Politécnico
Castelo Branco
Escola Superior
de Tecnologia

Sistema para otimização dinâmica de transporte de utentes com necessidades especiais
Projeto I

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 Santo, do Instituto Politécnico de Castelo Branco.

Janeiro 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.

O nosso projeto busca responder a essa necessidade através do desenvolvimento de uma aplicação web para a gestão interna dos serviços de transporte de pessoas com limitações de mobilidade. A aplicação foi desenhada para ser intuitiva e funcional, permitindo às associações gerir eficientemente motoristas, veículos, agendamentos, paragens e as necessidades específicas dos utentes. Com base em um modelo de dados cuidadosamente elaborado, o sistema integra múltiplas tabelas relacionadas, como Motorista, Veículo, Utente, Agenda, e Caraterísticas, para garantir que as informações sejam organizadas de forma lógica e acessível.

A aplicação também inclui módulos para monitorizar o estado de veículos e motoristas, garantindo que 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.

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.

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*

1. Introdução

Esta secção aborda o problema apresentado, os objetivos, a planificação do trabalho e a organização do documento

1.1. Contexto

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

Neste contexto, seria útil uma solução tecnológica inovadora para automatizar a gestão interna dos serviços de transporte de pessoas com limitações de mobilidade.

Para tal é necessária a criação de rotas eficientes no transporte de utentes. Esta situação ocorre quando uma entidade de transportes tem de levar múltiplos utentes a vários locais de tratamento, apesar de ser possível a criação de uma rota manualmente, a partir do momento que se começa a adicionar vários utentes com destinos diferentes e que têm de ser apanhados em locais diferentes torna este processo mais difícil.

Por isso foi proposta a criação de um sistema que possa criar estas rotas de maneira eficiente e dinâmica. Para isso o sistema tem de ser capaz de identificar a melhor rota com base nos utentes que necessitem de transporte desenhando a melhor rota para os apanhar e deixar, nem todos os utentes vão para o mesmo destino e existe normalmente mais do que um veículo para fazer a recolha, tudo isto também tem de ser considerado.

Além disso, alguns utentes têm necessidades especiais a nível de transporte nomeadamente, a necessidade de maca ou equipamento elétrico (exemplo respirador artificial), sendo assim, o programa também tem de ser capaz de identificar se o veículo é capaz de transportar tal utente.

1.2. Âmbito e definição do problema

O objetivo fundamental deste projeto é criar uma solução informática para otimizar a gestão diária do transporte dos utentes de Associações de ajuda a pessoas com necessidades especiais.

A solução deste projeto consiste em desenvolver um sistema de gestão de utentes com necessidades especiais, isto é, um sistema que seja capaz de implementar uma rota eficiente entre o utente e onde este necessite de transporte (por exemplo hospital), o objetivo é que esta rota seja otimizada para levar mais que um utente. Os utentes que se possam encontrar numa "viagem" não necessitam de ir todos para o mesmo sítio, cabe ao sistema averiguar a melhor rota para entregar todos os utentes ao seu destino.

Alem disso o sistema também tem de ter em mente que diferentes utentes têm diferentes necessidades, uns podem ir sentados, outros podem ter de ir de maca, outros talvez precisam de máquinas especializadas que necessitem de ligação elétrica por exemplo. Com base nisso o sistema tem de ser capaz escolher o veículo correto para transportar os vários utentes.

Para atingir este objetivo o sistema é dividido em várias partes sendo a base deste, a base de dados, aqui serão guardadas todas as informações incluindo informações sobre os utentes e condutores, possíveis destinos, os veículos e seus equipamentos, agendamentos e viagens.

Para fazer a ligação entre a base de dados e as aplicações que iremos falar a seguir foi desenvolvida uma API, esta permite fazer as operações CRUD assim como operações de autenticação e outras que sejam necessárias para a implementação das rotas.

Foi desenvolvido também um *dashboard* de administração web onde um administrador pode manualmente fazer criação de utentes, veículos, agendamentos entre outros tudo através da API referida anteriormente.

Para projeto II serão desenvolvidas duas aplicações móveis uma para o motorista outra para o utente. A aplicação do motorista tem como objetivo servir de sistema de navegação para indicar as próximas paragens e possíveis desvios devido a trânsito por exemplo. A aplicação do utente serve o propósito de fazer marcações e indicar o horário que os utentes tenham de estar preparados para serem transportados. Será ainda desenvolvido o código necessário para a criação das rotas de maneira dinâmica e eficiente.

1.3. Objetivos

A solução tem como objetivo minimizar os custos, o tempo, os recursos materiais e humanos despendidos na gestão dos transportes, bem como evitar incompatibilidades, atrasos e falhas no serviço. O processo envolverá a afetação de veículos em função das suas caraterísticas e das necessidades específicas dos utentes, bem como a coordenação entre vários veículos e percursos. Idealmente, esta solução proporcionaria serviços mais eficientes, rápidos e fiáveis para pessoas com limitações de mobilidade, reduzindo os tempos de espera e de viagem, melhorando o conforto e a qualidade do tempo despendido nas deslocações.

Para atingir este objetivo principal, estes são os objetivos iniciais:

- Levantamento do estado da arte e análise tecnológica
- Modelação da Aplicação
- Capacitação em Desenvolvimento Front-End
- Desenvolvimento do Front-End
- Desenvolvimento do Back-End

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

Etapa 1 – Focada na investigação e compreensão das necessidades do projeto. O objetivo foi identificar as melhores práticas e tecnologias aplicáveis ao desenvolvimento do sistema.

Etapa 2 – Consistiu na criação do Diagrama Entidade-Relacionamento (ER) para estruturar a base de dados.

Etapa 3 – Foi realizado um estudo intensivo sobre o framework React[1], com foco no desenvolvimento de interfaces modernas e responsivas.

Etapa 4 – Implementação do sistema de interface gráfica. A acessibilidade e usabilidade também foram priorizadas.

Etapa 5 – Criação da camada de servidor responsável pelo processamento de dados e fornecimento de APIs RESTful.

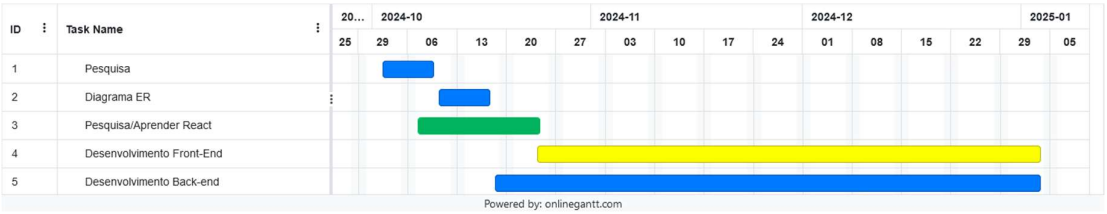


Figura 1 - Diagrama de Gantt[2]

1.4. Organização do documento

Este relatório está organizado em capítulos para facilitar a compreensão do projeto desenvolvido. No primeiro capítulo, é apresentado o contexto do projeto, os objetivos gerais e específicos, o âmbito e a definição do problema. Esta secção inclui também uma breve descrição dos objetivos principais que orientaram o trabalho.

No segundo capítulo, é explicada a arquitetura da solução, detalhando os diferentes componentes do sistema. Esta parte aborda a estrutura da base de dados, a implementação da API e o desenvolvimento do front-end, descrevendo como cada um desses elementos contribui para o funcionamento do projeto.

O terceiro capítulo trata da análise realizada para a escolha do serviço de mapas a ser utilizado. Aqui são explicados os critérios considerados, como custo, precisão, facilidade de uso e integração com o sistema, além de justificar a seleção do serviço escolhido.

No quarto capítulo, é apresentada a conclusão, que resume o trabalho desenvolvido e os resultados alcançados. Este capítulo destaca os principais pontos do projeto e propõe os próximos passos para a continuidade do trabalho na próxima fase.

Por fim, o relatório inclui as referências bibliográficas utilizadas e, nos anexos, são disponibilizadas informações adicionais, como a tabela de endpoints da API e outros dados relevantes para complementar a documentação do projeto.

