TQS: Product specification report

Carolina Prata [114246], Diogo Domingues [114192], João Varela [113780]
v2025-06-09

1 Introduction

1.1 Overview of the project

This project was developed within the scope of the TQS (Teste e Qualidade de Software) course, where the main objective is to apply principles of quality-driven software design and validation to a realistic product development scenario. The assignment focuses on translating user needs into functional requirements, structuring a modular and testable system, and ensuring the implementation of high-value features through well-defined user stories and acceptance criteria.

Our application, **VoltConnect**, is a web-based platform designed to streamline and enhance the electric vehicle (EV) charging experience for both end-users and infrastructure operators. The core aim is to tackle common frustrations experienced by EV drivers, such as difficultyto find available chargers, unreliable stations, or lack of route planning tools, while simultaneously providing backoffice capabilities for station operators to manage their infrastructure.

VoltConnect is a web platform that simplifies the EV charging process. It helps **drivers** find nearby stations, filter by charger type and availability, book time slots, start and stop sessions remotely and pay digitally.

For station operators, the platform provides tools to register and manage their stations, including the ability to create, edit, or delete chargers. They can also view reviews on their stations to help improve the quality of their service.

By focusing on real user needs like route planning and high-frequency charging, VoltConnect offers a unified, reliable experience for both everyday drivers and infrastructure managers.

1.2 Known limitations

One of the known limitations of the project is the absence of CO_2 emission tracking per charging session. This feature was initially considered to raise environmental awareness among users, but it was not implemented due to time constraints and prioritization of core functionalities. Additionally, charger status management such as marking a charger as "under maintenance" or "out of service" was not included in the current version. This was planned as part of the station operator tools to improve transparency and user experience.

1.3 References and resources

https://go-tou.com/pt/news/sustainable-ev-charging-how-advanced-reservation-technology-for-ev-chargers-is-game-charging
https://www.edp.pt/particulares/servicos/mobilidade-eletrica/app-edp-charge/

2 Product concept and requirements

2.1 Vision statement

VoltConnect is a web-based platform designed to unify and simplify the electric vehicle (EV) charging experience for both drivers and charging station operators. The platform addresses the growing need for **accessible, reliable, and intelligent EV charging infrastructure** in urban and interurban environments.

Our project focuses on reducing friction in daily EV use whether for commuting, commercial driving (e.g., ride-hailing), or long-distance travel by offering an integrated set of tools to plan, book, and manage charging sessions efficiently.

VoltConnect solves the **fragmentation and inefficiency** of current EV charging solutions by providing:

- **Station Discovery**: A map interface where EV drivers can locate available charging stations, apply filters (e.g., connector type, availability), and view detailed station data.

- **Slot Booking & Scheduling**: Users can reserve a charging slot at a specific time, ensuring availability on arrival and avoiding queueing or wait times.

- **Charging Session Management**: Drivers can remotely monitor real-time charging progress (kWh, time remaining), and receive summaries upon completion.

- **Digital Payments & Charging History**: After a session ends, users can securely pay through the platform and access a history of their past sessions, including costs.

- **Station Management**: Station operators use a dedicated backoffice interface to register/edit stations, mark them under maintenance, and monitor performance metrics such as occupancy and energy output.

Some features originally envisioned such as **route-based charging planning** and **community-driven station ratings** were **deprioritized for the MVP** and planned for later stages. The decision was based on development constraints and the focus on core functionality for everyday users (e.g., Miguel and António, the commuter and ride-hailing personas).

VoltConnect draws inspiration from popular EV charging platforms like **Chargemap**, **PlugShare**, and **Ionity**, but differentiates itself by combining real-time station availability, **slot booking**, and **remote control of sessions** in a single, seamless workflow. Unlike static station finders, VoltConnect emphasizes interactivity, personalization, and proactive planning tools for both casual and high-frequency users.

