ainda capaz de enviar informações aos utentes através de SMS.

A aplicação foi desenvolvida no Android Studio com Kotlin e integra diversas bibliotecas para suportar todas as funcionalidades necessárias. As bibliotecas principais utilizadas foram as fornecidas pelo Mapbox, nomeadamente a biblioteca de mapas, que permite funcionalidades como a geração de rotas com base em pontos e tendo em conta o trânsito; a biblioteca de navegação, que disponibiliza os componentes necessários para implementar uma navegação Turn-by-Turn e a biblioteca navigation-compose, que fornece componentes visuais para facilitar o desenvolvimento da navegação.

Para além destas, foram também utilizadas outras bibliotecas, como o Retrofit em conjunto com o OkHttp, para a comunicação com a API.

Para realizar a comunicação tanto com a API desenvolvida para o projeto como com a API do Mapbox[4], é inicialmente necessário criar uma instância do Retrofit. Nesta instância é definido o nome a utilizar para efetuar os pedidos, é feita a associação ao ficheiro que contém todos os endpoints da API, e é configurado o URL base para os pedidos. Além disso, é especificada a conversão do JSON para data classes do Kotlin. Estas configurações podem ser observadas na figura 33.

```
val api: AuthService by lazy {
    Retrofit.Builder()
        .baseUrl("http://192.168.1.69:5000/api/v1/")
        //.baseUrl("http://10.0.2.2:5000/api/v1/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()
        .create(AuthService::class.java)
}

val mapBoxApi: MapBoxService by lazy {
    Retrofit.Builder()
        .baseUrl("https://api.mapbox.com/optimized-trips/v1/mapbox/driving-traffic/")
        .client(client)
        .addConverterFactory(GsonConverterFactory.create())
        .build()
        .create(MapBoxService::class.java)
}
```

**Figura 33 -** Instância do Retrofit

Na figura 34 pode-se ver o ficheiro como todos os endpoints da API necessários para o funcionamento da aplicação.

```
interface AuthService {
    @POST("login")
    fun login(@Body request: LoginRequest): Call<LoginResponse>

    @GET("viagens/viagem/{id}")
    suspend fun getTripDetails(
        @Header("Authorization") authHeader: String,
        @Path("id") tripId: String
    ): Response<TripDetails>

    @POST("register")
    fun register(
        @Body request: RegisterRequest
    ): Call<Void>

    @POST("motoristas/location")
    suspend fun enviarLocalizacao(
        @Body request: LocationRequestBody,
        @Header("Authorization") authHeader: String
    ): Response<Void>

    @PATCH("viagens/viagem/{id}")
    suspend fun addDriver(
        @Header("Authorization") authHeader: String,
        @Path("id") tripId: String,
    ): Response<Void>


}
```

**Figura 34 -** Interface da API

Na figura anterior é também possível verificar que é neste ponto que se especifica a data class para cada resposta. Numa data class é apenas necessário incluir os campos que se pretendem guardar; no entanto, é essencial associar cada campo ao respetivo atributo presente no JSON. Um exemplo pode ser observado na figura 35.

```
data class TripDetails(
    @SerializedName("TripId")
    val tripId: Int,

    @SerializedName("Plate")
    val plate: String,

    @SerializedName("TripAppointments")
    val tripAppointments: List<Appointment>
)

data class Appointment(
    @SerializedName("AppointmentId")
    val appointmentId: Int,

    @SerializedName("PhoneNumber")
    val phoneNumber: String,

    @SerializedName("Time")
    val time: String,

    @SerializedName("DeliveryType")
    val deliveryType: String,

    @SerializedName("Name")
    val name: String,

    @SerializedName("AddressStreetAndNumberOrigin")
    val addressStreetAndNumberOrigin: String,

    @SerializedName("AddressCityOrigin")
    val addressCityOrigin: String,

    @SerializedName("LocationLatitudeOrigin")
    val locationLatitudeOrigin: Double?,

    @SerializedName("LocationLongitudeOrigin")
    val locationLongitudeOrigin: Double?,

    @SerializedName("AddressStreetAndNumberDestination")
    val addressStreetAndNumberDestination: String,

    @SerializedName("AddressCityDestination")
    val addressCityDestination: String,

    @SerializedName("LocationLatitudeDestination")
    val locationLatitudeDestination: Double?,

    @SerializedName("LocationLongitudeDestination")
    val locationLongitudeDestination: Double?,

    var smsSent: Boolean = false
)
```

**Figura 35 -** Data class dos agendamentos

E também importante referir que as bibliotecas do Mapbox utilizadas não são as mais recentes (foi usada a versão 2, enquanto a mais atual é a versão 3). Esta escolha deveu-se à maior complexidade da nova versão e à disponibilidade limitada de documentação e exemplos. A versão 2 oferece mais conteúdo online e exemplos semi-completos no GitHub do Mapbox[5], o que facilitou o processo de desenvolvimento.