2. Arquitetura da solução

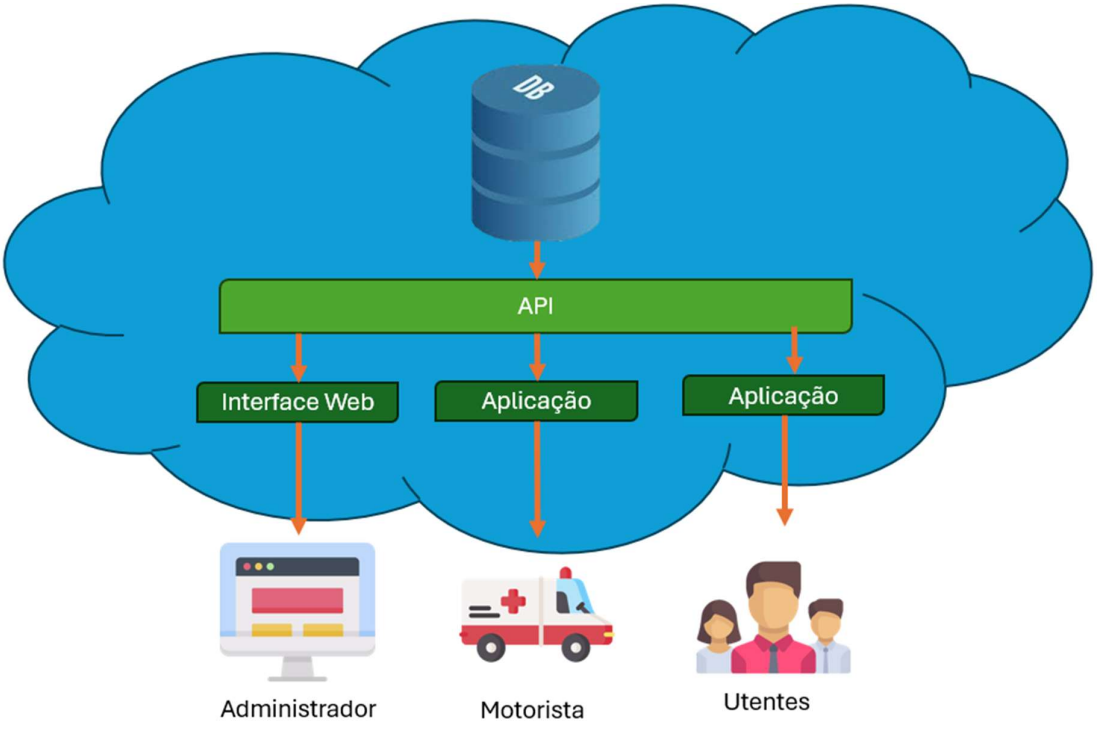


Figura 2 - Diagrama arquitetura da solução

O sistema é composto por diversos componentes, nesta primeira fase focamos-mos no desenvolvimento da base de dados, que armazena a informação, a API que faz a ligação entre a bd e o resto dos serviços como a interface web e as aplicações móveis que vão ser desenvolvidas futuramente, e a interface web que e voltada para os administradores permitindo fazer o gerenciamento dos diversos dados.

Existem três tipos de utilizadores, o administrador que como referido anteriormente faz a gestão do sistema, o motorista que terá acesso a uma aplicação onde irá aparecer a sua rota e as paragens que terá de fazer para levantar ou deixar utentes, e os utentes que também terão uma aplicação que permite a marcação de deslocações. Ambas as aplicações serão apenas desenvolvidas em projeto II.

Este diagrama pode vir a ser alterado quando for implementado a otimização de rotas, mas isso também será apenas desenvolvido em projeto II.

Neste capítulo também cera discutido a escolha do serviço de mapas, que será implementado no futuro.

2.1. Componentes da solução

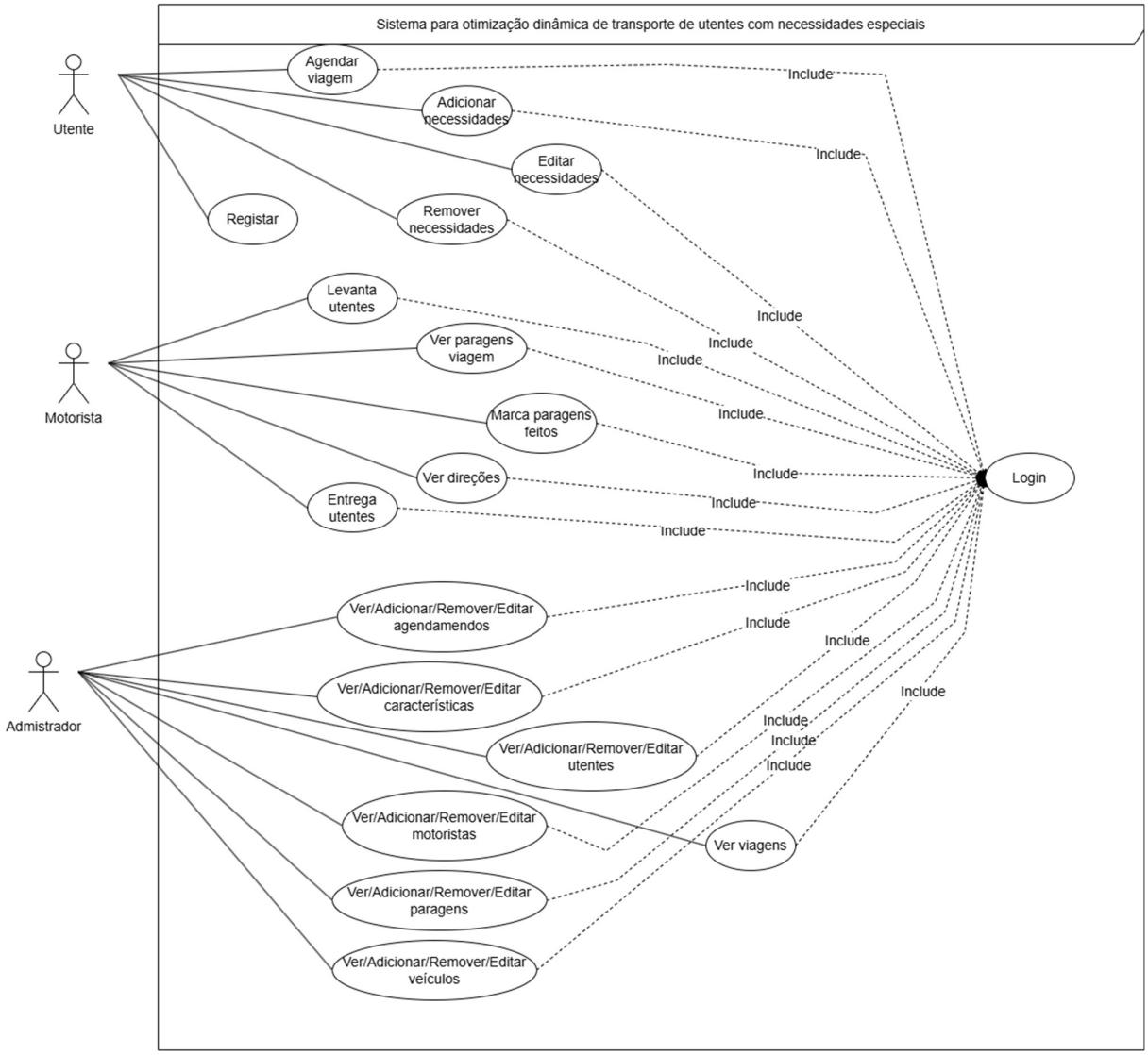


Figura 3 - Diagrama de caso de usos

Este diagrama foi utilizado como base para decidir que componentes seriam necessários no sistema. Com este diagrama apontamos 3 atores (bonecos na figura 2) utentes, motoristas e administradores e as funcionalidades que estes podem aceder. Em projeto 1 foi mais focado no desenvolvimento das funções de administrador, também é possível que este diagrama possa vir a ser atualizado quando outras partes dos sistemas forem desenvolvidas.

Os próximos subcapítulos falam mais em detalhe sobre cada componente da implementação.

2.1.1. Base de dados

A base de dados é o centro da aplicação, onde todas as informações são guardadas e distribuídas com a ajuda da API pelo resto das aplicações sejam elas Web ou móveis.

A base de dados é construída utilizando uma abordagem relacional, com tabelas interligadas por chaves primárias e estrangeiras para garantir a integridade dos dados. Cada entidade representa um elemento essencial do sistema, como Motorista, Veículo, Utente, Viagem e Agenda. Relações são estabelecidas para modelar interações como as características de veículos, as necessidades dos utentes e as paragens realizadas em viagens. As variáveis “Localizacao”, “Levantado” e “Feita” espalhados pelas várias tabelas estão previamente colocadas ainda sem qualquer utilização (exceto na tabela Paragem) pois vão servir para a gestão de rotas com a implementação de mapas que vão ser falados posteriormente.

O servidor da base de dados escolhido foi o MySQL[3] Server, apesar de inicialmente termos optado pelo Microsoft SQL Server[4], acabamos por escolher o MySQL pois este era mais fácil de implementar. Neste momento o Servidor de base de dados encontra-se hospedado individualmente no computador de cada elemento do projeto. Isto será algo que terá de ser resolvido em projeto II devido às aplicações móveis.

