TQS: Product specification report

Diogo Oliveira [113 664], Bruno Charão [111 590], Hugo Sousa [112 733], Guilherme Mesquita [112 957]

v2025-04-25

[1 Introduction 1](#_Toc196497709)

[1.1 Overview of the project 1](#_Toc196497710)

[1.2 Known limitations 1](#_Toc196497711)

[1.3 References and resources 2](#_Toc196497712)

[2 Product concept and requirements 2](#_Toc196497713)

[2.1 Vision statement 2](#_Toc196497714)

[2.2 Personas and scenarios 2](#_Toc196497715)

[2.3 Project epics and priorities 2](#_Toc196497716)

[3 Domain model 3](#_Toc196497717)

[4 Architecture notebook 3](#_Toc196497718)

[4.1 Key requirements and constrains 3](#_Toc196497719)

[4.2 Architecture view 3](#_Toc196497720)

[4.3 Deployment view 3](#_Toc196497721)

[5 API for developers 3](#_Toc196497722)

# Introduction

## Overview of the project

Este projeto foi desenvolvido no âmbito da unidade curricular de **Teste e Qualidade de Software (TQS)** e tem como principal objetivo a aplicação prática de princípios e ferramentas de **Software Quality Assurance (SQA)** e **DevOps** ao longo do ciclo de vida de um projeto de software.

O produto em desenvolvimento, denominado **PlugPoint,** consiste numa **plataforma digital para gestão e utilização de estações de carregamento de veículos elétricos**. A aplicação serve dois grandes tipos de utilizadores: os condutores de veículos elétricos (que podem procurar, reservar e usar pontos de carregamento) e os operadores das estações (que gerem a disponibilidade e monitorizam a operação).

O produto também deve ser capaz de apresentar um mapa de estatísticas e dados relevantes para análise, sendo também importante implementar um sistema de pagamento, ainda que seja simulado.

A arquitetura da solução assenta em serviços web RESTful desenvolvidos em **Spring Boot**, com integração de pipelines de **CI/CD**, testes automatizados e monitorização de qualidade de código, segundo práticas ágeis de desenvolvimento.

## Known limitations

Durante o desenvolvimento da plataforma PlugPoint, identificaram-se as seguintes limitações conhecidas:

* **Integração com sistemas de pagamento real**: A funcionalidade de pagamento não foi implementada com a integração com gateways reais (ex: Stripe, MBWay), devido à limitação de tempo e complexidade legal associada a pagamentos online.
* **Push Notifications**: A funcionalidade de notificações push diretamente na app (mobile ou web) não foi implementada por falta de integração com serviços como Firebase Cloud Messaging.
* **Gestão avançada de frotas**: Algumas funcionalidades pensadas para gestores de frotas empresariais, como dashboards personalizados por veículo ou visualização agregada por centro logístico, ficaram fora do MVP. O controlo de utilizadores também não foi totalmente implementado, como na página AdminPage na secção Utilizador não aparecem informações necessárias ao Admin ou aos gestores.
* **Agendamento inteligente de carregamentos**: A sugestão de horários ideais com base em padrões de uso e previsões de procura não foi totalmente desenvolvida.
* **Modo offline / falhas de rede**: O comportamento da aplicação em caso de falhas de rede (ex: perda de ligação ao backend) não foi abordado detalhadamente.
* **Internacionalização (i18n)**: A aplicação está apenas disponível em português e não inclui suporte multilingue.

## References and resources

O desenvolvimento da plataforma PlugPoint envolveu a utilização de um conjunto abrangente de frameworks, bibliotecas, ferramentas de qualidade de software, práticas de DevOps e fontes de consulta técnica. Esta secção descreve os principais recursos utilizados que foram cruciais para a implementação, testes e manutenção da aplicação, e que poderão servir como base sólida para equipas futuras ou projetos semelhantes.

No lado do backend, a aplicação foi construída com base no **Spring Boot que** proporcionou uma estrutura robusta e produtiva para o desenvolvimento de serviços RESTful. Foram utilizados módulos como spring-boot-starter-web para a criação da API e spring-boot-starter-data-jpa para a ligação à base de dados, com persistência implementada através de JPA sobre **PostgreSQL**. A comunicação com esta base de dados foi suportada pelo driver JDBC e configurada em ambiente **Docker Compose**, permitindo simular facilmente um ambiente real em desenvolvimento e testes.

