TQS: Product specification report

Gonçalo Abrantes[104152] João Morais [103730] Pedro Rei [107463]
v2025-05-16

1 Introduction

1.1 Overview of the project

No âmbito da nossa cadeira de TQS, este projecto tem como objectivo aplicar os conhecimentos adquiridos no desenvolvimento de uma solução digital robusta, funcional e centrada no utilizador. O foco esteve na criação de um produto que resolvesse um problema real, recorrendo a boas práticas de engenharia de software, desde o levantamento de requisitos até à validação da solução. A iniciativa focou-se na aplicação de princípios fundamentais da engenharia de software, com especial destaque para as metodologias orientadas à qualidade, como o desenvolvimento orientado por testes (TDD) e os processos de Garantia da Qualidade (QA).

O resultado dos nossos esforços é a nossa aplicação, que tem como seu objetivo facilitar o acesso a postos de carregamento para veículos eléctricos. Através de uma plataforma integrada, o utilizador pode localizar postos disponíveis em tempo real, efectuar reservas e realizar pagamentos de forma simples e eficiente. Com a nossa aplicação, nós pretendemos tornar a experiência de carregamento mais fluida, previsível e adaptada às exigências da mobilidade elétrica moderna.

Durante o desenvolvimento, adotou-se a abordagem TDD, onde os testes são escritos antes da implementação funcional. Esta metodologia permitiu garantir que o código fosse construído de forma incremental, com maior confiança na sua robustez e na detecção precoce de erros. Paralelamente, foram aplicados conceitos de QA como a validação contínua, testes unitários e testes de usabilidade, assegurando que o sistema final não só cumprisse os requisitos técnicos, mas também proporcionasse uma boa experiência ao utilizador.

1.2 Known limitations

O sistema implementado apresenta-se como uma solução robusta, integrando funcionalidades fundamentais como a apresentação de estatísticas, um mapa interativo e outras características consideradas essenciais para este tipo de plataforma. No entanto, existem alguns aspetos que poderiam ter sido mais desenvolvidos ou otimizados:

- Ausência de cálculo de CO2 poupado: Embora o sistema apresente estatísticas históricas, estas não são complementadas com qualquer tipo de processamento adicional, como a estimativa das emissões de CO2 evitadas. Esta funcionalidade seria particularmente relevante para reforçar o impacto ambiental positivo do sistema.

- Falta de distinção entre utilizador e administrador: Atualmente, não existe uma separação clara entre os perfis de utilizador comum e administrador. Ambos acedem à mesma interface web, sendo que o administrador apenas necessita de introduzir uma palavra-passe. Este método representa uma vulnerabilidade, pois qualquer utilizador com acesso à palavra-passe pode desempenhar ações administrativas.

- Inexistência de uma aplicação móvel: O sistema foi desenvolvido exclusivamente como uma plataforma web, sem qualquer versão dedicada para dispositivos móveis. A inclusão de uma aplicação móvel poderia melhorar significativamente a acessibilidade e usabilidade, especialmente para utilizadores em movimento.

1.3 References and resources

Neste projeto foi realizada a integração com o OpenStreetMap para a criação de mapas interativos, proporcionando uma visualização geográfica dinâmica e informativa das estações de abastecimento. Adicionalmente, foi utilizado o Swagger para a documentação e teste da API, facilitando a compreensão e a interação com os serviços disponibilizados pelo sistema.

2 Product concept and requirements

2.1 Vision statement

Um dos principais desafios enfrentados por quem possui um veículo elétrico é a dificuldade em encontrar e utilizar estações de carregamento de maneira rápida, confiável e integrada. Em áreas urbanas, essa limitação acaba por comprometer a praticidade e a eficiência da mobilidade elétrica. Pensando nisso, o sistema desenvolvido busca justamente facilitar esse processo. A proposta é oferecer uma aplicação que reúna, numa única plataforma, recursos para localizar, reservar e pagar pelo uso das estações de carregamento, tornando a experiência do utilizador mais simples e eficiente.