The system concept was developed iteratively through:

- **Persona creation** (e.g., Miguel, Ana, António, Carla), covering multiple user types with distinct goals;
- Definition of **user stories** and **acceptance criteria** for each feature, mapped to business goals;
- Regular feedback sessions with TQS course instructors and peers, which helped refine the scope and feature priorities;
- Internal design sprints where feature feasibility, user experience, and testing strategy were evaluated collaboratively.

2.2 Personas and scenarios

To guide the design and development of the platform, we defined several representative personas, each reflecting a specific type of user interacting with the system. These personas help us understand user goals, frustrations, and expectations, ensuring that our solution aligns with real-world needs.

Personas

Miguel - EV Driver (34 years old, Software Engineer)

Miguel is a 34-year-old software engineer who switched to an electric vehicle (EV) two years ago. He uses his EV daily for commuting to work and occasionally for weekend trips. Although he enjoys the EV driving experience, he often feels frustrated by the fragmented nature of charging networks. To simplify his routine, Miguel is looking for a unified platform that allows him to manage all aspects of EV charging in one place. His main needs include finding available and compatible charging stations, booking time slots in advance, starting and stopping charging sessions remotely, paying securely through the platform, and reviewing past sessions with details on energy usage and costs. Miguel's goals are to save time by quickly locating chargers, reduce expenses by using off-peak discounts, track his energy consumption to optimize his routine, and benefit from a smooth, integrated charging experience.

António – EV Driver (28 years old, Ride-Hailing Driver)
António is a 28-year-old ride-hailing driver working full-time in the Lisbon metropolitan area. He relies entirely on his electric vehicle to earn a living, completing dozens of rides every day. Given the intensity of his schedule, António needs to charge his vehicle multiple times per day and cannot afford delays caused by unavailable or malfunctioning stations. He depends on a platform that allows him to quickly locate nearby stations with available chargers, reserve time slots in advance, view real-time charger status to avoid wasted trips, and track his charging costs. He also values peer reviews that help him choose reliable stations. António's main goals are to maximize uptime, minimize energy costs by taking advantage of off-peak hours, ensure fast and dependable charging, and maintain smooth operations throughout the day.

Ana – EV Driver (39 years old, Road Trip Traveler)
Ana is a 39-year-old high school teacher who enjoys weekend road trips and family vacations with her electric vehicle. She is very mindful of the vehicle's battery range and prefers to plan all her routes in advance, ensuring that each charging stop is well-placed and trustworthy. Ana needs to visualize charging stations along her planned route, filter for fast chargers to reduce downtime, and access ratings and reviews to ensure the station is safe and functional. She also benefits from being able to reserve charging slots during key rest breaks. Her goals are to travel long distances confidently without range anxiety, avoid getting stranded in underserved areas, coordinate charging with rest stops to save time, and provide a stress-free travel experience for her family.

Bruno – Technician (29 years old)
Bruno is a 29-year-old technician responsible for monitoring and expanding the charging infrastructure. He oversees the performance of specific stations assigned to him and uses the platform to track how well they are serving users. Bruno regularly checks the chargers under his responsibility, paying close attention to user reviews. When he notices recurring complaints, signs of overload, or high demand in specific time slots, he evaluates the need to add new chargers to that station. His tasks involve reviewing operational data, analysing feedback, and making decisions to improve service availability. Bruno's goals are to ensure that all chargers remain functional and responsive to demand, proactively address user pain points, and contribute to the strategic expansion of the charging network.

Sofia – Platform Administrator (38 years old)
Sofia is a 38-year-old administrator working at VoltConnect headquarters. She has full access to the platform and oversees all charging stations, including their chargers and user reviews. Sofia monitors global activity across the network, ensuring that stations are performing as expected and identifying patterns that may indicate operational issues or areas for improvement. She uses the platform to view aggregated statistics such as usage frequency, charger availability, and customer satisfaction. Reviews left by users are a key input for her when auditing service quality and making decisions about platform improvements or operator support. Sofia's goals are to maintain high reliability across the network, ensure consistent service quality, support operators in optimizing their stations, and use data insights to guide strategic development.