Para facilitar o desenvolvimento e manter o código limpo e conciso, foi incluída a biblioteca **Lombok**, que reduz a necessidade de código repetitivo. A documentação da API foi gerada automaticamente com recurso ao **Springdoc-OpenAPI**, oferecendo uma interface Swagger acessível via browser. A nível de testes, foram integradas ferramentas como **JUnit 5**, **Mockito** e **AssertJ**, garantindo a cobertura de testes unitários e de integração. Além disso, **Testcontainers** foi utilizado para criar instâncias reais de PostgreSQL nos testes, assegurando realismo sem comprometer a automação.

Foi também adotada uma abordagem orientada a comportamento (BDD) com **Cucumber JVM**, permitindo a definição de testes legíveis e alinhados com critérios de aceitação. A camada de testes end-to-end (E2E) foi implementada com **Selenium ,** utilizando **WebDriverManager** para facilitar a gestão dos drivers de browser. Os resultados destes testes foram integrados automaticamente na plataforma **Xray**, através de pipelines GitHub Actions, permitindo um controlo eficaz de qualidade.

Do lado do frontend, a aplicação foi desenvolvida em **React** , com o sistema de build ultrarrápido **Vite**. A navegação entre páginas foi gerida com **React Router DOM.**  Para representar geograficamente as estações de carregamento, foi usada a biblioteca **React Leaflet**, baseada no Leaflet 1.9. A visualização de estatísticas e métricas operacionais foi suportada por **Recharts**, que permitiu construir gráficos interativos, e as animações subtis da interface foram conseguidas com **Framer Motion**. A manipulação de datas e horas no frontend foi simplificada com **date-fns**, uma biblioteca leve e moderna. A qualidade e consistência do código JavaScript/TypeScript foram asseguradas com **ESLint**, configurado com regras específicas para React e React Hooks.

Em termos de DevOps e qualidade, o projeto utilizou **GitHub Actions** para implementar pipelines CI/CD completas que incluíam linting, execução de testes, análise de qualidade com **SonarCloud**, build de imagens Docker e publicação automatizada. Todo o projeto foi containerizado, com o backend, base de dados e frontend orquestrados em conjunto com **Docker Compose**, facilitando a replicação do ambiente de produção. A documentação da API REST foi disponibilizada automaticamente via Swagger, e os testes de BDD e E2E foram ligados ao sistema de gestão de testes Xray, integrando os resultados diretamente no ciclo de desenvolvimento.

O ambiente de desenvolvimento foi uniformizado com **Java,** **Node JS** e ferramentas como Maven, Vite CLI e IDEs como IntelliJ IDEA e VS Code. Além das tecnologias aplicadas, a equipa recorreu frequentemente à documentação oficial de cada tecnologia e a plataformas como **Baeldung**, **Stack Overflow**, e os sites oficiais de Spring, React, Leaflet, Recharts e Testcontainers. Estas fontes serviram como suporte constante ao longo do desenvolvimento e resolução de desafios técnicos.

Em resumo, o projeto PlugPoint assentou numa stack tecnológica moderna, alinhada com boas práticas de engenharia de software e qualidade, suportada por ferramentas de automação, testes e documentação, garantindo um desenvolvimento ágil, testável e escalável.

# Product concept and requirements

## Vision statement

A plataforma PlugPointvisa resolver o problema do isolamento **dos serviços de carregamento elétrico**, oferecendo uma solução unificada para pesquisa, reserva e utilização de estações de carregamento em redes distintas.

Os principais objetivos funcionais da aplicação incluem:

* Permitir que os **condutores**:
* Pesquisem estações de carregamento por localização e tipo
* Reservem slots de carregamento disponíveis
* Desbloqueiem e iniciem carregamentos
* Consultem o histórico de carregamentos e estatísticas de consumo
* Consultem o planeador de viagem para sugestões de paragens destinadas a carregamento
* Recebam notificações via email/SMS a informar o condutor do inicio da reserva
* Guardar estações de preferência nos favoritos e consultar histórico de estações utilizadas
* Permitir que os **operadores de estações**:
* Registem e configurem as estações e os respetivos pontos de carregamento
* Atualizem o estado e disponibilidade dos carregadores
* Monitorizem consumos, reservas e estatísticas em tempo real
* Recolham relatórios em formato CSV para discussão com superiores.