Para atender a essa demanda, o sistema oferece diversas funcionalidades integradas. A primeira delas é a exibição em tempo real das estações de carregamento disponíveis, por meio de um mapa interativo que permite a aplicação de filtros como tipo de carregador e velocidade de carga. Outra funcionalidade importante é a possibilidade de reservar uma estação com antecedência, o que ajuda a evitar filas e otimizar o tempo do utilizador. O pagamento também pode ser feito diretamente através da aplicação, com suporte a diferentes métodos, como cartões de crédito e débito, além de

carteiras digitais. As estações apresentam ainda uma seção com informações detalhadas, como o tipo de carregador, a potência oferecida, os horários de funcionamento, o status atual (ocupada ou disponível) e as avaliações feitas por outros usuários. Para completar, o sistema envia notificações automáticas, como lembretes sobre o início e o fim do carregamento, horários de reserva e alterações no status das estações, garantindo que o usuário esteja sempre informado.

Esta aplicação também terá acessos exclusivos a funcionários das estações de carregamento, de forma a serem capazes de saber em tempo real o estado das estações, como por exemplo, se precisam de manutenção. Além disso, também poderão ver a quantidade de remuneração que cada estação anda a trazer.

Embora existam outras plataformas populares no setor, como a PlugShare e a Electromaps, que também oferecem mapas com a localização de estações e informações técnicas, a aplicação destaca-se por integrar, de forma contínua, todas as etapas do processo num único ambiente. Enquanto outras soluções exigem que o utilizador recorra a diferentes ferramentas para localizar, reservar e pagar, o sistema proposto concentra tudo isso em um só lugar, proporcionando uma experiência mais prática e fluida.

2.2 Personas and scenarios

Persona 1: **Joana Silva (Condutora de Veículos Elétricos)**
Idade: 32 anos
Naturalidade: Lisboa

Joana Silva é uma dona de casa, com dois filhos e um marido que trabalha. Como encarregada de educação, Joana vê-se a utilizar o carro elétrico dela para conduzir as crianças para a escola delas, tal como para as suas atividades extracurriculares e para quaisquer recados que ela tenha. Desta forma, o carro precisa sempre de ser carregado com frequência numa estação que existe em frente ao edifício em que a família vive.

Para garantir que consegue carregar, ela verifica sempre através da aplicação se existe alguma estação livre, e reserva online, de forma a poder recarregar o veículo e também precisa de uma forma de pagamento fácil.

Persona 2: **Carlos Mendes (Funcionário do Posto de carregamento)**
Idade: 45 anos
Naturalidade: Braga

Carlos Mendes é um funcionário dos Postos de carregamento de carros elétricos, tendo como função verificar os estados das estações e verificar se os níveis de consumo estão dentro do normal. Para poder realizar a sua função, ele precisará de um acesso exclusivo para funcionários, de forma que utilizadores normais não sejam capazes de aceder a informação potencialmente sensível. Este acesso para funcionários terá que ser capaz de fornecer informação acerca dos postes, quais estão disponíveis, e quais precisam de manutenção, e obter informação acerca do consumo agregado dos postos.

Persona 3: **Graça Gomes (Condutora)**
Idade: 25 anos
Naturalidade: Porto

Graça é uma estudante universitária, que teve a sorte de ganhar um carro elétrico num sorteio. No entanto, viver na residência de estudantes causa o único problema de ter que procurar por estações de carregamento para o seu carro. Além disso, a quantidade de dinheiro que ela tem para pagar não é sempre muito. Por estas razões, Graça procura juntar o útil ao agradável, e precisa de ser capaz de ver os custos das estações de carregamento perto dela, de forma a encontrar uma que seja perto da residência e que seja barato o suficiente para justificar a distância.

2.3 Project epics and priorities

Neste projeto, adotamos o Scrum com sprints semanais e reuniões de planeamento às sextas-feiras, às 16h. Nessas reuniões, definiremos as *issues* da semana, os respetivos critérios de aceitação e os *reviewers*. O fluxo de desenvolvimento seguirá o GitHub Flow: para cada nova *issue*, criaremos uma *feature branch* isolada a partir da *main branch*. Após a implementação, o programador garantirá a aprovação de todos os testes unitários e uma cobertura de código de pelo menos 80%, além de realizar uma análise com o SonarQube para identificar e corrigir problemas críticos. Uma *pull request* só será aceite na *main branch* após aprovação do *reviewer* designado no planeamento da *sprint*. Caso sejam identificados erros após a integração do código na branch principal, a responsabilidade pela correção recai sobre o programador que criou o Pull Request original.