Main Scenarios

Miguel – EV Driver (Software Engineer)
- Quickly locating and booking a charger near work
    - During a busy weekday, Miguel opens the app before heading to the office. He filters the map to show fast chargers near his company, checks availability, and books a slot to charge during lunch break.

António – EV Driver (Ride-Hailing Driver)
- Avoiding downtime between rides

    - While driving in Lisbon, António sees that his battery is running low. He opens the app, filters for fast and available chargers, and instantly reserves a slot at the closest station to avoid waiting in line.
- Using reviews to avoid unreliable stations
    - After encountering issues at a previous station, António reads recent reviews from other drivers before selecting a charger. He chooses a well-rated station with high reliability to ensure smooth operations for the rest of his day.

Ana – EV Driver (Road Trip Traveler)
- Planning a weekend route with charging stops
    - Before leaving for a road trip, Ana uses the platform to visualize chargers along her planned route. She selects stations near rest areas and reserves slots that match her estimated arrival times.
- Avoiding range anxiety during travel
    - While driving, Ana notices her battery dropping faster than expected. She opens the app to check nearby chargers and filters for fast ones. Thanks to the map and availability indicators, she makes a safe stop without delay.

Bruno – Technician
- Expanding a station based on user feedback
    - Bruno reviews weekly reports and notices that one station frequently receives complaints about long waits. He checks usage patterns and recommends adding a new charger to address the demand.

Sofia – Platform Administrator
- Auditing low-rated stations
    - Sofia reviews the latest platform-wide statistics. She checks their reviews and performance data to identify patterns and flags the most critical ones for operator follow-up.

2.3 Project epics and priorities

To ensure an incremental and manageable development process, the project was structured around several key epics, each focusing on a core area of the platform. These epics were distributed across multiple sprints, with the objective of delivering meaningful value at each stage while maintaining a clear path toward the full solution.

The **Station Discovery epic (15/05 – 04/06)** focused on enabling users to locate available charging stations through a map interface, with support for filtering by charger type and availability. This was a foundational feature, as it supports most user interactions.

Next, **Slot Booking (21/05 – 04/06)** allowed users to reserve a charging time at a selected station, helping reduce waiting times and ensuring a smoother experience, especially for high-frequency users like ride-hailing drivers.

Charging Session Management (22/05 – 04/06) introduced the ability to view live status, and process payments. This epic brought together real-time interaction between the user and charger.