A plataforma incluirá ainda um sistema de **integração com pagamentos** (simulado inicialmente), e dashboards para **visualização de dados agregados**, com possibilidade de expansão para funcionalidades ecológicas (ex: poupança de CO₂) ou sincronização de recursos.

## Personas and scenarios

## **2.2.1 Persona 1: João Ferreira – Condutor de veículo elétrico**

**Nome**: João Ferreira   
 **Idade**: 35 anos  
 **Género**: Masculino  
 **Nacionalidade**: Portuguesa  
 **Residência**: Braga, Portugal  
 **Estado civil**: Casado, 1 filha de 5 anos  
 **Profissão**: Engenheiro de software numa fintech  
 **Educação**: Mestrado em Engenharia Informática  
 **Rendimento**: Médio-alto  
 **Estilo de vida**: Sustentável, preocupado com o ambiente  
 **Tecnologia**: Utiliza aplicações de mobilidade, home automation e tem um carro elétrico (Nissan Leaf)

**Objetivos**:

* Carregar o veículo sem complicações, com previsibilidade e segurança
* Planear viagens familiares com carregamentos garantidos
* Monitorizar consumo e ganhos ambientais

**Frustrações**:

* Chegar a um ponto de carregamento e encontrá-lo ocupado ou avariado
* Falta de visibilidade sobre custos ou horários de menor procura

### **Cenário 1 – João usa o planeador de viagem e favoritos**

João está a planear uma viagem em família até ao Algarve. Entra na aplicação **PlugPoint** no telemóvel e insere o destino. A aplicação sugere automaticamente duas estações de carregamento rápidas ao longo do trajeto. João reserva um slot numa delas, perto de Évora, para as 12h30.

Antes de sair, recebe uma notificação por SMS a lembrar o início da reserva. Ao chegar, desbloqueia o carregador diretamente na app. Durante o carregamento, consulta o histórico dos últimos meses e fica satisfeito ao ver a poupança de 210 kg de CO₂. No fim, adiciona a estação aos seus favoritos para futuras viagens.

### **Cenário 2 – João tenta o pagamento e alerta de falha**

João termina um carregamento num ponto rápido numa estação nova. No fim da sessão, recebe uma notificação com o valor estimado a pagar (modo simulado). No dia seguinte, tenta reservar novamente essa estação, mas recebe um alerta de que está **temporariamente indisponível devido a avaria**. Agradece a fiabilidade da informação, e reserva noutra próxima.

---------------------------------------------------------------------------------------------------------------------------

**2.2.2 Persona 2: Marta Lopes – Operadora de rede de carregamento**

**Nome**: Marta Lopes  
 **Idade**: 42 anos  
 **Género**: Feminino  
 **Nacionalidade**: Portuguesa  
 **Residência**: Porto, Portugal  
 **Estado civil**: Divorciada, sem filhos  
 **Profissão**: Gestora de operações na empresa **EnerGrid**  
 **Educação**: Licenciatura em Engenharia Eletrotécnica  
 **Rendimento**: Médio  
 **Estilo de vida**: Organizada, prática, com foco em eficiência e resultados  
 **Tecnologia**: Utiliza ferramentas de gestão (CRM, dashboards), tem literacia digital elevada

**Objetivos**:

* Monitorizar o desempenho das estações sob sua gestão
* Minimizar falhas e indisponibilidades
* Aplicar estratégias para reduzir os horários de pico

**Frustrações**:

* Não conseguir prever a ocupação
* Ter de responder a reclamações por estações em falha sem aviso prévio

### **Cenário 1 – Marta usa o painel de controlo e gestão de disponibilidade**

Marta inicia a sua rotina no portal web da **PlugPoint**. No dashboard, identifica que uma das estações na zona da Boavista apresenta uma taxa de falha acima do normal. Decide colocar dois dos quatro carregadores em modo de manutenção e agenda uma intervenção técnica para o dia seguinte.

Aproveita para analisar a ocupação média nas últimas duas semanas e repara que o uso baixa significativamente entre as 14h e as 17h. Atualiza as tarifas nesses horários, oferecendo um desconto de 30%, e regista o impacto no dashboard de KPIs para avaliar nos próximos dias.

No final do dia, exporta um relatório em CSV com o consumo energético das 12 estações sob sua gestão, a ser usado numa reunião mensal.