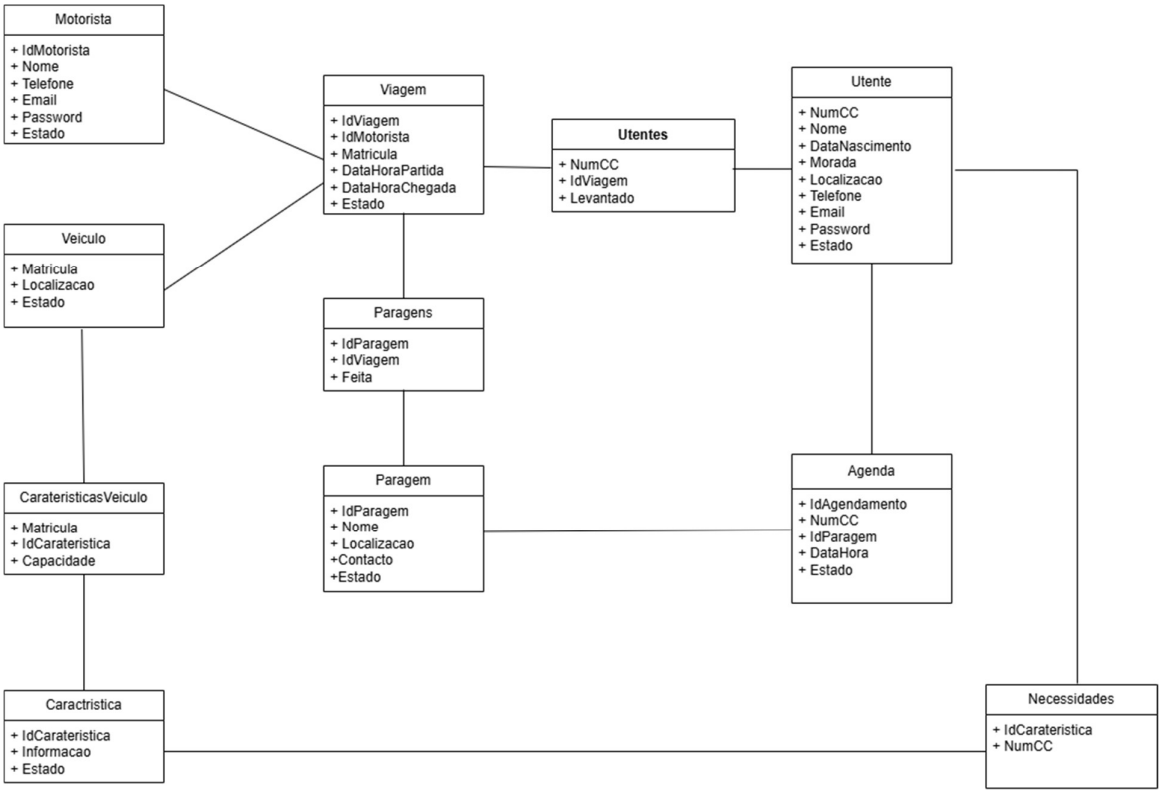


Figura 4 - Diagrama de classes

2.1.1.1. Tabela Motorista

Na tabela Motorista, o motorista contém um identificador único (IdMotorista), o nome, telefone e email do mesmo, além de possuir uma palavra-passe (Password) que essa será encriptada pelo back-end para garantir máxima segurança. Irá existir também um campo denominado por “Estado” que servirá para identificar se esse mesmo motorista está apto para realizar deslocações, ou seja, é útil para ativar ou desativar motoristas na aplicação.

2.1.1.2. Tabela Viagem

Na tabela Viagem, existe um identificador único para cada viagem (IdViagem) e, está associada a um motorista (IdMotorista) e ao veículo que ele conduziu, representado pela matrícula (Matrícula). A tabela também armazena informações sobre a data e hora de partida (DataHoraPartida) e chegada (DataHoraChegada), além do campo “Estado” que informa se a viagem foi concluída ou está pendente.

Existe também a tabela Utentes, que faz a associação entre as tabelas Utente e Viagem (NumCC, IdViagem), existindo também um campo “Levantado” que servirá para mostrar que o utente já está a realizar a deslocação ou ainda está à espera.

2.1.1.3. Tabela Utente

Na tabela Utente, são registados todos os passageiros do sistema. Cada utente possui um identificador único, no caso o número do cartão de cidadão (NumCC), nome, data de nascimento, localização, telefone, email e palavra-passe (Password) que da mesma maneira como é processada na tabela Motorista é igualmente na tabela Utente para garantir segurança. Já no campo “Estado” é essencial caso o utilizador deixe de ser ativo no sistema.

Além disso a tabela Necessidades é mais uma tabela intermediária, porém esta tem o objetivo de coligar a tabela Utente (NumCC) e tabela Caraterística (IdCarateristica) com o objetivo de saber as necessidades de cada utente para as devidas deslocações.

2.1.1.4. Tabela Paragem

A tabela Paragem contém informações mais pormenorizadas sobre os locais de paragem disponíveis, isto é, cada paragem possui um identificador único (IdParagem), nome, localização e contacto. O campo “Estado” informa se a paragem está disponível ou não para ser utilizada nos agendamentos.

Já na tabela Paragens é uma tabela para fazer a ligação de muitos para muitos, ou seja, a viagem pode conter várias paragens.

2.1.1.5. Tabela Veículo

A tabela Veículo armazena informações sobre os veículos existentes para deslocações. Cada veículo é identificado pela matrícula (Matrícula), e os outros campos (Localizacao e Estado), vão permitir na aplicação identificar onde o veículo está e se está ativo ou inativo no momento.

A tabela CaraterísticasVeículo é composta por um campo (Capacidade) e este é responsável por colocar a capacidade de cada característica num veículo, isto é, existe o campo "Informacao" alocado na tabela Característica que determina os extras que cada veículo tem e este campo "Capacidade" vem a determinar a quantidade que existe em cada extra, na **Figura 10**, o veículo "00-43-SE" está descrito que contém "Maca" que vem do campo "Informacao" e o que vem de seguida "(5)" é onde o campo "Capacidade" atua dizendo que este veículo contém 5 macas.

2.1.1.6. 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 utilizador registar na aplicação. 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).

2.1.1.7. Tabela Caraterística

Na tabela Caraterística possuímos o seu identificador único (IdCarateristica), um campo de informação (Informacao) e este, é essencial para armazenar o que cada veículo pode conter, por exemplo maca, bomba de oxigénio, etc…E um campo estado para ativar/desativar essas várias caraterísticas dependente do veículo.

2.1.2. API

A API foi desenvolvida para fazer uma ligação segura entre a bd e o resto dos componentes, como as aplicações moveis e o front-end. Esta foi desenvolvida em Node.Js[5] com a ajuda da *framework* Express[6]. O Node.Js foi escolhido devido a sua flexibilidade para criar API RESTful e permitindo assim também que o front-end e back-end partilhem a mesma linguagem (JavaScript). A framework Express foi utilizada para a facilitar e acelerar o desenvolvimento da API.

A API permite a um administrador do sistema fazer todas as operações CRUD das tabelas da base de dados, permite ainda o registo de utilizadores e motoristas.

Para fazer a ligação até à base de dados foi utilizada a biblioteca mysql2[7] que, permite a ligação ao servidor e a execução de códigos SQL, permite também a execução de códigos de controlo de transição como "*beginTransaction*" e "*rollback*" para prevenir que informações sejam guardas na bd se acontecer algum tipo de erro. O código também foi desenvolvido de maneira a que as informações da base de dados como o *host* ou *password* possam ser facilmente alteradas utilizando um ficheiro ".env".

A API pode ser dividida em diversos "endpoints" sendo estes os URI utilizados para a visualização ou alteração dos dados.

Todos estes caminhos começam com o URL do servidor por exemplo "localhost:5000" seguido de "/api/v1/", o resto é preenchido conforme a operação que se queira executar, por exemplo para se poder receber todos os utentes pode-se fazer um pedido GET para "localhost:5000/api/v1/utentes", quando é feito este pedido ou qualquer outro para receber todos os dados, apenas são enviados valores que se encontram ativos, ou seja, é verificado o campo "estado" referido anteriormente no capítulo da base de dados. As informações recebidas e enviadas são em formato JSON, em baixo encontra-se o exemplo de um pedido de um utente.


```
[
  {
    "NumCC": 123456789,
    "Nome": "Tiago Teodoro",
    "DataNascimento": "2004-04-19T23:00:00.000Z",
    "Morada": "Rua b",
    "Localizacao": "Aqui",
    "Telefone": "+351147258369",
    "Email": "tiago@email.com",
    "Necessidades": [
      {
        "IdCarateristica": 1,
        "Informacao": "Maca"
      },
      {
        "IdCarateristica": 2,
        "Informacao": "Assento"
      }
    ]
  }
]
```


Figura 5 - Exemplo de resposta da API

A tabela abaixo é um exemplo de alguns dos endpoints disponíveis na API, neste caso do motorista. Para cada tabela principal da BD (as que são usadas para fazer ligação de muitos para muitos não contam) existem pelo menos 5 chamadas possíveis sendo 3 destas para alteração de uma entidade já existente, a escolha da ação depende do método HTTP usado, e diferentes métodos requerem parâmetros diferentes, por exemplo, um GET apenas necessita do id do motorista enquanto um PUT precisa de um objeto JSON com as informações do motorista. Para a criação de um novo motorista, ou para receber uma lista de todos os motoristas não é necessário o id. A tabela completa de endpoints pode ser encontrada nos anexos.