Esta aplicação recorre a várias tecnologias distintas para apresentar informações ao utilizador e processar dados em segundo plano. Para a interface visual, foi utilizado o Jetpack Compose em praticamente toda a aplicação, com exceção da componente de navegação, onde foi usada uma View. Esta exceção deve-se ao funcionamento do SDK de navegação do Mapbox.

Foram também utilizados serviços para gerir a localização do motorista e para o envio de SMS aos utentes.

Quando a aplicação é iniciada pela primeira vez, é necessário efetuar login com uma conta de motorista (esta conta apenas pode ser criada através do front-end). Após o login, a sessão é mantida durante 30 dias. Este período foi escolhido por corresponder à validade do token JWT gerado pela API, o qual é utilizado para

autenticação e validação tanto na aplicação como no front-end. A página de login pode ser visualizada na Figura 36.

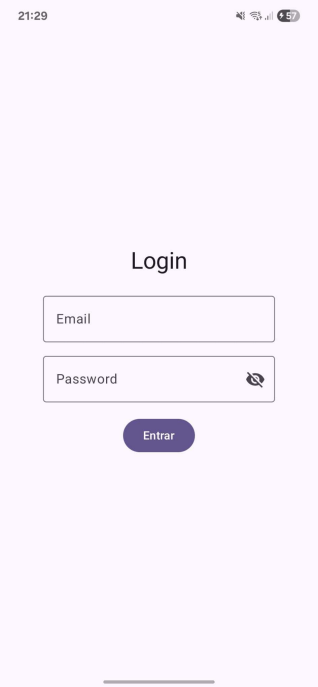


**Figura 36 -** Página de login aplicação

Após o login, é solicitado ao utilizador o ID da viagem que pretende seguir. Este ID pode ser obtido através do PDF gerado pelo front-end e deverá ser inserido no campo de texto apresentado na Figura 37. Assim que o ID for introduzido e o utilizador clicar em 'Continuar', a aplicação irá solicitar à API os dados correspondentes à viagem com esse ID, guardando todas as informações necessárias numa data class, que será posteriormente utilizada nas páginas seguintes.


**Figura 37 -** Página para inserir o ID aplicação

A página que surge de seguida é responsável por listar todos os pontos da viagem, conforme ilustrado na Figura 38. Nesta interface, são apresentados o ID da viagem, a matrícula do veículo atribuído e todas as paragens, incluindo o nome do utente, a morada, a hora e o tipo de paragem.

É ainda possível voltar atrás, caso o ID inserido esteja incorreto. É também nesta página que se pode iniciar a navegação Turn-by-Turn.

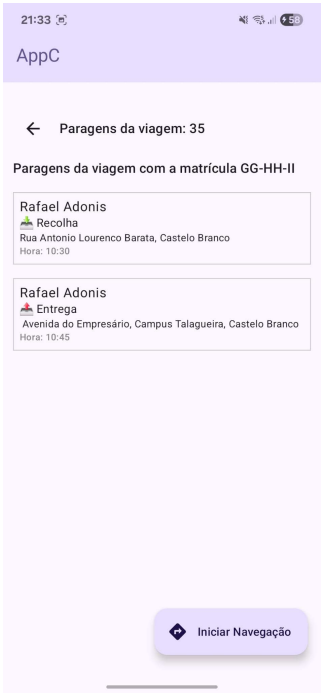


**Figura 38 -** Página lista de pontos da aplicação

Quando o utilizador pressiona o botão, a primeira ação que a aplicação realiza é verificar se possui todas as permissões necessárias, nomeadamente, acesso à localização, envio de notificações e envio de SMS. Caso alguma destas permissões não esteja concedida, a aplicação apresenta um pop-up a solicitar que o utilizador a autorize.

Dado que estas permissões são essenciais para o funcionamento da aplicação, esta não poderá prosseguir sem que todas estejam devidamente concedidas.

Outra ação realizada nesta fase é a associação do ID do motorista à viagem, ficando assim registado qual o condutor responsável por essa viagem

Após as permissões serem concedidas, dois componentes são inicializados: a navegação Turn-by-Turn e o serviço de localização. O serviço de localização, conforme mencionado anteriormente, é executado em segundo plano e é responsável por duas funcionalidades principais: enviar, a cada minuto, a localização do motorista para a base de dados através da API; e enviar a primeira mensagem de aviso aos utentes.

Quando este serviço é ativado, é criada uma notificação a informar o motorista de que a sua localização está a ser partilhada. Este serviço termina automaticamente quando a navegação é concluída ou a aplicação é encerrada.

Relativamente ao envio de SMS, esta funcionalidade foi implementada porque, conforme já referido, a aplicação destinada aos utentes não foi desenvolvida. Assim, optou-se por uma alternativa de comunicação através de SMS. Durante a viagem, são enviados dois SMS aos utentes, sendo o primeiro enviado pelo serviço de

localização. Este SMS é enviado no início da viagem a todos os utentes associados, informando que a viagem teve início e incluindo uma estimativa da hora de chegada do veículo. Esta estimativa é obtida através da API de otimização do Mapbox, sendo feita com base num pedido que utiliza a localização atual do motorista e a localização do utente.