### **Cenário 2 – Detecção e gestão de incidente**

Marta recebe uma notificação automática da plataforma indicando que uma estação em Aveiro teve 3 tentativas de carregamento falhadas nas últimas 2 horas. Acede ao painel de falhas, coloca a estação em modo de **manutenção com causa identificada**, e cria uma tarefa para a equipa técnica. Ativa uma mensagem de alerta visível aos utilizadores da aplicação PlugPoint.

----------------------------------------------------------------------------------------------------------------

### Persona 3: Ricardo Almeida – Responsável de Frota Empresarial

**Nome:** Ricardo Almeida

**Idade:** 47 anos

**Género:** Masculino

**Residência:** Lisboa, Portugal

**Estado civil:** Casado, 2 filhos adolescentes

**Profissão:** Gestor de frota na empresa de logística EcoTrans

**Educação:** Licenciatura em Gestão

**Rendimento:** Médio-alto

**Estilo de vida:** Pragmático, orientado para resultados, defensor da transição energética

**Tecnologia:** Utiliza software de gestão de frotas, dashboards, ferramentas de reporte automatizado

### **Objetivos**

* Garantir que os veículos da frota estão carregados e operacionais diariamente
* Minimizar custos com carregamentos e tempos mortos
* Monitorizar consumos energéticos e impacto ambiental
* Automatizar o agendamento de carregamentos por turnos

### **Frustrações**

* Dificuldade em reservar múltiplos carregamentos em simultâneo
* Falta de relatórios agregados por veículo e por período
* Atrasos causados por carregadores ocupados ou avariados sem aviso prévio

### **Cenário 1 – Gestão semanal da frota com reservas automatizadas**

Ricardo gere uma frota de 12 carrinhas elétricas usadas para entregas urbanas. Através da versão web da PlugPoint, acede ao painel de controlo empresarial. Utiliza a funcionalidade de **reserva em lote**, onde define que todos os veículos devem estar prontos para sair às 8h. A aplicação sugere janelas de carregamento durante a noite em três estações próximas com múltiplos pontos disponíveis.

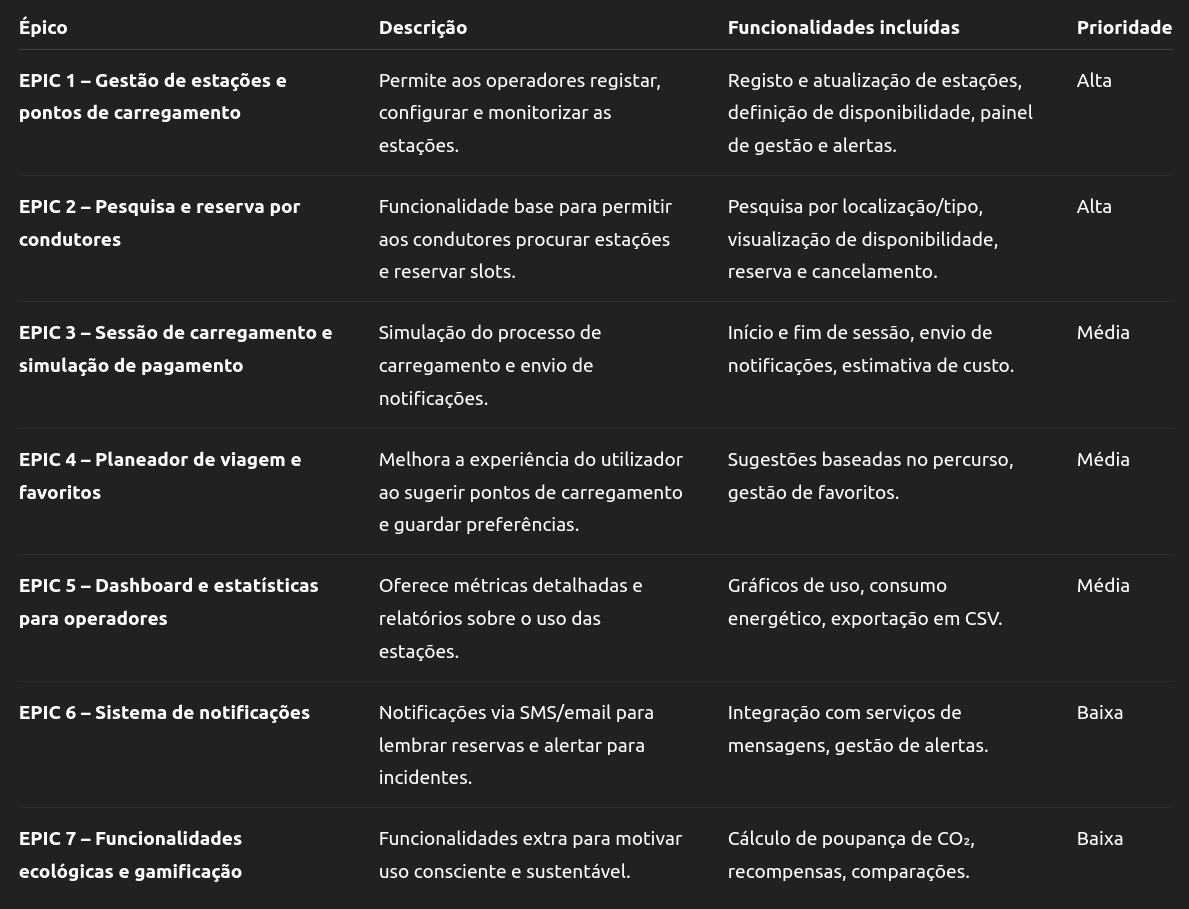
Ricardo recebe um **resumo automático em PDF** todas as manhãs com o estado de carregamento de cada veículo. Ao identificar que duas carrinhas não carregaram devido a falhas no carregador, agenda manutenção preventiva através do suporte integrado.