3 Domain model

Neste domínio, os principais conceitos de informação que serão geridos são:

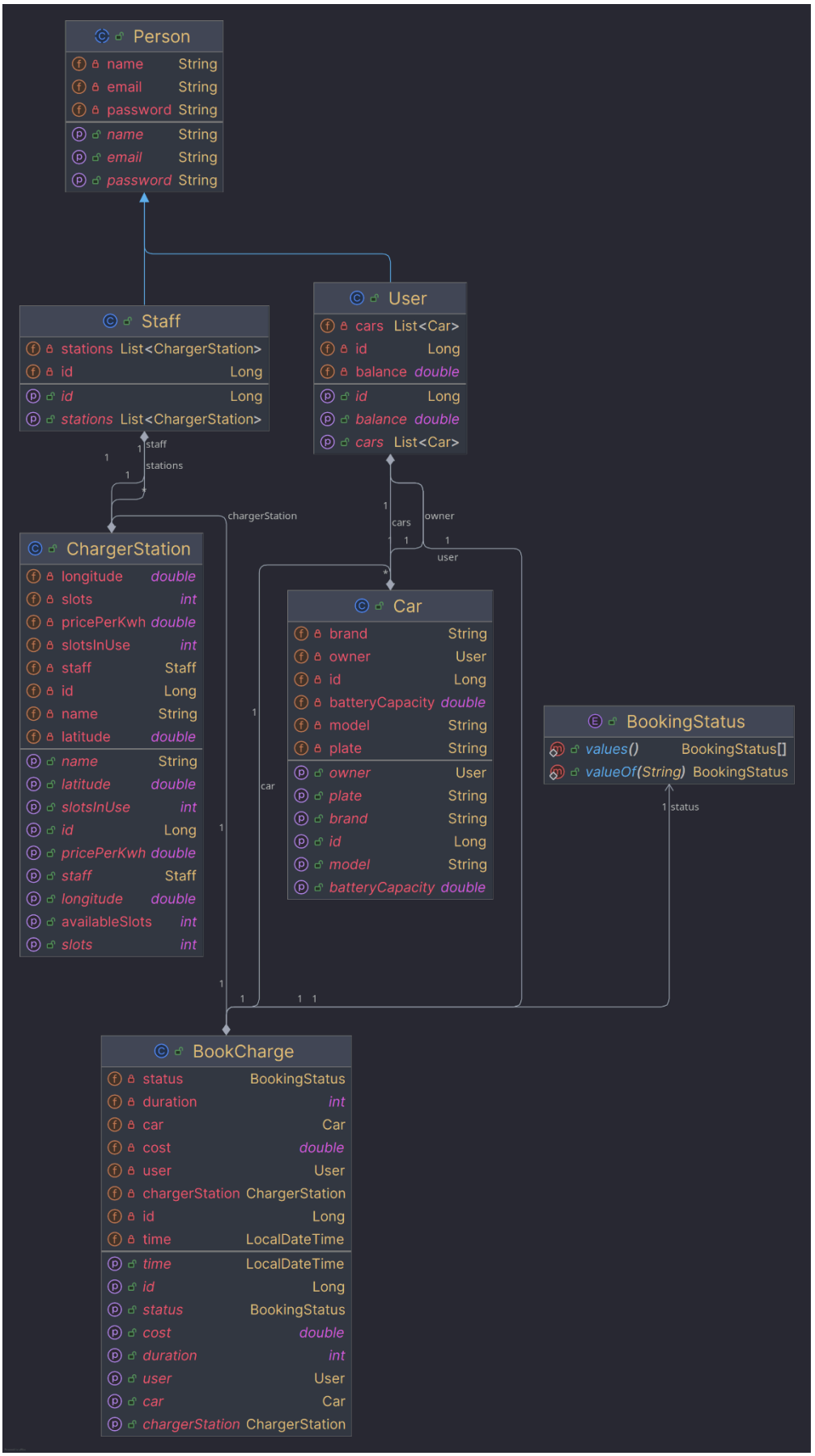
- **Utilizador (User)**: Representa a pessoa que utiliza o sistema para reservar sessões de carregamento de veículos elétricos.
- **Carro (Car)**: Representa um veículo pertencente a um utilizador, que será utilizado para o carregamento.
- **Estação de Carregamento (ChargerStation)**: Representa o local físico onde os veículos podem ser carregados.
- **Reserva de Carregamento (BookCharge)**: Representa uma reserva feita por um utilizador para carregar o seu carro numa estação específica. Contém informações como a data e hora da reserva, a duração, o custo e o estado da reserva.