Tabela 1 - Tabela exemplo endpoints

Endpoint	Método HTTP	Descrição	Parâmetros	Códigos de Resposta
/api/v1/motoristas	GET	Receber todos os motoristas	Nenhum	200: Success 404: Não existe motoristas 400: Error
/api/v1/motoristas	POST	Criar motorista	motorista	201: Criado 400: Erro
/api/v1/motoristas/{id}	GET	Recebe motorista pelo id	id	200: Success 404: Não existe motorista 400: Error
/api/v1/motoristas/{id}	PUT	Editar motorista	id	201: Criado 404: Não existem motorista 400: Erro
/api/v1/motoristas/{id}	DELETE	Eliminar motorista pelo id	id	200: Success 404: Não existe motorista 400: Error

Para ajudar a testar a API e também para poder ser utilizado posteriormente como documentação, foi implementado o Swagger-UI[8], esta ferramenta permite de maneira gráfica executar as funções descritas. A seguir encontra-se uma foto com parte desta interface.

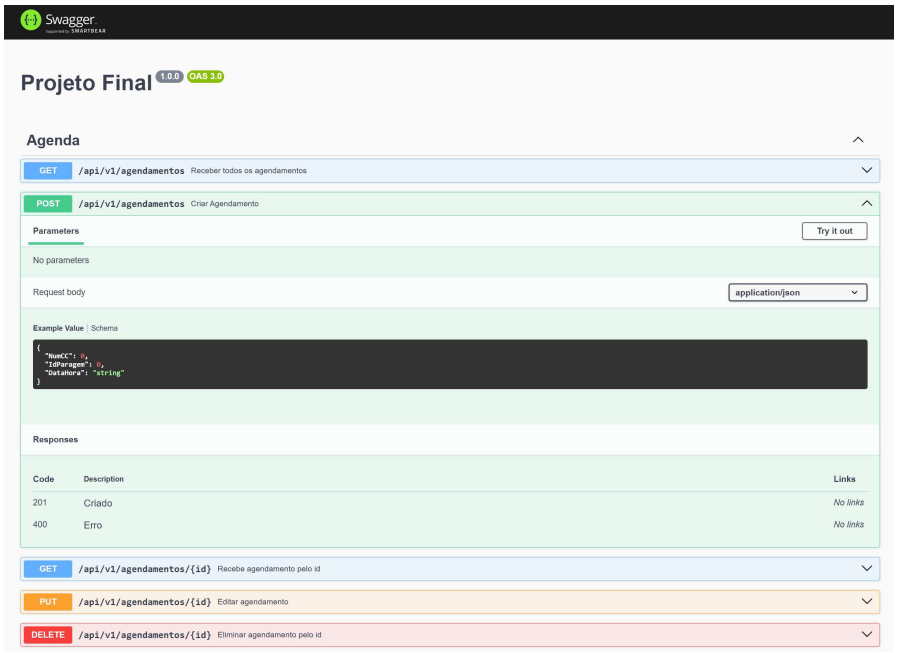


Figura 6 - Swagger-UI

Na figura pode-se observar os diversos “endpoints” e o que cada faz também e possível ver os métodos HTTP que é necessário utilizar para executar cada ação. E possível também ver como podemos fazer um pedido a API, neste caso a criação de um agendamento.

Para a implementação de segurança foram utilizados JSON Web Tokens (JWT). Esta implementação baseia-se na criação e validação de Tokens que são inicialmente criados pela API quando o utilizador faz login, utilizando o seu email e password, que serão pesquisados na base de dados, e guardados como cookies no browser, posteriormente a todos os pedidos feitos a API será enviado esse token, que após ser validado, as informações são enviadas. No caso do administrador, por enquanto a palavra-passe e única para todos e guardada no ficheiro de variáveis.

2.1.3. Front-end

O front-end é onde o administrador consegue configurar todos os dados existentes para de certa maneira, conseguir controlar e se certificar de que está tudo organizado e a ser executado da forma correta.

Optamos por utilizar JavaScript como linguagem principal no desenvolvimento do front-end devido à sua capacidade de criar interfaces dinâmicas e interativas. Além disso, JavaScript possui ótimas ferramentas para aplicações modernas como é o caso do nosso projeto. Para maximizar a linguagem JavaScript, adotamos o React[1], uma biblioteca JavaScript focada em criar interfaces para o utilizador eficientes e reutilizáveis. Com o React, é possível otimizar o desenvolvimento e a manutenção do nosso projeto. A abordagem baseada em um DOM virtual também melhora o desempenho, tornando-o ideal para a criação de interfaces responsivas e escaláveis.

Ao iniciar a aplicação a primeira interação é com uma password que até então falta aprimorar, contudo já é ideia definida.

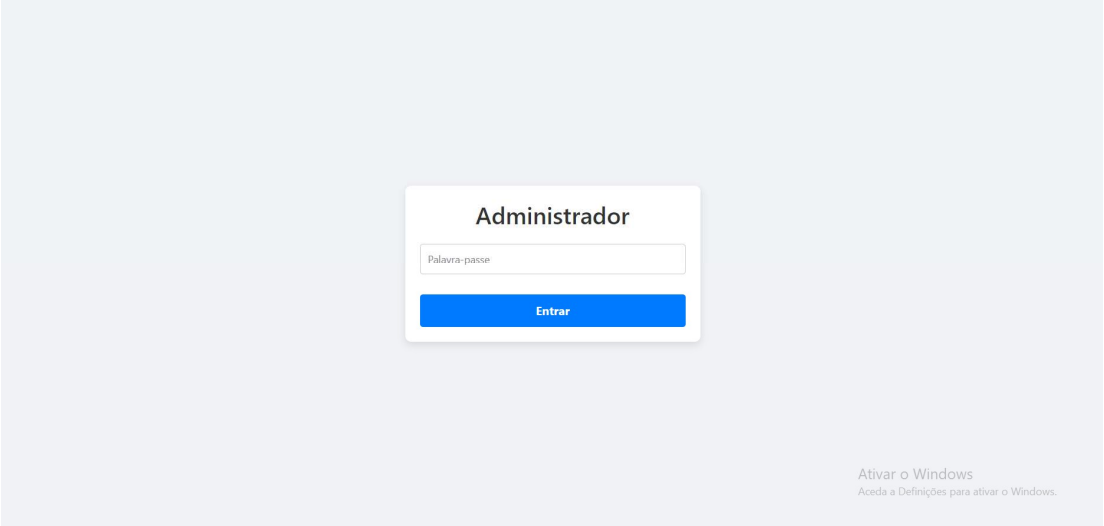


Figura 7 - Sistema de login

Após a inserção da palavra-passe por parte do administrador a página inicial será essa em que será possível ter acesso a tudo:

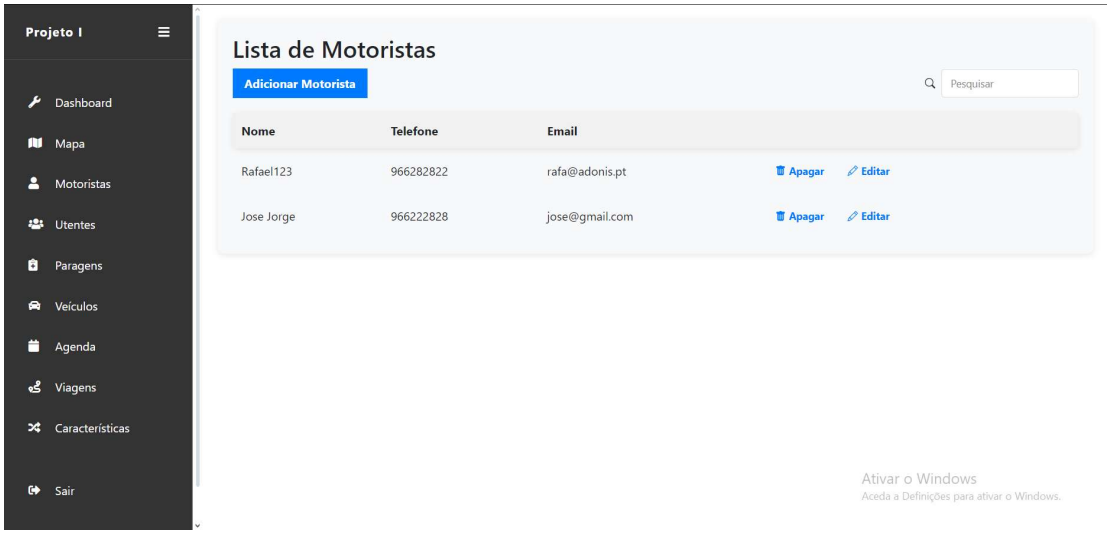


Figura 8 - Página Inicial

Como o nosso propósito era ter uma interface web minimalista, mas ao mesmo tempo fácil de interagir e de fácil aprendizagem sem nenhuma indicação optamos primeiramente por uma *sidebar*. Isto porque é algo mais fácil de visualizar e ao mesmo tempo usufruir das funcionalidades dela. Aqui apenas foram feitas poucas mudanças nomeadamente os ícones e os nomes pois é uma *template*. Está constituída com um ícone que redireciona para cada tabela além de constituir uma para *dashboard* que no projeto II prevemos adicionar diversas estatísticas relativamente à aplicação além de possuir um botão para sair da conta.