O outro componente que é iniciado é a navegação Turn-by-Turn, cujo objetivo é fornecer uma navegação ponto a ponto, à semelhança de aplicações como o Google Maps. Uma das razões pelas quais se optou por desenvolver a navegação diretamente na aplicação, em vez de utilizar soluções externas, deve-se às limitações impostas por estas aplicações no número de pontos que podem ser utilizados para calcular a rota. Para além disso, o desenvolvimento interno permite maior flexibilidade na modificação e adição de funcionalidades adaptadas às necessidades específicas da associação.

A navegação implementada inclui todas as funcionalidades esperadas de uma aplicação deste tipo, nomeadamente instruções de manobras, informações por voz e a capacidade de recalcular a rota em caso de erro por parte do motorista. A rota é calculada no momento em que a navegação é iniciada, tendo em conta as condições de trânsito atuais. Todas estas funcionalidades foram possíveis graças às bibliotecas do Mapbox referidas anteriormente.

A página de navegação, apresentada na Figura 39, foi desenvolvida com base na View fornecida pelo Mapbox. No entanto, a secção que exibe o tempo e a distância até ao próximo ponto foi desenvolvida de raiz, uma vez que a componente padrão apenas considerava o destino final da viagem, não tendo em conta que a viagem podia ter várias paragens.

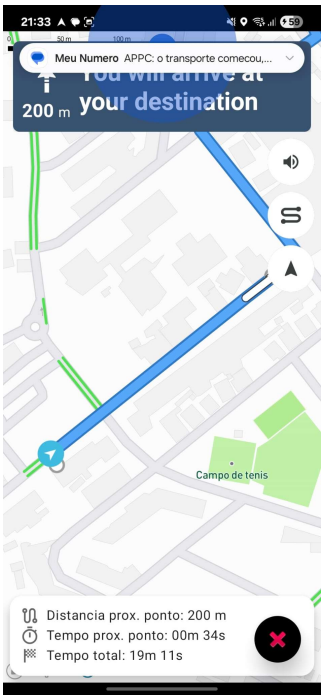


**Figura 39 -** Página de navegação da aplicação

Outra funcionalidade da navegação é o envio do segundo SMS ao utente. Utilizando a funcionalidade do Mapbox que permite identificar qual será a próxima leg da viagem, através do método “onNextRouteLegStart”, é possível determinar qual será o próximo ponto de paragem, bem como obter uma estimativa do tempo necessário para lá chegar. Com base nessas informações, é enviado um SMS ao utente quando este é o próximo a ser recolhido, indicando a estimativa da hora de chegada do veículo.

Combinando este SMS com o enviado inicialmente, é possível manter o utente informado ao longo da viagem, mesmo na ausência de uma aplicação dedicada. No entanto, o sistema foi concebido de forma a permitir, no futuro, o desenvolvimento e integração de uma aplicação específica para os utentes.

Apesar disso, o envio de SMS diretamente a partir da aplicação apresenta algumas vantagens. Como a mensagem é enviada através do telemóvel do motorista e utiliza o seu número, o utente passa a dispor de um meio simples de contacto direto com o condutor. Importa ainda relembrar que muitos dos utentes da APPC têm dificuldades, ou mesmo incapacidade, em utilizar um telemóvel. Por esse motivo, o número de contacto fornecido pode não ser do próprio utente, mas sim de um cuidador, o que facilita igualmente a comunicação entre este e o motorista.

## 5.1. Exemplo de uso da aplicação

Neste subcapítulo será feita uma demonstração do funcionamento da aplicação, utilizando uma rota criada com vários pontos na cidade de Castelo Branco, ilustrada na figura 40.