- **Estado da Reserva (BookingStatus)**: Enumeração que representa o estado atual da reserva (por exemplo: PENDENTE, CONFIRMADA, CANCELADA).

Os conceitos estão relacionados de forma a refletir a lógica do domínio da aplicação. Um utilizador pode realizar várias reservas de carregamento, sendo que cada reserva está associada a um único utilizador. Cada reserva de carregamento refere-se também a um carro específico, pertencente ao utilizador, e a uma estação de carregamento, onde o processo ocorrerá.

Um carro está sempre ligado a um único utilizador, mas pode estar envolvido em várias reservas ao longo do tempo. Do mesmo modo, uma estação de carregamento pode ser utilizada em várias reservas diferentes por diversos utilizadores. Além disso, cada reserva possui um estado, definido por um valor da enumeração BookingStatus, que indica em que fase se encontra (por exemplo, pendente, confirmada ou cancelada).

Esta estrutura permite ao sistema representar de forma clara e eficiente todas as interações entre os utilizadores, os veículos, os locais de carregamento e o histórico das reservas efetuadas.


Person
name String
email String
password String
name String
email String
password String
Staff
stations List<ChargerStation>
id Long
id Long
stations List<ChargerStation>
User
cars List<Car>
id Long
balance double
id Long
balance double
cars List<Car>
staff
stations
chargerStation
cars
owner
user
ChargerStation
longitude double
slots int
pricePerKwh double
slotsInUse int
staff Staff
id Long
name String
latitude double
name String
latitude double
slotsInUse int
id Long
pricePerKwh double
staff Staff
longitude double
availableSlots int
slots int
Car
brand String
owner User
id Long
batteryCapacity double
model String
plate String
owner User
plate String
brand String
id Long
model String
batteryCapacity double
car
BookingStatus
values() BookingStatus[]
valueOf(String) BookingStatus
status
BookCharge
status BookingStatus
duration int
car Car
cost double
user User
chargerStation ChargerStation
id Long
time LocalDateTime
time LocalDateTime
id Long
status BookingStatus
cost double
duration int
user User
car Car
chargerStation ChargerStation

4 Architecture notebook

4.1 Key requirements and constraints

A aplicação terá uma interface intuitiva e de fácil utilização, dada a diversidade de perfis de utilizadores. Essencialmente, integrará dois serviços externos principais: um para o processamento de pagamentos e outro para fornecer um mapa atualizado da cidade. Por ser uma aplicação de carregamento, prevemos picos de utilização nas horas de almoço e de saída do trabalho, exigindo que o sistema responda eficientemente a múltiplos utilizadores simultaneamente. Dada a sua operação em múltiplos dispositivos móveis e computadores, o desenvolvimento terá especial atenção à responsividade do *design* para diferentes dimensões de ecrã.

4.2 Architecture view

Para a arquitetura do nosso sistema, optámos por uma abordagem monolítica, com um backend desenvolvido em Spring Boot e uma base de dados partilhada em MySQL, que serve tanto os utilizadores comuns como os membros da equipa (staff). Esta escolha foi feita para simplificar o desenvolvimento inicial e acelerar a fase de arranque do projeto.

Em vez de desenvolver duas interfaces distintas, implementámos uma *Single Page Application* (SPA) com React, onde o conteúdo e as funcionalidades apresentadas variam consoante o tipo de sessão (utilizador ou staff). Isto permitiu-nos centralizar o desenvolvimento da interface, mantendo uma separação clara entre os papéis.