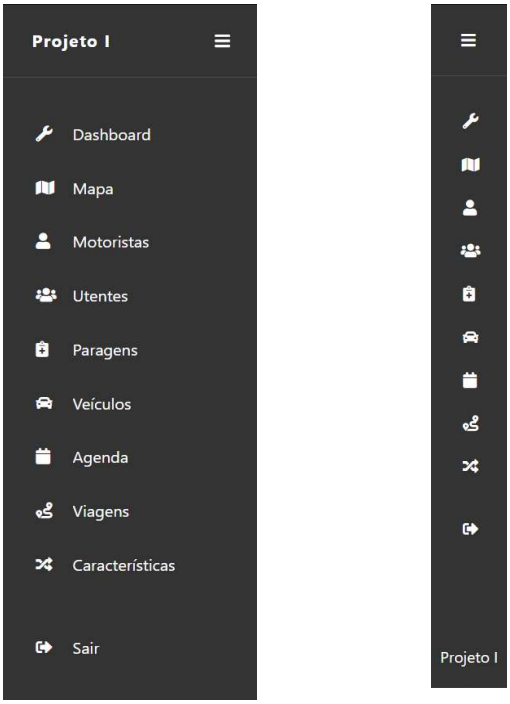


Figura 9 - Sidebar

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.

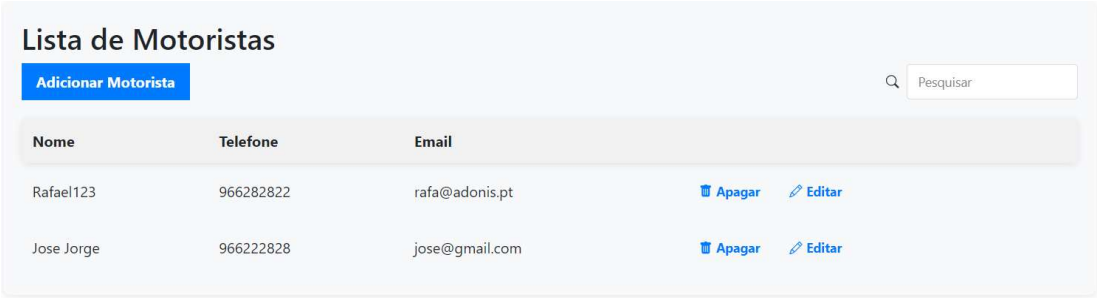
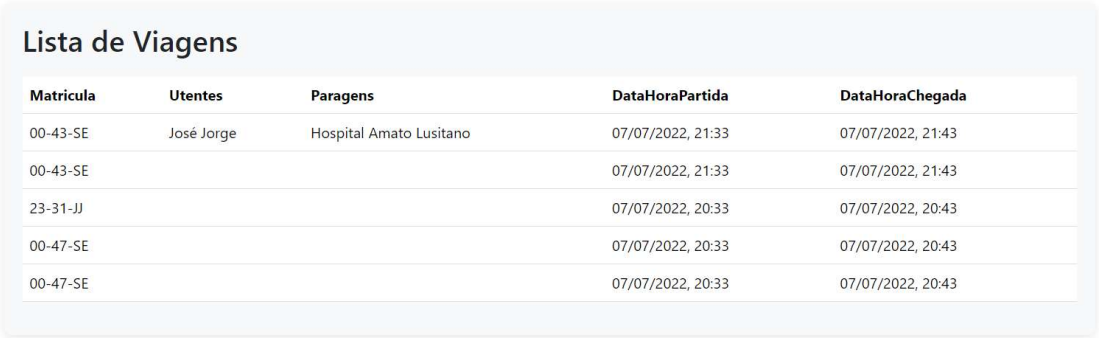


Figura 10 - Lista de Motoristas



Lista de Viagens

Matricula	Utentes	Paragens	DataHoraPartida	DataHoraChegada
00-43-SE	José Jorge	Hospital Amato Lusitano	07/07/2022, 21:33	07/07/2022, 21:43
00-43-SE			07/07/2022, 21:33	07/07/2022, 21:43
23-31-JJ			07/07/2022, 20:33	07/07/2022, 20:43
00-47-SE			07/07/2022, 20:33	07/07/2022, 20:43
00-47-SE			07/07/2022, 20:33	07/07/2022, 20:43

Figura 11 - Lista de Viagens

Lista de Utentes

Adicionar Utente

Pesquisar

CC	Nome	D.Nascimento	Morada	Telefone	Email	Necessidades	
56541654	Rafael	11/01/2003	Rua Sport Benfica e Castelo Branco	911222111	rafael@gmail.com	Maca, Cama, Oxigénio	Apagar Editar
245251966	Jose	09/12/2000	Rua Álvaro Cabral	923456789	joseee@gmail.com	Maca	Apagar Editar

Figura 12 - Lista de Utentes

Lista de Paragens

Adicionar Paragem

Pesquisar

Nome	Localizacao	Contacto		
Hospital Amato Lusitano	Avenida Pedro Alvares Cabral 3, 6000-085 Castelo Branco	272000272	Apagar	Editar
Centro de Saude de Sao Tiago	Rua Antonio Sergio 10, 6000-152 Castelo Branco	272340290	Apagar	Editar
Centro de Saude de Sao Miguel	Avenida da Europa Apartado 110 6000-491 Castelo Branco	272340150	Apagar	Editar
Hospital de Coimbra	Avenida Alvares	272111888	Apagar	Editar
Hospital Lisboa	Avenida da Liberdade	977222828	Apagar	Editar

Figura 13 - Lista de Paragens

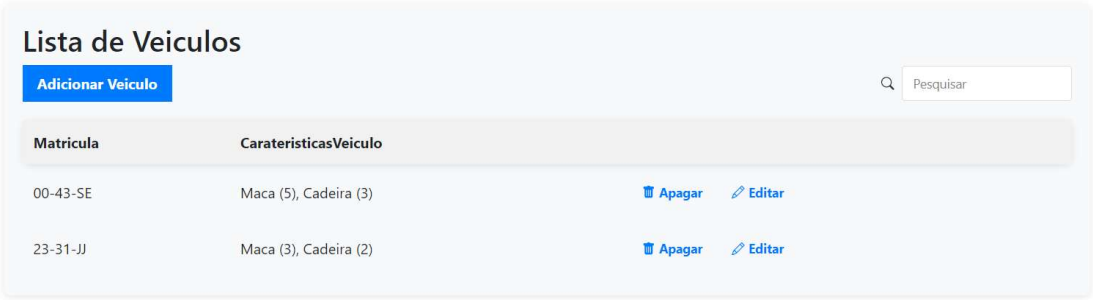


Figura 14 - Lista de Veículos

Figura 15 - Lista de Agendamentos

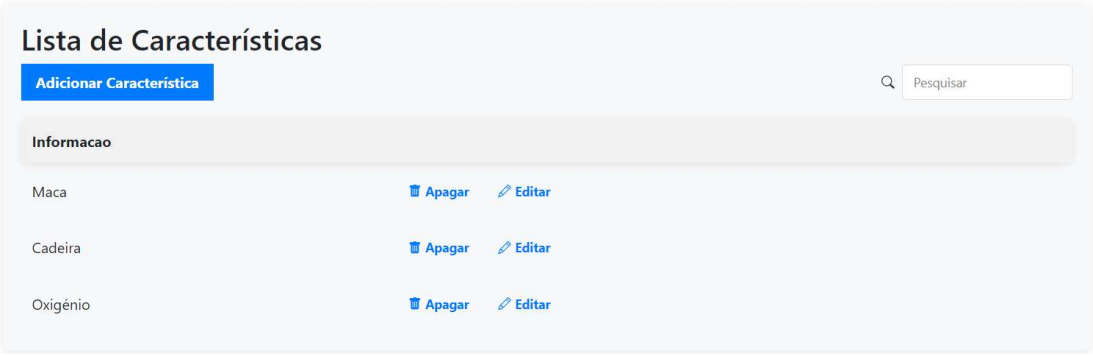


Figura 16 - Lista de Características

Já no contexto de operações CRUD, como exemplo vamos utilizar a tabela Motorista e Paragem, podemos observar algo minimalista por opção, mas muito objetiva em que temos disponível a opção de adicionar um motorista, editar e eliminar caso seja necessário. Existe também uma barra de pesquisa no canto superior direito para ser mais fácil procurar qualquer dado assim que a lista se começar a estender. Todavia, também retiramos partido do react no que toca a componentes, minimizamos o máximo de páginas ao realizar qualquer operação na página do mesmo em vez de ao editar ir para uma página, adicionar para outra e assim sucessivamente. Nas outras páginas embora com diferentes funcionalidades e validações nos campos o intuito é o mesmo.