### **Cenário 2 – Análise de custos e relatório ambiental trimestral**

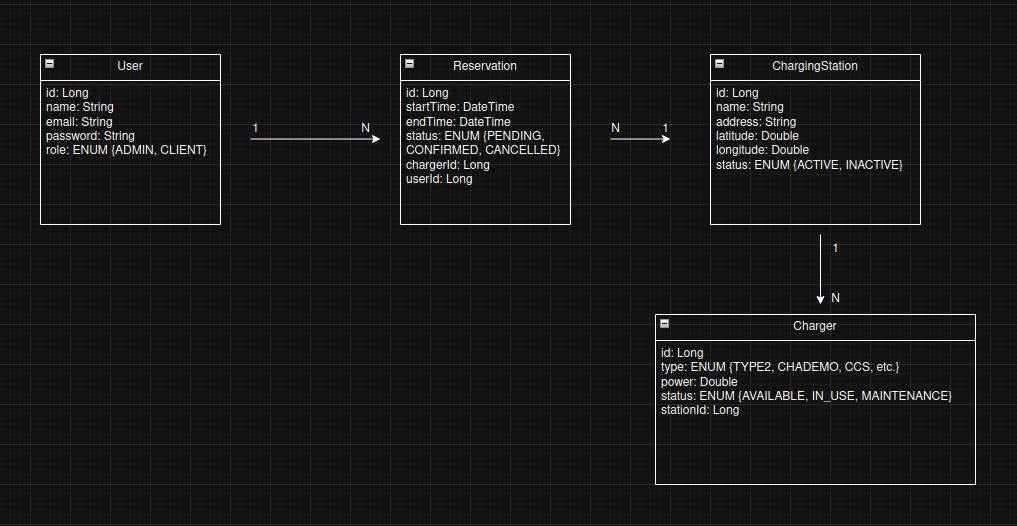
No final do trimestre, Ricardo utiliza a funcionalidade de **exportação de relatórios por veículo**, com detalhes sobre consumo, custos estimados e emissões evitadas. Apresenta os dados à direção como parte da estratégia ESG (Environmental, Social, Governance). Através de filtros na plataforma, consegue visualizar o **pico de carregamentos** e decide ajustar os horários para evitar sobrecarga nos carregadores entre as 17h e 20h.

## Project epics and priorities

O desenvolvimento da aplicação PlugPoint foi organizado em múltiplas iterações, seguindo uma abordagem incremental. Os principais epics foram definidos de acordo com as funcionalidades críticas para os dois tipos principais de utilizadores — condutores e operadores — e priorizados com base no impacto funcional e dependências técnicas.