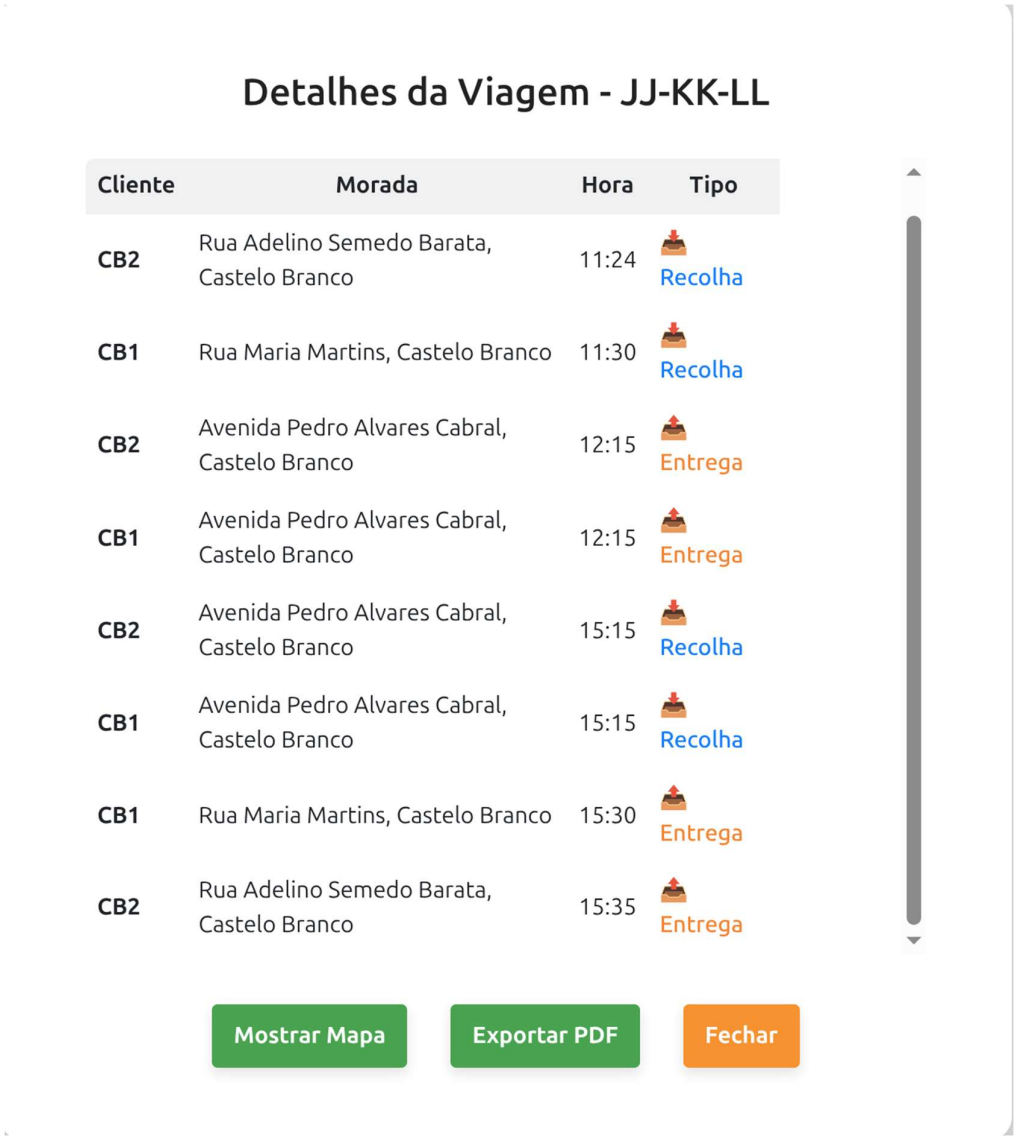


**Figura 40 -** Rota criada para demonstração da aplicação

O primeiro passo consiste em verificar o ID da viagem que se pretende seguir, assumindo que o motorista já efetuou o login. Esse ID é obtido através do PDF gerado no front-end, conforme referido anteriormente.

De seguida, é apresentada a página que lista todos os pontos da viagem, bem como o veículo atribuído para a realizar, conforme se pode observar na figura 41.


**Figura 41 -** Lista dos pontos da rota

A partir deste ecrã, é possível iniciar a navegação através do botão 'Iniciar Navegação'. Para a simulação apresentada neste exemplo, foi utilizada a funcionalidade de simulação de localização do emulador do Android Studio. A localização inicial selecionada foi a dos Bombeiros Voluntários de Castelo Branco, na Rua Comandante Lúcio Pereira, conforme ilustrado na figura 42.