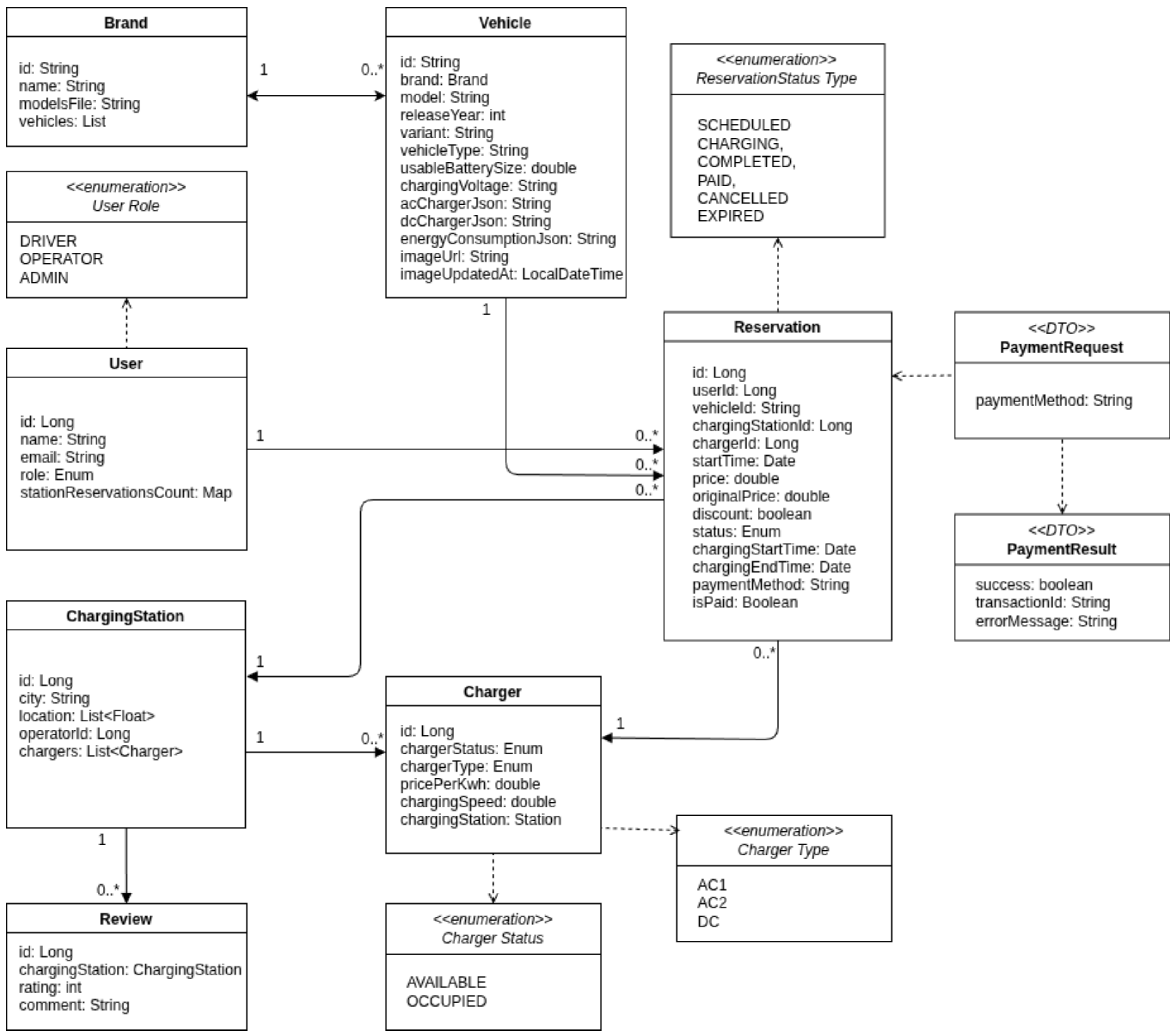
On the operator side, **BackOffice Management (30/05 – 04/06)** provided tools for registering stations, updating charger details, and managing availability.

The **Route-Based Charging Planning epic (30/05 – 04/06)** enabled users to visualize and plan charging stops along a predefined route, a feature especially useful for long-distance drivers like Ana.

Finally, **Community Feedback & Trust (30/05 – 04/06)** introduced the review system, allowing users to rate and comment on stations, contributing to transparency and informed decision-making.

These epics were prioritized to support the most common user flows first (discovery and booking), then expand into operational tools and value-added features like planning and feedback. The Jira timeline reflects this progression, allowing overlapping work while respecting dependencies between features.

3 Domain model



4 Architecture notebook

4.1 Key requirements and constrains

We recognized several essential architectural drivers and characteristics that need to be carefully addressed to ensure the system is designed and implemented effectively.

Through integration with OpenStreetMap, the application can accurately provide users with the real-time location of nearby charging stations.

For secure authentication and authorization, we will integrate JWT tokens that users must present when making authenticated API requests.

Role-based access control (RBAC) will be implemented to manage permissions and access levels for different user roles, including Drivers, Technicians, and Administrators.

To ensure consistent deployment in all environments, the backend, frontend, and database are packaged in Docker containers. These containers include everything needed to run the app, so it works the same on a developer's computer, a staging server, or in production. This makes deployment easier and reduces environment-related issues.

4.2 Architecture view

The architecture of the system follows a layered and modular design, combining elements of microservices and service-oriented architecture (SOA). The system is logically divided into four main functional layers: the Presentation Layer, the Application Layer, the Data Layer, and the External Integration Layer.