# Domain model



# Architecture notebook

## Key requirements and constrains

## Requirements

* **Multiplataforma e Interface Responsiva**  
   O sistema deve estar disponível numa **plataforma web** acessível em computadores, tablets e telemóveis. Este requisito influenciou a escolha de **React** para o frontend devido à sua flexibilidade de componentes reutilizáveis e responsividade.
* **Integração com Sistemas Externos (em potencial)**  
   Embora atualmente o sistema funcione com base de dados local/PostgreSQL, existe **potencial de integração com sistemas de autenticação centralizada (SSO)** ou APIs públicas como a da **Mobi.E**, que pode influenciar decisões de desacoplamento da lógica de negócio e autenticação.
* **Separação de responsabilidades e escalabilidade**  
   O backend foi desenvolvido com **Spring Boot** e segue uma arquitetura **em camadas (Controller-Service-Repository)**, de forma a isolar dependências, facilitar manutenção e testes, e preparar o sistema para escalabilidade horizontal (e.g., Docker, microserviços).

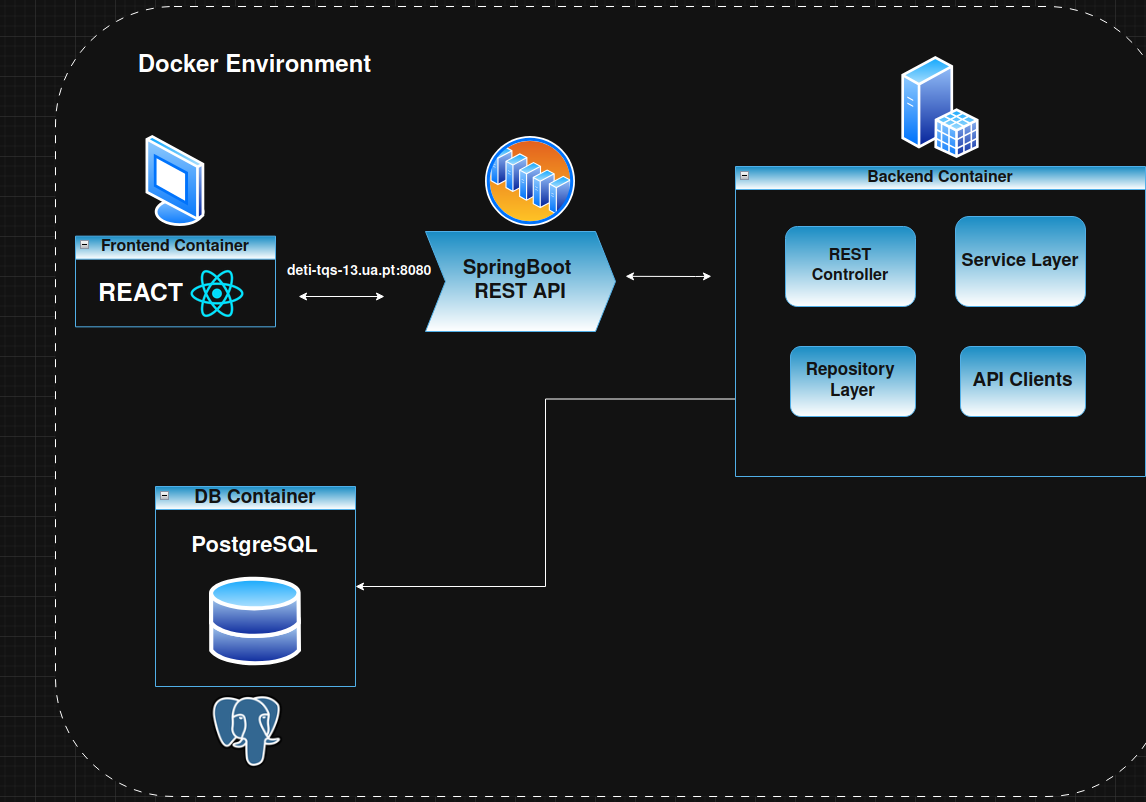
**Characteristics**

* **Disponibilidade -** Como se trata de um sistema de reservas de carregamento, **interrupções não são aceitáveis** durante horas de pico. A infraestrutura deve suportar alta disponibilidade.
* Desempenho e eficiência - O sistema deve **responder rapidamente a ações do utilizador** (reservas, cancelamentos, stats). A base de dados e endpoints foram otimizados com queries diretas por ID.
* Usabilidade - A experiência do utilizador é fundamental. Foi usado um **design moderno (UI Neon/Minimal)**, com tooltips, animações e organização visual clara.