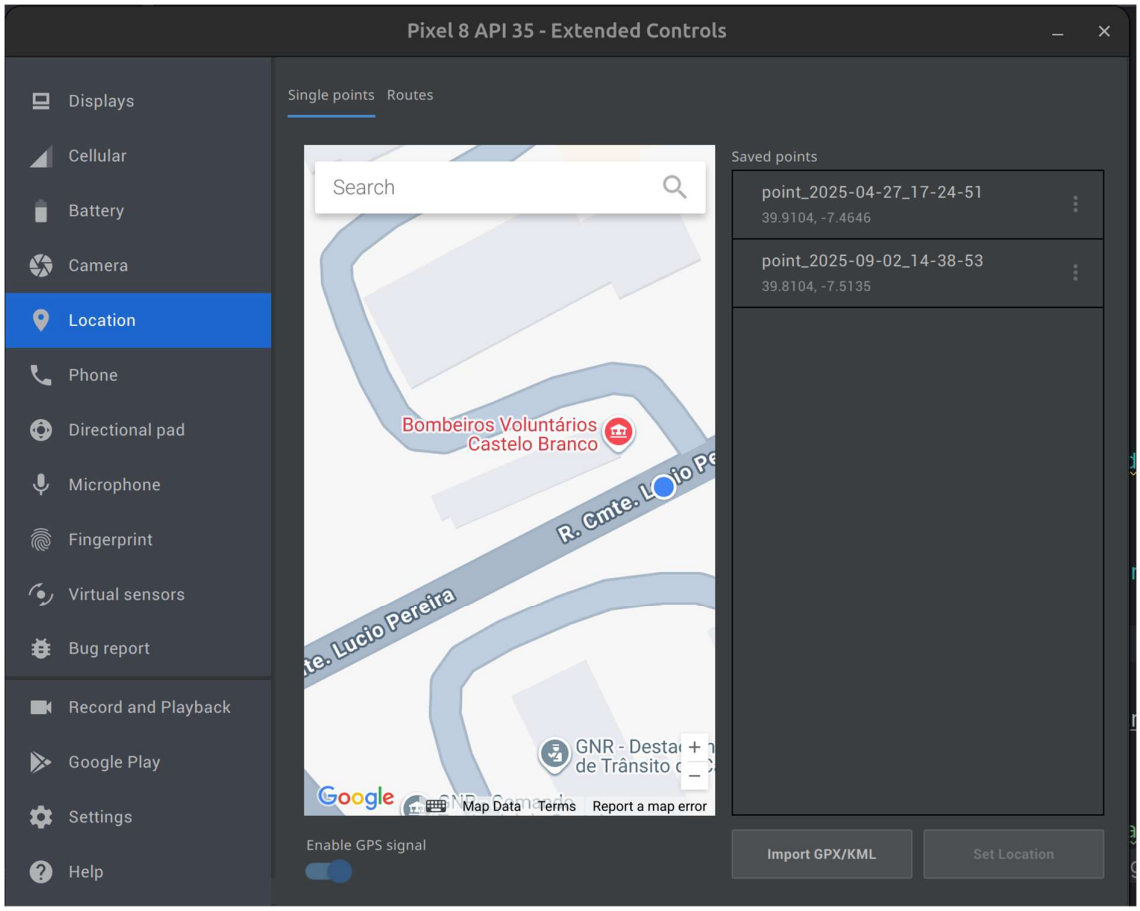


**Figura 42 -** Ecrã de simulação da localização

Quando se inicia a navegação, é enviado um SMS com a informação relativa ao tempo estimado de chegada. No contexto desta demonstração, devido à utilização de um emulador, foi usado o número do próprio emulador como número de telefone do utente. Este facto explica também porque é visível apenas um SMS em vez de dois, uma vez que ambos os utentes em transporte estão associados ao mesmo número.

Para evitar que um mesmo utente receba vários SMS no início da viagem com a estimativa de tempo para cada uma das suas paragens, a aplicação foi programada para enviar apenas uma mensagem por número de telefone. Este SMS pode ser visto na figura 43.

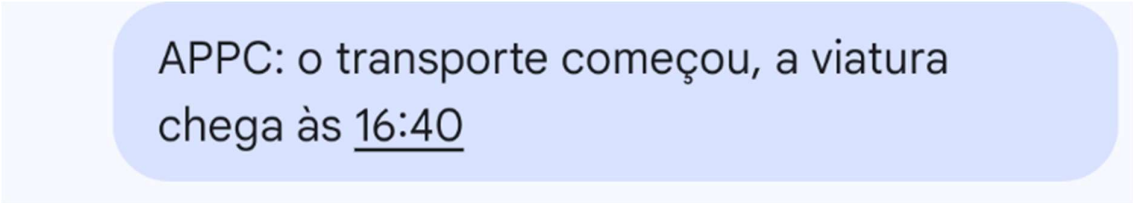


**Figura 43 -** SMS enviado no início da viagem

Pode-se agora observar o ecrã de navegação, dando particular ênfase à barra inferior, onde são apresentadas informações como a distância até ao próximo ponto, o tempo estimado até esse ponto e o tempo total da viagem. Este exemplo encontra-se ilustrado na figura 44.

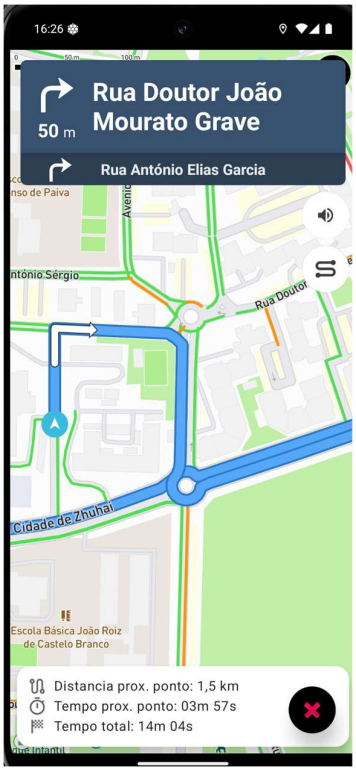


**Figura 44 -** Ecrã da navegação 1

De seguida, recorrendo à funcionalidade "Routes" do emulador do Android, foi inserida a rota entre a posição inicial e a primeira paragem, iniciando-se assim a simulação. Quando a localização do ponto é atingida, surge uma pequena mensagem no ecrã de navegação, conforme ilustrado na figura 45. É também possível observar as alterações nos tempos estimados para os pontos seguintes.