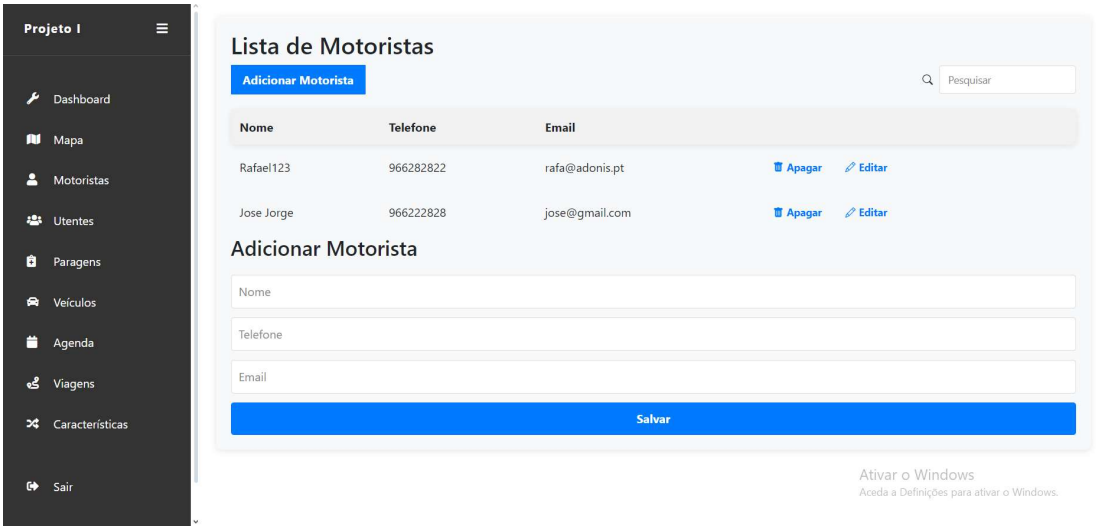


Figura 17 - Adicionar Motorista

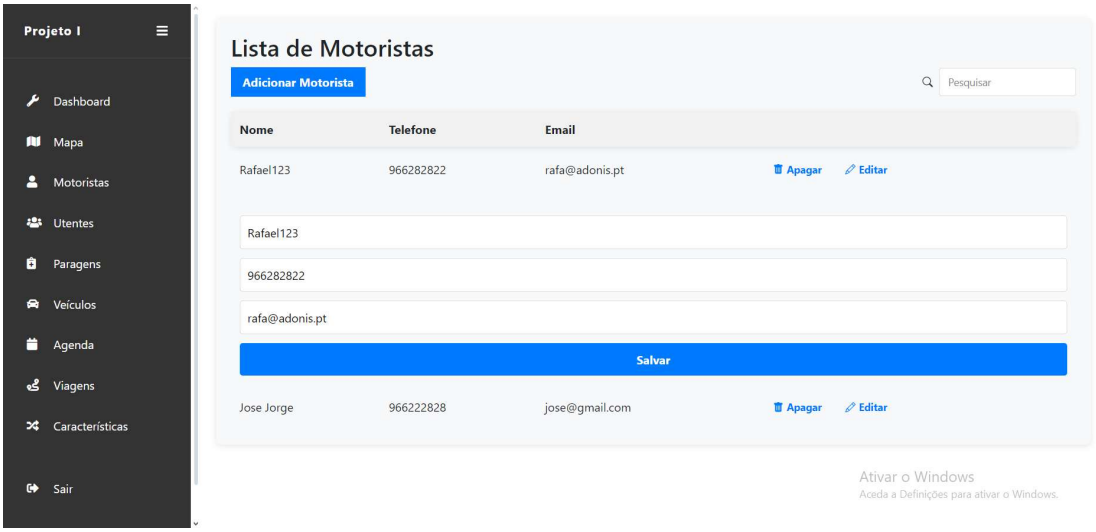


Figura 18 - Editar Motorista

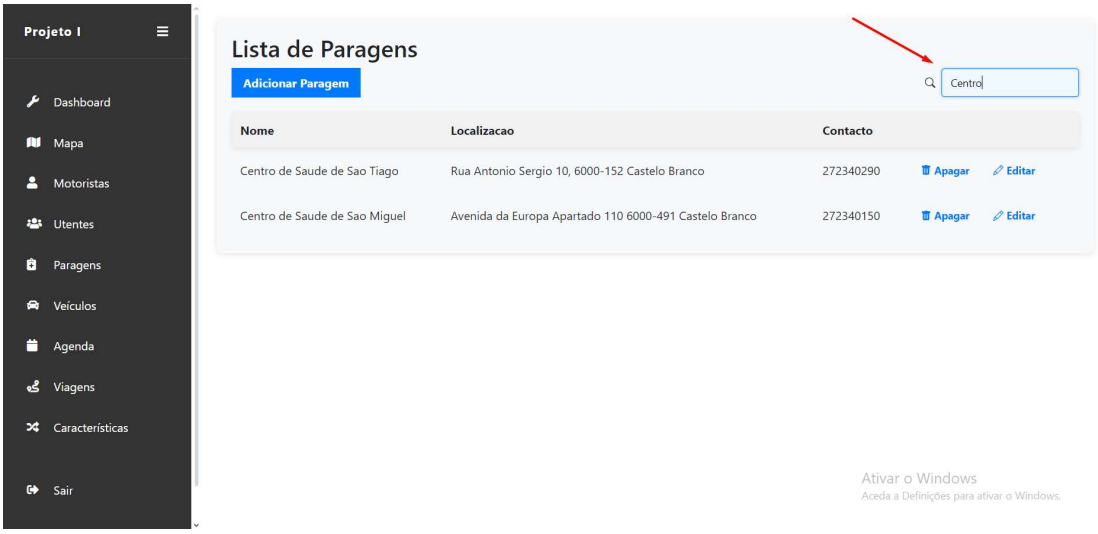


Figura 19 - Barra de Pesquisa

No que toca a mensagens também optamos por uma dinâmica de exibir mensagens de texto caso haja algum erro num formulário, falte preencher algum campo e até mesmo ao concluir a edição, remoção e adição de qualquer operação exibe uma mensagem no topo da página com a principal diferença de cor caso esteja certo ou errado durante três segundos.