* Extensibilidade - O sistema foi desenhado para **permitir fácil introdução de novos tipos de carregadores e estações**
* **Manutenção e testabilidade -** A estrutura modular com separação clara entre lógica de negócio e acesso a dados **facilita testes unitários, de integração e de aceitação**.A estrutura modular com separação clara entre lógica de negócio e acesso a dados **facilita testes unitários, de integração e de aceitação**.
* Portabilidade - O projeto é **dockerizável e portátil**, podendo ser facilmente movido para cloud ou integrado com CI/CD pipelines (GitHub Actions, GitLab CI).
* Observabilidade - Logs, notificações e estatísticas são registadas com base em reservas e sessões, o que permite análises futuras (e.g., gráficos de CO₂, utilização, etc)

## Architecture view



## Deployment view



# API for developers

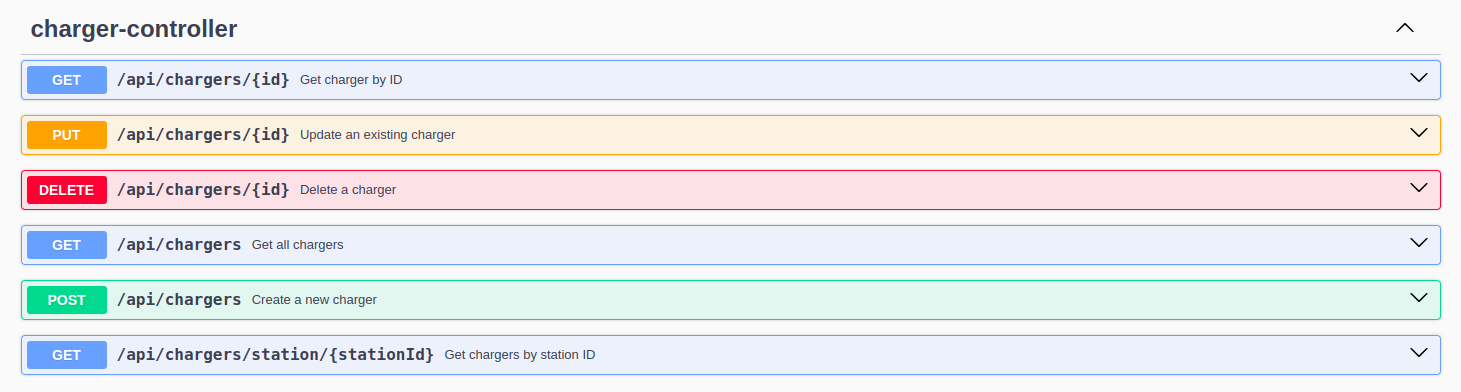
**Visão Geral**

A API do PlugPoint segue um design RESTful, com endpoints projetados para interagir com os principais recursos do sistema. Cada recurso (como **estações de carregamento**, **reservas**, **carregadores**, etc.) tem endpoints dedicados para a criação, leitura, atualização e exclusão de dados, respeitando as práticas recomendadas para APIs REST.

A API é estruturada de forma que cada recurso seja identificado por um URI único, e as interações com esses recursos sejam realizadas usando os métodos HTTP apropriados (GET, POST, PUT, DELETE). Para facilitar o consumo, a API também fornece documentação interativa via **Swagger**, permitindo que os desenvolvedores testem e explorem os endpoints diretamente.