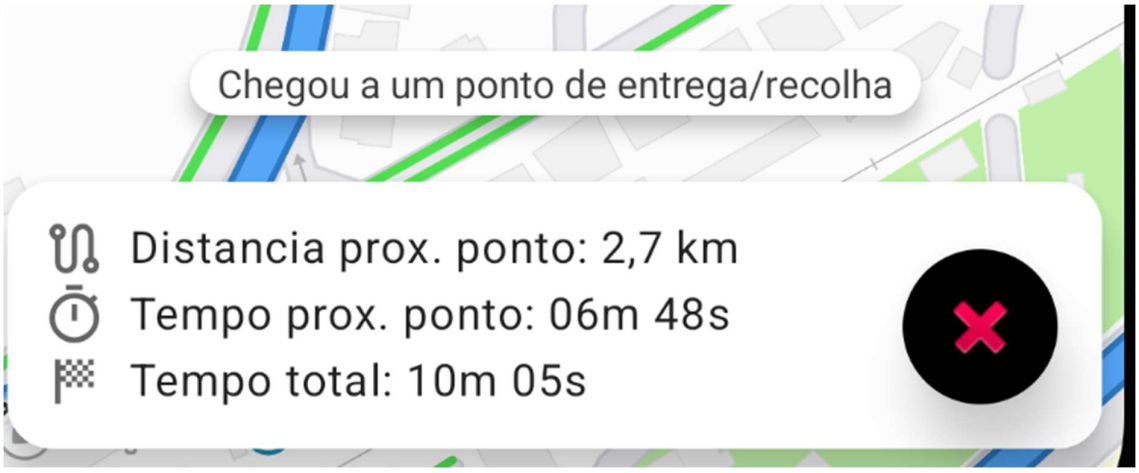


**Figura 45 -** Mensagem de chegada ao ponto

Na chegada a um ponto, é ainda enviado um SMS ao utente da paragem seguinte com a informação relativa à hora prevista de chegada do transporte. Este SMS está ilustrado na figura 46.

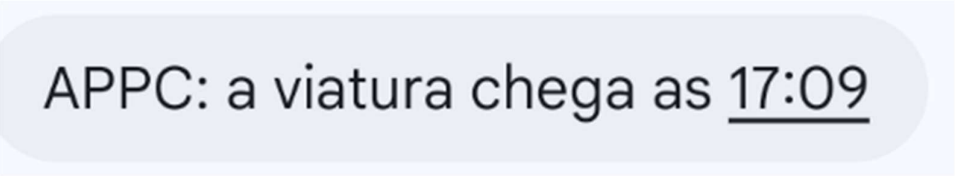


**Figura 46 -** SMS com hora de chega prevista ao próximo ponto

De seguida, será demonstrada a capacidade da aplicação para recalcular a rota em caso de erro ou engano no seu seguimento. A rota inicialmente recomendada pela aplicação pode ser observada na figura 47.

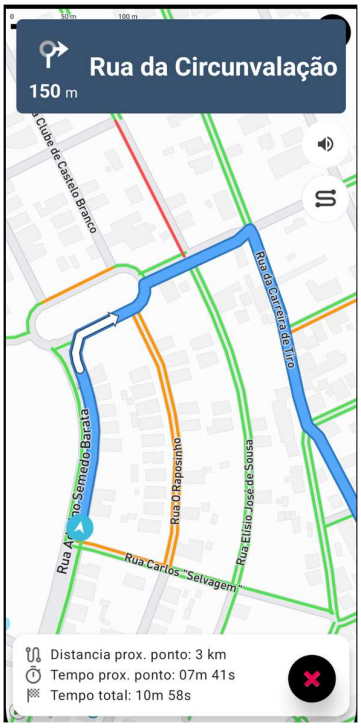


**Figura 47 -** Ecrã da navegação 2

Para demonstrar esta funcionalidade, foi criada a rota apresentada na figura 48.

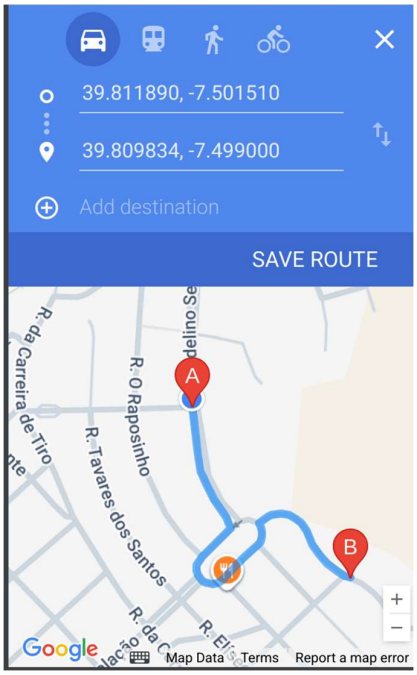


**Figura 48 -** Rota criada para demonstrar a capacidade de recalcular a rota

Após a conclusão desta rota, pode observar-se na figura 49 que a rota foi recalculada e que o tempo e a distância restantes foram atualizados.