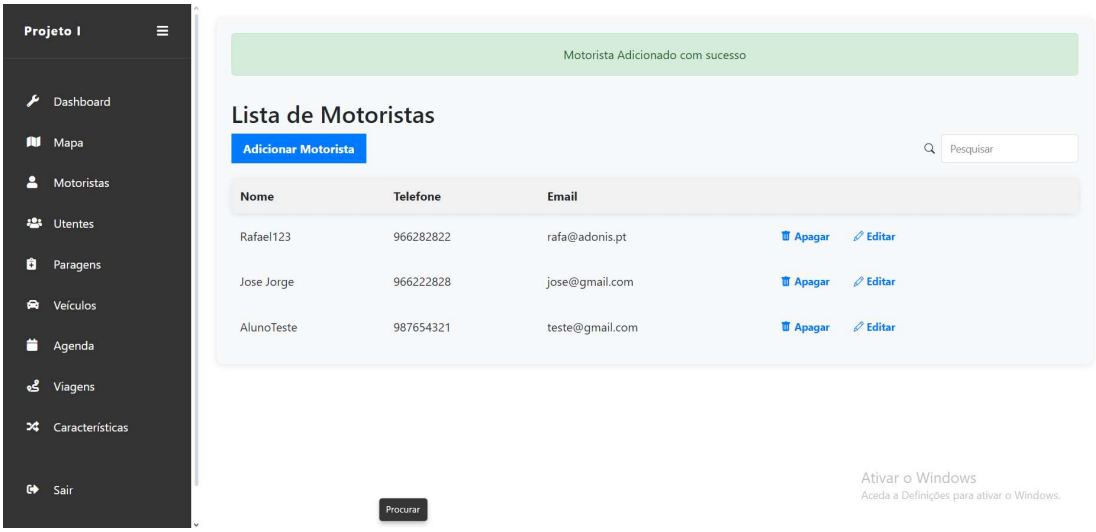


Figura 20 - Mensagem com Sucesso

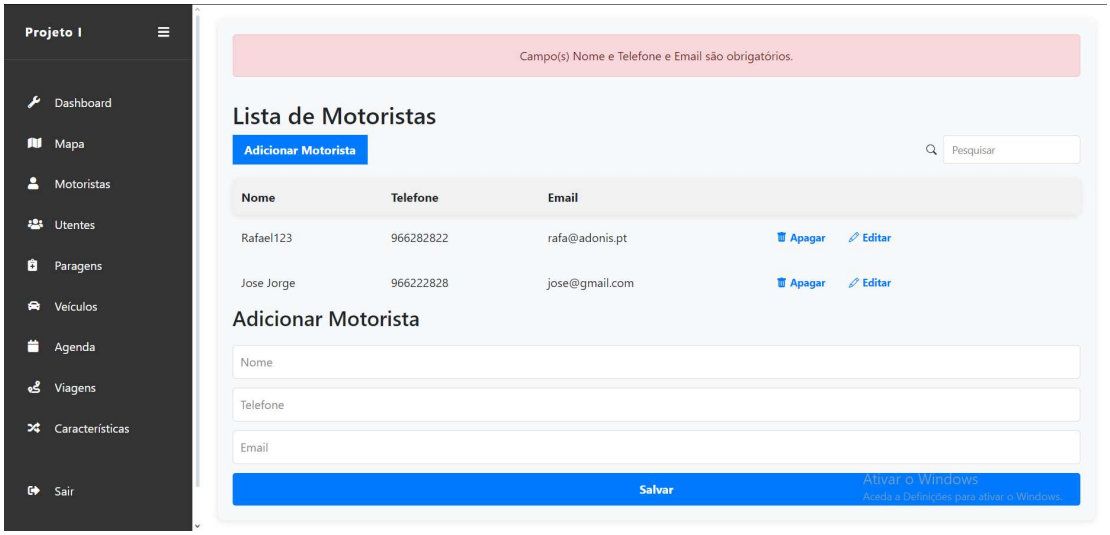


Figura 21 - Mensagem com Erro

2.2. Serviço de mapas

O objetivo principal deste projeto e desenvolver um sistema que permita a criação de rotas de maneira dinâmica e eficiente. Para atingir esse objetivo, será necessário integrar produtos de terceiros que oferecem funcionalidades como mapas, geração de itinerários entre outros que possam ser uteis para fazer a otimização das rotas como por exemplo informações de trânsito. As opções que decidimos analisar foram o Google Maps[9], Azure Maps[10] e o Mapbox[11].

Embora o desenvolvimento desta funcionalidade fique apenas para o projeto II, já foi realizada uma pesquisa para escolher a melhor solução tendo em conta fatores como custo, funcionalidades e facilidade de integração.

Inicialmente a solução preferida era o Azure Maps, pois tínhamos conhecimento prévio do mesmo, e possuía os serviços de criação de itinerários e trânsito em tempo real que necessitávamos. No entanto após uma discussão entre os membros do projeto e o orientador foi decido não utilizar este serviço devido aos custos associados ao mesmo.

Devido a isso foram exploradas outras alternativas e acabou por ser escolhido o Mapbox. Este serviço não só oferece funcionalidades semelhantes ao Azure Maps, como também tem um tier gratuito mais acessível. Alem disso o Mapbox oferece recursos adicionais como SDKs[12] para criações de aplicações móveis de navegação, que possibilita a criação de uma experiência melhor para os motoristas, e ainda com recursos para criar mapas interativos no front-end possibilitando a visualização das rotas a serem executadas.

Adicionalmente o Mapbox oferece uma documentação[13] robusta e fácil de interpretar, conta também com várias demonstrações[14] de como os serviços funcionam facilitando assim a futura implementação.

Como base nessas considerações, foi escolhido o Mapbox por ser a solução que oferecia o melhor conjunto de funcionalidades e um modelo de preços mais compatível com as exigências do projeto.

3. Conclusão

Nesta secção, faremos um resumo do trabalho realizado durante o Projeto I e apresentaremos o planejamento para as atividades futuras que serão desenvolvidas no Projeto II.

3.1. Resumo do trabalho realizado

Para o Projeto I, conseguimos concretizar, com rigor, todos os aspetos que inicialmente propusemos. A maior dificuldade foi, num primeiro momento, comprometermo-nos a aprender novas tecnologias para o desenvolvimento do projeto. No entanto, recorrendo a vídeos[15] e informações disponíveis na internet, obtivemos os conhecimentos necessários para dar início ao trabalho.

Desenhámos uma base de dados que passou por várias revisões e modificações ao longo do processo. Embora atualmente esteja definida, prevemos que no Projeto II possam ocorrer novas alterações.

Iniciámos então o desenvolvimento do front-end e do back-end, beneficiando do facto de ambos utilizarem Node.js, o que permitiu uma integração mais eficiente entre os dois. Além disso, para otimizar o progresso e a construção do projeto, decidimos dividir as responsabilidades, com cada membro da equipa focado numa parte específica. Acreditámos que, ao distribuir as tarefas desta forma, conseguiríamos cumprir os prazos e alcançar todos os objetivos estabelecidos.

3.2. Trabalho futuro

Para concluir este relatório, iremos falar sobre o trabalho que será desenvolvido em Projeto II. Esse trabalho será dividido em três componentes principais: o desenvolvimento de duas aplicações moveis, o sistema de criação de rotas dinâmicas e a integração entre todos os componentes.

3.2.1. Aplicações Móveis

Serão desenvolvidas duas aplicações móveis diferentes para cada tipo de utilizador (utente e motorista).

Aplicação para o utente:

- A aplicação do utente irá permitir este realizar as seguintes operações:
  - Agendar transporte
  - Consultar o histórico de viagem realizadas
  - Criar e editar a sua conta pessoal

Aplicação para o motorista:

- A aplicação do motorista servira como um guia, tendo como funcionalidade:
  - Marca passageiros e paragens como feitas
  - Fornecer instruções de navegação

Ambas as aplicações serão desenvolvidas utilizando a linguagem Kotlin[16] e a plataforma Android Studio[17], será também utilizado as SDK disponibilizada pelo Mapbox referidas no ponto “Escolha do serviço de mapas”.

3.2.2. Sistema de criação de rotas dinâmicas

Um dos componentes mais importantes do projeto e o componente responsável pela criação das rotas. Alguns dos requisitos para este sistema são:

- **Criação eficiente de rotas:** O sistema deve ser capaz de criar rotas automaticamente e de maneira eficiente (mais rápida por exemplo) tendo em conta a localização dos utentes, motoristas e paragens.
- **Adaptação em tempo real:** O sistema será capaz de ajustar rotas em tempo real, reagindo a fatores como o trânsito ou acidentes
- **Criação de novas rotas:** Em situações em que a criação de uma nova rota seja mais eficiente do que adaptar uma existente, o sistema deverá fazer essa alteração de maneira automática.

4. Referências

[1] “React.” Accessed: Jan. 23, 2025. [Online]. Available: https://react.dev/

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

[3] “MySQL.” Accessed: Jan. 22, 2025. [Online]. Available: https://www.mysql.com/

[4] “SQL Server Downloads | Microsoft.” Accessed: Jan. 22, 2025. [Online]. Available: https://www.microsoft.com/en-us/sql-server/sql-server-downloads

[5] “Node.js — Run JavaScript Everywhere.” Accessed: Jan. 22, 2025. [Online]. Available: https://nodejs.org/en

[6] “Express - Node.js web application framework.” Accessed: Jan. 22, 2025. [Online]. Available: https://expressjs.com/

[7] “mysql2 - npm.” Accessed: Jan. 23, 2025. [Online]. Available: https://www.npmjs.com/package/mysql2

[8] “REST API Documentation Tool | Swagger UI.” Accessed: Jan. 22, 2025. [Online]. Available: https://swagger.io/tools/swagger-ui/

[9] “Google Maps Platform | Google for Developers.” Accessed: Jan. 23, 2025. [Online]. Available: https://developers.google.com/maps

[10] “Azure Maps | Microsoft Azure.” Accessed: Jan. 23, 2025. [Online]. Available: https://azure.microsoft.com/en-us/products/azure-maps