1. Presentation Layer

- Frontend Module
  - A web-based user interface built with React and styled using TailwindCSS.
  - Delivered via a frontend Docker container.
  - Communicates with backend services via RESTful APIs through a reverse proxy.

2. Application Layer (Backend Services)

Hosted in a Spring Boot backend container, the core business logic is distributed across several independent services:

- UserService – Handles user registration, authentication, and role management.
- VehicleService – Manages information related to electric vehicles owned by users.
- ReservationService – Manages booking and scheduling of charging sessions.
- ReservationMaintenanceService – Periodically validates and cleans up outdated reservations.
- ChargingStationService – Handles station discovery, availability, and location data.
- ChargerService – Manages individual charging units and their status.
- PaymentService – Processes payments related to charging sessions (planned).
- ReviewService – Allows users to leave feedback on stations or chargers.
- BrandService – Manages EV brands and models used within the system.
- StartupService – Initializes application data and configurations at boot.

3. Data Persistence Layer

- PostgreSQL Database
    - Hosted in its own Docker container.
    - Accessed via Spring Data JPA Repositories, which provide abstraction over SQL operations.

4. External Integration Layer

- The system integrates with two internet-based APIs:
    - OpenStreetMap API – Provides geolocation and mapping services for charging station display.
    - Open EV Data API – Supplies external EV-related datasets (e.g., vehicle specs, connector types).

5. Monitoring and Observability Layer

- Grafana Monitoring Stack
    - Prometheus collects metrics from all backend services.
    - Loki aggregates logs for real-time and historical analysis.
    - Grafana provides dashboards and alerting features to monitor system health and performance.

docker
Docker Environment
Frontend Docker Container
Backend Docker Container (Spring Boot)

4.3 Deployment view

To maintain consistency across development, testing, and production environments, the system will be deployed using five main Docker containers, each encapsulating a distinct component of the application. These containers will run on a virtual machine (VM) and interact with one another internally:

1. User Interface Container

- Purpose: Serves the web application built with React and Vite.
- Functionality: Provides the user interface to end users and handles user-initiated actions by making HTTP requests to the backend API.
- Port: Exposed on port 5000.

2. Backend Container

- Purpose: Hosts the Spring Boot application.
- Functionality: Processes all business logic, communicates with the PostgreSQL database, and integrates with external APIs (e.g., OpenStreetMap, EV data).
- API Access: Exposes RESTful endpoints over HTTP.
- Port: Runs on port 8080.

3. Database Container

- Purpose: Runs the PostgreSQL database instance.
- Functionality: Stores all persistent application data, including users, reservations, reviews, and vehicle data.
- Access: Not exposed externally only accessible by the backend container for security.

4. Visualization Container – Grafana

- Purpose: Visualizes metrics collected by Prometheus.
- Functionality: Connects to Prometheus as a data source and provides interactive dashboards to monitor system health, such as CPU usage, request latency, and database connections.
- Port: Available on port 3000, offering a web interface for developers and operators.

Together, these containers provide a clean separation of concerns, ensure environment parity, and simplify deployment, monitoring, and maintenance of the application.

5 API for developers

The backend API of this system follows a RESTful architecture, with a resource-oriented design. Each controller manages a specific type of resource (e.g., users, vehicles, stations), and exposes endpoints that allow standard CRUD operations using HTTP methods such as GET, POST, PUT, and DELETE.