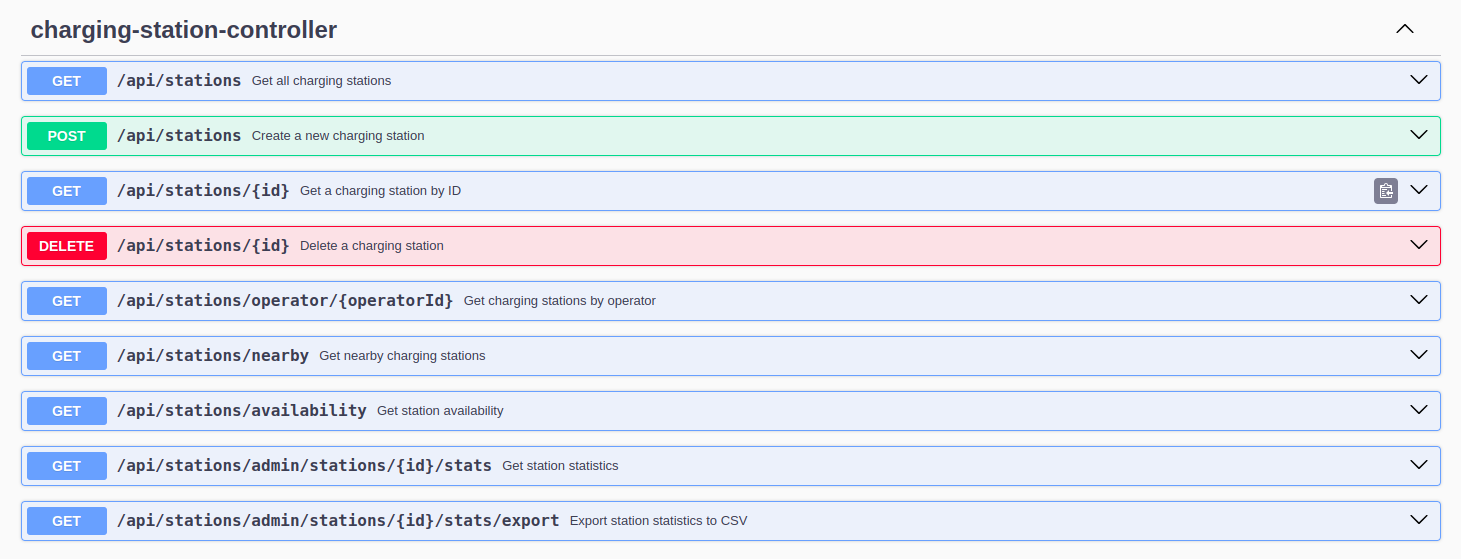
* **Chargers (Carregadores)**

Um **carregador** é um ponto específico dentro de uma estação onde um veículo elétrico pode ser carregado. Os carregadores podem ter diferentes tipos (ex: **TYPE2**, **CHADEMO**, **CCS**, etc.), potências e status (disponível, em uso, manutenção). Os carregadores estão sempre associados a uma **estação de carregamento**.



* **Charging Stations (Estações de carregamento)**

Uma **estação de carregamento** representa um ponto físico onde os condutores podem carregar seus veículos elétricos. Cada estação pode ter múltiplos carregadores associados a ela. Além disso, uma estação pode ter várias informações de localização, como **nome**, **endereço** e **coordenadas geográficas**.

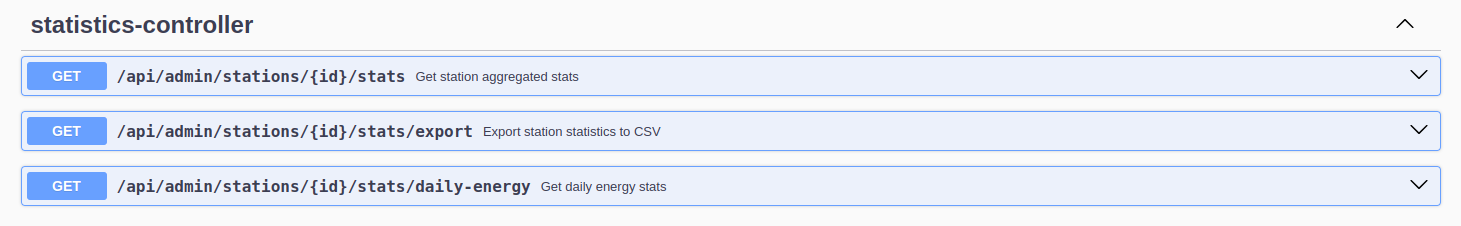


* **Reservations (Reservas de carregamentos)**

Uma **reserva** permite que um condutor de veículo elétrico reserve um carregador em uma estação de carregamento. A reserva contém informações sobre o carregador, o horário de início e fim, e o status da reserva (confirmada, cancelada.



* **Statistics (Estatisticas)**



O Swagger UI da aplicação está disponível de duas maneiras: quando a aplicação está a correr localmente, pode ser acedido via <http://localhost:8080/swagger-ui/index.html>, estando disponível apenas para a máquina onde está a ser executada. Quando a aplicação está a correr na máquina virtual (VM), o Swagger UI pode ser acedido através de <http://deti-tqs-13.ua.pt:8080/swagger-ui/index.html>, precisa estar conectado à rede Eduroam ou à VPN da universidade para que o acesso funcione corretamente, pois o servidor está restrito a essas redes.