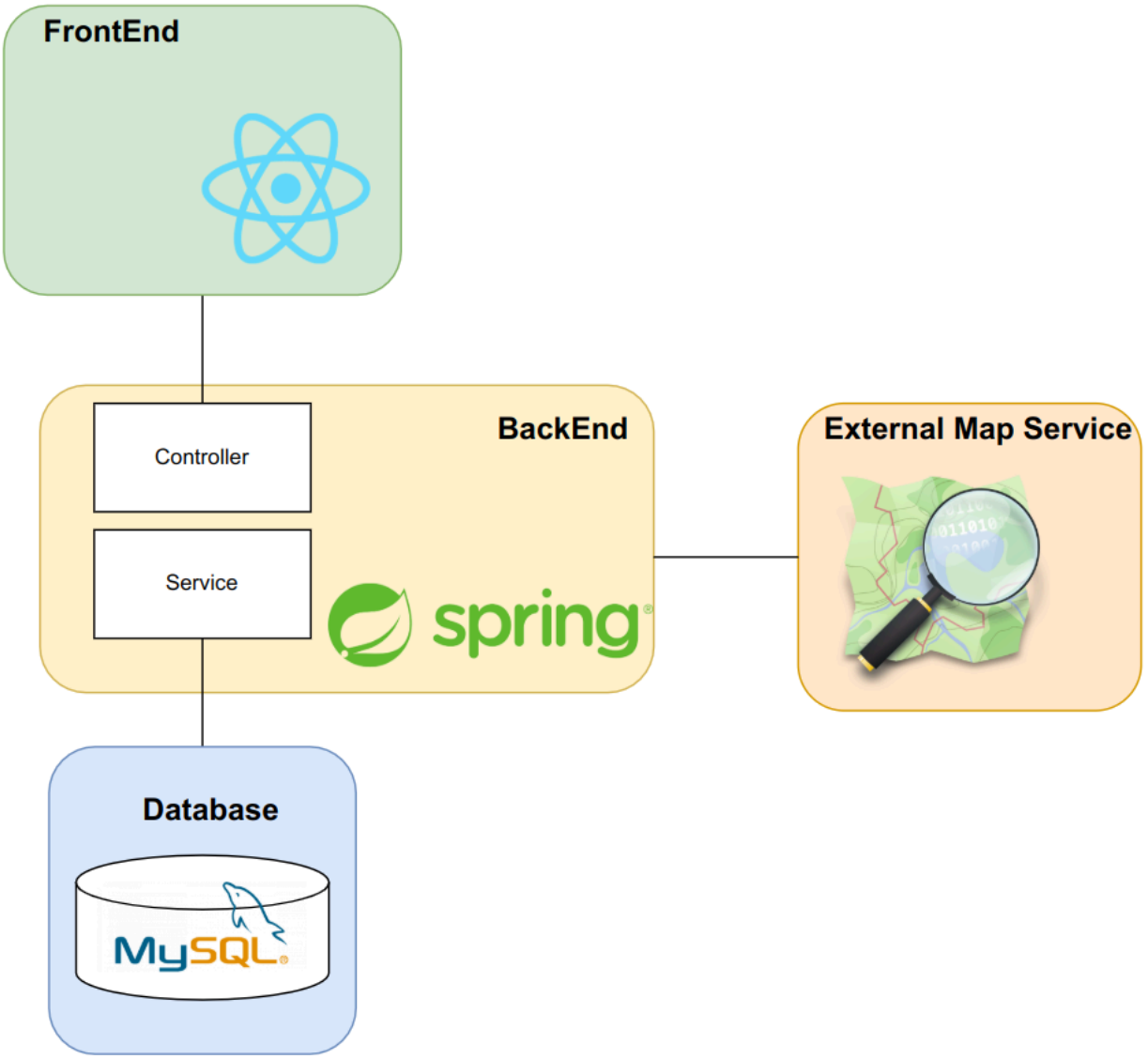
No arranque da aplicação, o backend em Spring Boot gera automaticamente as tabelas da base de dados com base nas definições dos DTOs e das entidades. Para facilitar os testes e a validação das funcionalidades, corre-se também um script de inicialização que preenche a base de dados com dados simulados (*mock*), como utilizadores, veículos, estações de carregamento e reservas.

Quanto ao frontend, este foi desenvolvido com React, proporcionando uma experiência fluida e interativa. Os utilizadores podem registar-se, adicionar os seus veículos elétricos à plataforma e, através de um mapa interativo alimentado pelo serviço externo OpenStreetMap, localizar e selecionar a estação de carregamento da sua preferência. Depois de escolher uma estação, o sistema verifica se o utilizador possui saldo suficiente. Caso tenha, é criada uma reserva, que é armazenada na base de dados com o respetivo estado. O estado da reserva é posteriormente atualizado com base nas ações do utilizador (como conclusão ou cancelamento), permitindo uma gestão eficiente das sessões de carregamento.