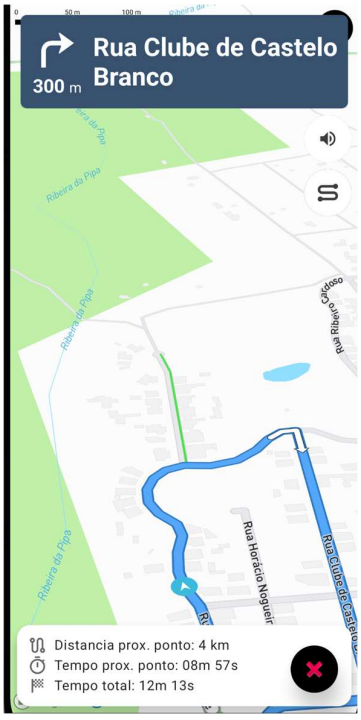


**Figura 49 -** Ecrã de navegação 3

Outra funcionalidade da navegação é a possibilidade de visualizar uma “overview” da rota, ao clicar no botão apresentado na figura 50.

**Figura 50 -** Botão de "overview"

A figura 51 apresenta um exemplo dessa vista geral.


**Figura 51 -** Ecrã de navegação 4

Para concluir este tópico, quando a viagem chega ao ponto final é apresentada ao utilizador uma pequena mensagem, bem como a indicação de chegada ao destino na barra superior. Este comportamento pode ser observado na figura 52.

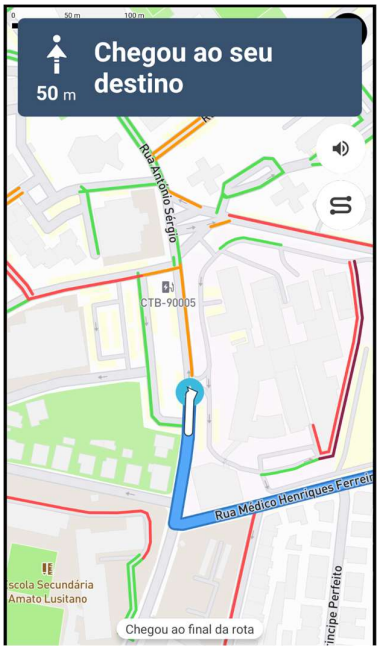


**Figura 52 -** Ecrã com informações de final de rota

## 5.2. Análise do funcionamento da aplicação móvel.

Foi ainda realizada uma análise das funcionalidades da aplicação em diferentes dispositivos de várias gamas, com o objetivo de garantir que a experiência do utilizador se mantinha consistente, independentemente do equipamento utilizado.

Para esse efeito, a aplicação foi testada em dois dispositivos: um mais recente, com um preço entre 400 € e 450 €, e outro mais antigo, com cerca de cinco anos e um preço original entre 150 € e 200 €. Os telemóveis são de marcas diferentes e utilizam versões distintas do Android, sendo que o mais recente utiliza a antepenúltima versão (Android 15), enquanto o mais antigo utiliza uma versão de 2021 (Android 12).

A versão mínima exigida para a utilização da aplicação é o Android 5 (lançado em 2014), permitindo assim o seu funcionamento em dispositivos significativamente antigos. Esta versão mínima foi definida com base nos requisitos do SDK de navegação do Mapbox.

Foi realizado um teste funcional no telemóvel mais antigo, o qual demonstrou um desempenho semelhante ao do dispositivo mais recente. Com base nesses resultados, pode afirmar-se que qualquer smartphone relativamente moderno (com até cinco anos), mesmo que de gama inferior, conseguirá executar a aplicação sem dificuldades.

Além disso, a aplicação ocupa apenas 98,38 MB, o que reduz a probabilidade de causar problemas em equipamentos com menor capacidade de armazenamento.

Foram ainda realizados alguns testes às funcionalidades de recalculamento de rotas, previsão de chegada e informações de trânsito.

Os testes permitiram concluir que a previsão de chegada é relativamente precisa, embora seja importante referir que a área onde os testes foram realizados não apresenta grandes níveis de trânsito.

No que diz respeito às informações de trânsito, esta foi a funcionalidade com resultados menos satisfatórios: em determinados momentos, o mapa indicava uma grande quantidade de trânsito em ruas que, na realidade, apresentavam um volume de carros normal ou até inferior ao habitual.

Por fim, o recalculamento da rota revelou-se funcional e eficaz, embora um pouco "apressado". A rota é recalculada assim que a localização do dispositivo deixa de coincidir com o trajeto originalmente definido, o que pode provocar o recalculamento mesmo em situações em que há apenas um pequeno erro de localização. No entanto, trata-se de um problema raro, mais comum em ambientes urbanos e quando se circula a velocidades mais elevadas do que o recomendado.

# 6. Conclusão

Neste tópico é feito um resumo do trabalho realizado, uma análise crítica do mesmo, e por fim discutido o possível trabalho futuro.

## 6.1. Resumo do trabalho realizado em projeto II

Durante o segundo semestre, o projeto evoluiu significativamente com base no envolvimento direto da associação interessada, o que levou a uma redefinição de alguns requisitos.