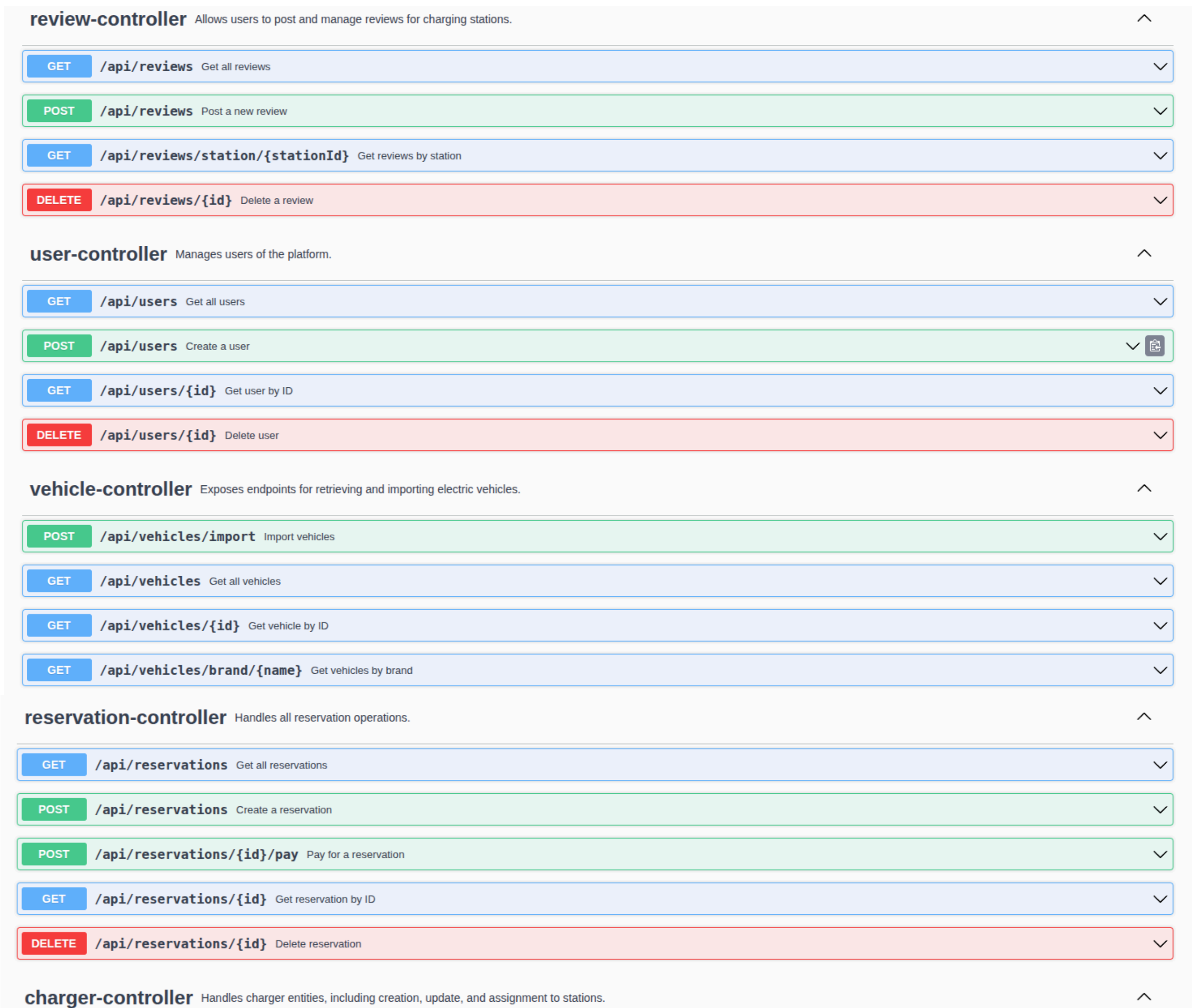
The API is organized in a modular way, with separate controllers for each resource domain. This ensures a clean separation of concerns and improves maintainability and scalability.

All endpoints are documented using Swagger (OpenAPI Specification), which provides a user-friendly interface for exploring and testing the API. The documentation is hosted within the application itself and is automatically generated based on annotations in the source code. This guarantees that the documentation remains synchronized with the implementation.