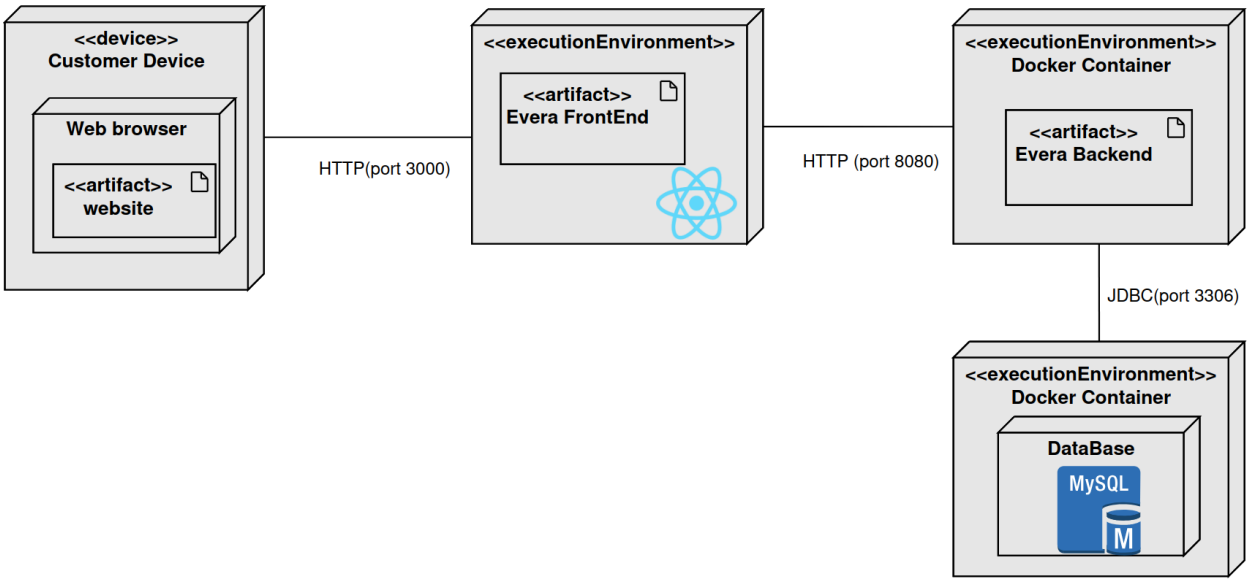
Os administradores acedem à sua área reservada através de um sistema baseado em palavra-passe. Após autenticação bem-sucedida, são redirecionados para uma interface específica de gestão, onde podem realizar diversas operações administrativas.

Nesta área, o staff pode executar operações CRUD (Criar, Ler, Atualizar e Eliminar) sobre as estações de carregamento, utilizadores registados e reservas existentes. Além disso, têm acesso a um painel de estatísticas de vendas, onde podem consultar informações relevantes sobre a utilização da plataforma, como número de reservas, carregamentos concluídos e total de receita gerada.

Estas estatísticas são geradas com base nos dados já armazenados na base de dados, sendo processadas no backend e apresentadas de forma visual na interface, permitindo uma análise rápida e eficaz da atividade da plataforma.



4.3 Deployment view

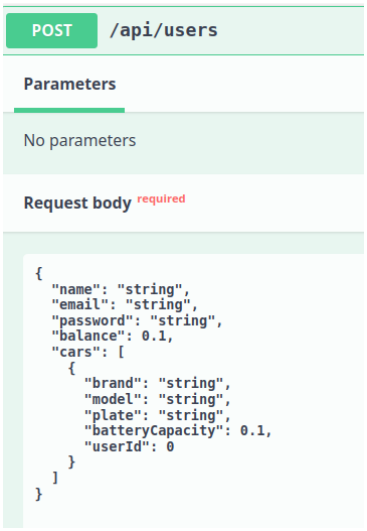


5 API for developers

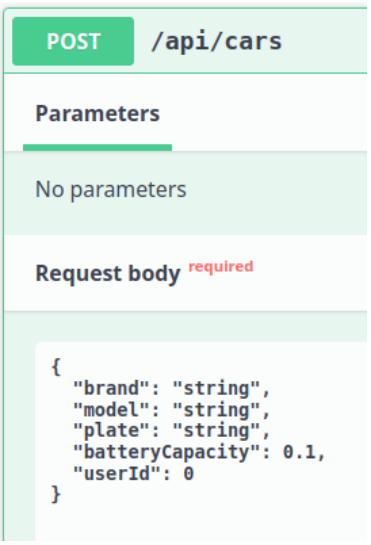
A API segue um modelo RESTful bem estruturado, para gerir utilizadores,carros,estações de carregamento, staff ,bookings e estatísticas relacionadas com o carregamento dos veículos elétricos. A api segue a standard OpenAPI 3.1.0 e foi documentada usando o Swagger UI (http://192.168.160.18:8080/swagger-ui/index.html). -> incluir link mesmo do deployed.