Foi realizada a reformulação da base de dados, com atualização do modelo entidade-relacionamento, para suportar novas funcionalidades como agendamentos recorrentes e diferentes tipos de utilizadores. Foram também feitas várias alterações no Front-end, tanto na melhoria de componentes existentes como na criação de novas páginas, como as de viagens e agendamentos, desenvolvimento das funcionalidades de agendamentos e viagens na API, para garantir assim a comunicação correta entre base de dados, interface web e aplicação móvel.

Foi também criada uma aplicação móvel para os condutores, programada com a tecnologia Kotlin e o SDK da Mapbox, com funcionalidades como login, visualização de viagens e navegação ponto a ponto e a implementação do sistema de envio de mensagens SMS para garantir que os clientes, mesmo sem acesso facilitado a tecnologias, fossem informados sobre as suas viagens.

Para terminar, foram ainda realizados testes ao Front-end, à criação de rotas e à aplicação móvel, com o objetivo de garantir o correto funcionamento de todas as funcionalidades, bem como identificar eventuais problemas.

## 6.2. Análise crítica do trabalho realizado

Este segundo semestre foi uma fase muito importante no desenvolvimento do projeto. O foco principal foi continuar o que já tínhamos feito e melhorar tudo com base no feedback da associação que demonstrou interesse na solução. O envolvimento desta entidade foi fundamental, pois ajudou-nos a perceber o que realmente era necessário para que o sistema fosse útil no dia a dia.

Apesar de termos feito um trabalho funcional no primeiro semestre, foi ao testar e tentar usar o sistema em situações reais que começaram a surgir alguns problemas que não tínhamos previsto. Um exemplo disso foi um erro que ocorria quando a API estava ligada há mais de 8 horas onde a ligação da base de dados, que era criada apenas quando a aplicação era executada, dava “timeout” e a aplicação deixava de funcionar, isto foi resolvido verificando a ligação à base de dados antes de se fazer qualquer pedido à mesma.

Outro desafio importante foi o desenvolvimento da aplicação do condutor. Esta revelou-se uma tarefa particularmente exigente, não só pela necessidade de aprendizagem de novas tecnologias (como Kotlin e o SDK de navegação da Mapbox), mas também pela complexidade funcional envolvida desde a autenticação e gestão de sessões até ao envio de notificações SMS e navegação ponto a ponto. Mesmo

com limitações de tempo e recursos, a aplicação alcançou um nível funcional adequado e integrou-se com sucesso no sistema completo. Também tivemos de encontrar uma solução prática para manter os utentes informados das suas viagens, visto que muitos deles não usam ou têm dificuldade em usar aplicações em smartphones. A solução encontrada — envio de mensagens SMS com informação relevante — revelou-se simples, eficaz e adaptada à realidade dos utentes da instituição.

Em resumo, o projeto evoluiu bastante, passando de um protótipo inicial para uma solução mais completa e mais próxima daquilo que será usado no mundo real. Conseguimos atingir os principais objetivos que nos tínhamos proposto e deixamos tudo preparado para que, no futuro, o sistema possa ser melhorado e expandido com novas funcionalidades.

## 6.3. Trabalho futuro

Apesar do sucesso do projeto, algumas funcionalidades poderão ser implementadas no futuro, além de adicionar funcionalidades relacionadas a cuidadores cujo tabelas já foram deixadas criadas na base dados, outras funcionalidades foram pensadas, mas não desenvolvidas.

Uma dessas funcionalidades será a melhoria da gestão de condutores, com a adição de um calendário de férias possibilitando assim o sistema saber quantos condutores estão disponíveis para fazer viagens podendo assim limitar o número de veículos com base nessas informações.

Já nos veículos, há também uma ideia parecida, mas relacionada com a manutenção, dar a habilidade ao administrador de marcar veículos como avariado ou dizer que por exemplo esta semana o veículo não vai estar disponível por qualquer motivo.

Devido à maneira como o projeto foi desenvolvido, dividido em várias partes, adicionar novas funcionalidades é relativamente mais fácil, melhorando assim também a vida útil do projeto.

# 7. Referências

[1] “Início - APPC.” Accessed: Jun. 14, 2025. [Online]. Available: https://www.appc.pt/

[2] “Free Online Gantt Chart Software.” Accessed: Jan. 23, 2025. [Online]. Available: https://www.onlinegantt.com/#/gantt

[3] “Optimization API v2 | API Docs | Mapbox.” Accessed: Jun. 12, 2025. [Online]. Available: https://docs.mapbox.com/api/navigation/optimization/

[4] “Optimization API v1 | API Docs | Mapbox.” Accessed: Sep. 12, 2025. [Online]. Available: https://docs.mapbox.com/api/navigation/optimization-v1/

[5] “mapbox/mapbox-navigation-android-examples at main-v2.” Accessed: Jun. 14, 2025. [Online]. Available: https://github.com/mapbox/mapbox-navigation-android-examples/tree/main-v2