The design of the API respects best practices for REST APIs, including:

- Use of meaningful and plural nouns for resource paths (e.g., /api/vehicles, /api/stations)
- HTTP methods that clearly reflect the action being performed
- Path parameters for specific resource access (e.g., /api/vehicles/{id})
- Clear status codes and response structures

This structure ensures the API is intuitive for developers, supports easy client integration, and is prepared for future extensions.
Available on: http://deti-tqs-04.ua.pt/swagger-ui/index.html#/



review-controller Allows users to post and manage reviews for charging stations.

GET /api/reviews Get all reviews
POST /api/reviews Post a new review
GET /api/reviews/station/{stationId} Get reviews by station
DELETE /api/reviews/{id} Delete a review

user-controller Manages users of the platform.

GET /api/users Get all users
POST /api/users Create a user
GET /api/users/{id} Get user by ID
DELETE /api/users/{id} Delete user

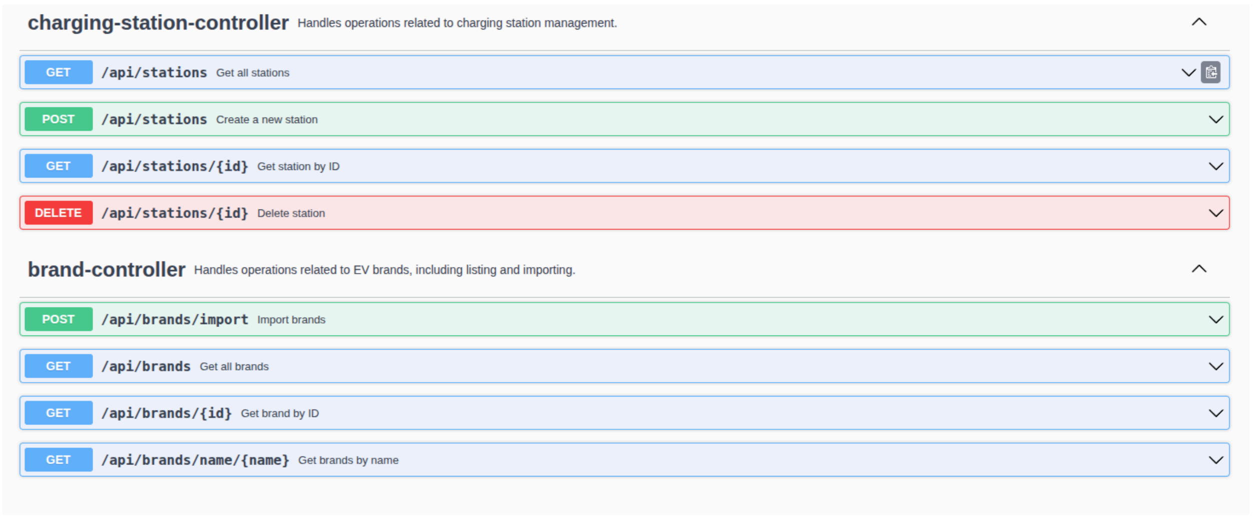
vehicle-controller Exposes endpoints for retrieving and importing electric vehicles.

POST /api/vehicles/import Import vehicles
GET /api/vehicles Get all vehicles
GET /api/vehicles/{id} Get vehicle by ID
GET /api/vehicles/brand/{name} Get vehicles by brand

reservation-controller Handles all reservation operations.

GET /api/reservations Get all reservations
POST /api/reservations Create a reservation
POST /api/reservations/{id}/pay Pay for a reservation
GET /api/reservations/{id} Get reservation by ID
DELETE /api/reservations/{id} Delete reservation

charger-controller Handles charger entities, including creation, update, and assignment to stations.



charging-station-controller Handles operations related to charging station management.

GET /api/stations Get all stations

POST /api/stations Create a new station

GET /api/stations/{id} Get station by ID

DELETE /api/stations/{id} Delete station

brand-controller Handles operations related to EV brands, including listing and importing.

POST /api/brands/import Import brands

GET /api/brands Get all brands

GET /api/brands/{id} Get brand by ID

GET /api/brands/name/{name} Get brands by name