Os principais endpoints da api são:

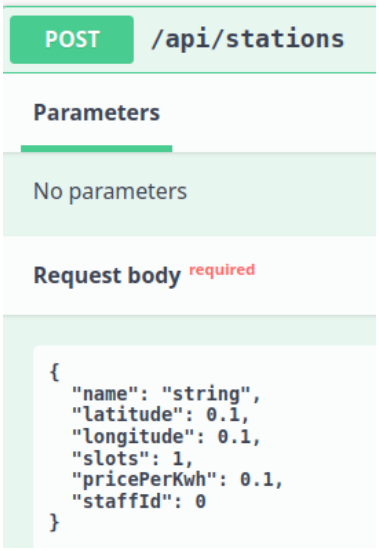
1. **Utilizadores (/api/users)**
   - GET /api/users – Listar todos os utilizadores
   - POST /api/users – Criar um novo utilizador
   - GET /api/users/{id} – Obter utilizador por ID
   - PUT /api/users/{id} – Atualizar utilizador
   - DELETE /api/users/{id} – Eliminar utilizador
   - PATCH /api/users/{id}/addFunds?amount=xx – Adicionar saldo ao utilizador
   - GET /api/users/{id}/balance – Consultar saldo do utilizador



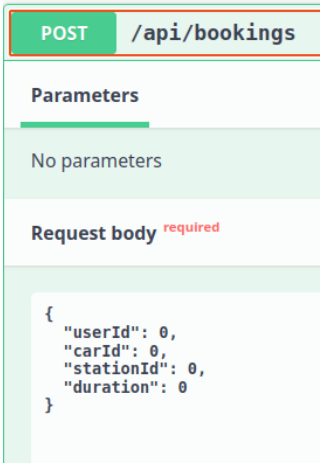
2. **Carros (/api/cars)**
    - GET /api/cars – Listar todos os carros
    - POST /api/cars – Registar um carro novo
    - GET /api/cars/{id} – Obter carro por ID
    - PUT /api/cars/{id} – Atualizar carro
    - DELETE /api/cars/{id} – Eliminar carro
    - GET /api/cars/user/{userId} – Listar carros de um utilizador



3. **Estações de carregamento (/api/stations)**
    - GET /api/stations – Listar todos as estações
    - POST /api/stations – Criar uma nova estação
    - GET /api/stations/{id} – Obter estação por ID
    - PUT /api/stations/{id} – Atualizar estação
    - DELETE /api/stations/{id} – Eliminar estação



4. **Reservas (/api/bookings)**
    - POST /api/bookings – Criar uma reserva
    - PATCH /api/bookings/{id}/status?status=COMPLETED|CANCELLED – Atualizar estado da reserva
    - GET /api/bookings/user/{userId} – Listar reservas de um utilizador



DELETE /api/users/{id}

DELETE /api/stations/{id}

DELETE /api/cars/{id}

GET /api/users/{id}

GET /api/stations/{id}

GET /api/cars/{id}

GET /api/users

GET /api/stations

GET /api/staffs

GET /api/cars

GET /api/users/{id}/balance

GET /api/stations/{stationId}/weekly-trends

GET /api/stations/{stationId}/rush-hour-stats

GET /api/stations/{stationId}/monthly-revenue

GET /api/staffs/{id}

GET /api/reservations/weekly-trends

GET /api/reservations/rush-hour-stats

GET /api/payments/monthly-revenue

GET /api/cars/user/{userId}

GET /api/bookings/user/{userId}

PATCH /api/users/{id}/addFunds

PATCH /api/bookings/{id}/status

POST /api/users

POST /api/stations

POST /api/stations/monthly-revenue

POST /api/staffs

POST /api/cars

POST /api/bookings

PUT /api/users/{id}

PUT /api/stations/{id}

PUT /api/cars/{id}