[11] “Mapbox | Maps, Navigation, Search, and Data.” Accessed: Jan. 23, 2025. [Online]. Available: https://www.mapbox.com/

[12] “Navigation SDK | Android Docs | Mapbox.” Accessed: Jan. 23, 2025. [Online]. Available: https://docs.mapbox.com/android/navigation/guides/

[13] “Mapbox Docs.” Accessed: Jan. 24, 2025. [Online]. Available: https://docs.mapbox.com/

[14] “Demos & Projects | Resources | Mapbox Docs | Mapbox.” Accessed: Jan. 24, 2025. [Online]. Available: https://docs.mapbox.com/resources/demos-and-projects/

[15] “Curso React: Introdução - #01 - YouTube.” Accessed: Jan. 30, 2025. [Online]. Available: https://www.youtube.com/watch?v=FXqX7oof0I4&list=PLnDvRpP8BneyVA0SZ2okm-QBojomniQVO

[16] “Kotlin Programming Language.” Accessed: Jan. 27, 2025. [Online]. Available: https://kotlinlang.org/

[17] “Download Android Studio & App Tools - Android Developers.” Accessed: Jan. 27, 2025. [Online]. Available: https://developer.android.com/studio

5. Anexos

Endpoint	Método HTTP	Descrição	Parâmetros	Códigos de Resposta
/api/v1/agendamentos	GET	Receber agendamentos	Nenhum	200: Success 404: Não existe agendamentos 400: Error
/api/v1/agendamentos	POST	Criar Agendamento	agendamento	201: Criado 400: Erro
/api/v1/agendamentos/{id}	GET	Recebe agendamento pelo id	id	200: Success 404: Não existe agendamento com esse id 400: Error
/api/v1/agendamentos/{id}	PUT	Editar agendamento	id	201: Criado 404: Não existem agendamento com esse id 400: Erro
/api/v1/agendamentos/{id}	DELETE	Eliminar agendamento pelo id	id	200: Success 404: Não existe com esse id 400: Error
/api/v1/agendamentos/{id}/feito	POST	Marca agendamento como feito	id	201: Criado 400: Erro
/api/v1/agendamentos/paragem/{id}	GET	Receber agendamentos pelo id paragens	id	200: Success 404: Não existe agendamentos para esta paragem 400: Error
/api/v1/agendamentos/data/{data}	GET	Receber agendamentos pela data	data	200: Success 404: Não existe agendamentos 400: Error
/api/v1/agendamentos/utente/{id}	GET	Receber agendamentos do utente pelo id	id	200: Success 404: Não existe agendamentos para esse utente 400: Error
/api/v1/carateristicas	GET	Receber todos as caraterísticas	Nenhum	200: Success 404: Não existem caraterísticas 400: Error
/api/v1/carateristicas	POST	Criar caraterísticas	caraterística	201: Criado 400: Erro
/api/v1/carateristicas/{id}	GET	Recebe caraterísticas pelo id	id	200: Success 404: Não existe caraterísticas

				com esse id 400: Error
/api/v1/carateristicas/{id}	PUT	Editar caraterísticas	id	201: Criado 404: Não existem caraterísticas com esse id 400: Erro
/api/v1/carateristicas/{id}	DELETE	Eliminar caraterísticas pelo id	id	200: Success 404: Não existe caraterísticas com esse id 400: Error
/api/v1/motoristas	GET	Receber todos os motoristas	Nenhum	200: Success 404: Não existe motoristas 400: Error
/api/v1/motoristas	POST	Criar motorista	motorista	201: Criado 400: Erro
/api/v1/motoristas/{id}	GET	Recebe motorista pelo id	id	200: Success 404: Não existe motorista 400: Error
/api/v1/motoristas/{id}	PUT	Editar motorista	id	201: Criado 404: Não existem motorista com esse id 400: Erro
/api/v1/motoristas/{id}	DELETE	Eliminar motorista pelo id	id	200: Success 404: Não existe motorista com esse id 400: Error
/api/v1/paragens	GET	Receber todas as paragens	Nenhum	200: Success 404: Não existem paragens 400: Error
/api/v1/paragens	POST	Criar Paragem	paragem	201: Criado 400: Erro
/api/v1/paragens/{id}	GET	Recebe paragem pelo id	id	200: Success 404: Não existe paragem com esse id 400: Error
/api/v1/paragens/{id}	PUT	Editar Paragem	id	201: Criado 404: Nao existem paragem com esse id 400: Erro

/api/v1/paragens/{id}	DELETE	Eliminar paragem pelo id	id	200: Success 404: Não existe paragem com esse id 400: Error
/api/v1/utentes	GET	Receber todos os utentes	Nenhum	200: Success 404: Não existe utente 400: Error
/api/v1/utentes	POST	Criar utente	utente	201: Criado 400: Erro
/api/v1/utentes/{id}	GET	Recebe utente pelo CC	id	200: Success 404: Não existe utente com esse CC 400: Error
/api/v1/utentes/{id}	PUT	Editar Utente	id	201: Criado 404: Não existem utente com esse CC 400: Erro
/api/v1/utentes/{id}	DELETE	Eliminar utente pelo CC	id	200: Success 404: Não existe utente com esse CC 400: Error
/api/v1/utentes/{id}/localizacao	GET	Recebe localização do utente pelo CC	id	200: Success 404: Não existe utente com esse CC 400: Error
/api/v1/utentes/{id}/localizacao	PUT	Editar localização do Utente	id	201: Criado 404: Não existem utente com esse CC 400: Erro
/api/v1/utentes/{id}/localizacao	DELETE	Eliminar localização do utente pelo CC	id	200: Success 404: Não existe utente com esse CC 400: Error
/api/v1/veiculos	GET	Receber todos os veículos	Nenhum	200: Success 404: Não existem veículos 400: Error
/api/v1/veiculos	POST	Criar veículo	veículo	201: Criado 400: Erro
/api/v1/veiculos/{matricula}	GET	Recebe veículo pela matricula	matrícula	200: Success 404: Não existe veículo com essa matrícula 400: Error

/api/v1/veiculos/{matricula}	PUT	Editar veículo	matrícula	201: Criado 404: Não existe veículo com essa matrícula 400: Erro
/api/v1/veiculos/{matricula}	DELETE	Eliminar veículo pela matrícula	matrícula	200: Sucesso 404: Não existe veículo com essa matrícula 400: Error
/api/v1/veiculos/{id}/localizacao	GET	Recebe localização do veículo pelo id	id	200: Success 404: Não existe veículo com esse id 400: Error
/api/v1/veiculos/{id}/localizacao	PUT	Editar localização do veículo	id	201: Criado 404: Não existem veículo com esse id 400: Erro
/api/v1/veiculos/{id}/localizacao	DELETE	Eliminar localização do veículo pelo id	id	200: Success 404: Não existe veículo com esse id 400: Error
/api/v1/viagens	GET	Receber todos as viagens	Nenhum	200: Success 404: Não existem viagens 400: Error
/api/v1/viagens	POST	Criar viagem	viagem	201: Criado 400: Erro
/api/v1/viagens/{id}	GET	Recebe viagem pelo id	id	200: Success 404: Não existe viagem com esse id 400: Error
/api/v1/viagens/{id}/estado	GET	Recebe estado da viagem pelo id	id	200: Success 404: Não existe viagem com esse id 400: Error
/api/v1/viagens/{id}/estado	PUT	Editar estado da viagem	id	201: Criado 404: Não existem viagem com esse id 400: Erro
/api/v1/viagens/{id}/paragem	PUT	Adiciona uma paragem a viagem	id	200: Success 404: Não existe viagem/paragem com esse id 400: Error
/api/v1/viagens/{id}/paragem	DELETE	Remove uma paragem a viagem	id	200: Success 404: Não existe viagem/paragem com esse id 400: Error

/api/v1/viagens/{id}/utente	PUT	Adiciona um utente a viagem	id	200: Success 404: Não existe viagem/utente com esse id 400: Error
/api/v1/viagens/{id}/utente	DELETE	Remove um utente a viagem	id	200: Success 404: Não existe viagem/utente com esse id400: Error
/api/v1/viagens/{id}/horapartida	PUT	Adicionar a hora de partida	id	200: Success 404: Não existe viagem esse id 400: Error
/api/v1/viagens/{id}/horachegada	PUT	Adicionar a hora de chegada	id	200: Success 404: Não existe viagem esse id 400: Error
/api/v1/viagens/{id}/marcarutente	PUT	Marcar utente como levantado	id	200: Success 404: Não existe viagem/utente esse id 400: Error
/api/v1/viagens/{id}/marcarparagem	PUT	Marcar paragem como feita	id	200: Success 404: Não existe viagem/paragem esse id 400: Error
/api/v1/admin/login	POST	Fazer login como administrador	Nenhum	200: Success 404: Login invalido 400: Error