45426 Teste e Qualidade de Software

TQS: Product specification report

André Ribeiro [112974], Violeta Ramos [113170], Diogo Guedes [114256]

v09/06/2025

1 Introduction

1.1 Overview of the Project

GreenEV emerges as an integrated technological solution for the electric vehicle ecosystem, comprehensively addressing the fragmentation in charging services. The name GreenEV reflects its essence: ”EV” represents the electric vehicles at the platform’s core, while ”Green” signifies the commitment to environmental sustainability.

Positioned as a comprehensive digital platform, GreenEV seamlessly integrates three essential components: real-time charging station discovery, smart reservation system, and unified payment mechanism. This integration delivers a frictionless experience for EV drivers and charging station operators, eliminating current industry barriers.

1.2 Limitations

As a minimum viable product (MVP), GreenEV has the following limitations:

- Limited integration with third-party charging networks
- Basic analytics for station operators
- Single payment gateway implementation

2 Product Concept and Requirements

2.1 Vision Statement

GreenEV establishes a complete ecosystem that intelligently coordinates drivers' needs, infrastructure managers' operational requirements, and society's environmental goals, positioning itself as a catalyst for Portugal's transition to electric mobility.

The platform's competitive advantage lies in:

- Full interoperability with major national charging networks
- Intelligent algorithms predicting charging needs and optimizing resource distribution
- Intuitive and accessible user interface design

2.2 Personas and Scenarios

2.2.1 Personas

Maria Silva - Urban EV Driver
35-year-old materials engineer living in an apartment without private charging. Depends entirely on public charging stations. Needs real-time availability information and route optimization to minimize charging time.

Carlos Santos - Long-Distance EV Traveler
42-year-old photographer requiring intercity travel planning. Needs reliable advance reservations and route planning to alleviate range anxiety during rural travel.

Ana Ferreira - Charging Network Manager
38-year-old operations manager needing centralized station monitoring, maintenance alerts, and usage analytics to optimize network performance.

2.2.2 Scenarios

Scenario 1: Find and Reserve

Maria with 30% battery opens GreenEV, views available stations, filters by fast-charging speed, reserves a slot for 20 minutes later, and receives navigation instructions.

Scenario 2: Trip Planning

Carlos plans a 400km trip. GreenEV automatically suggests optimal charging points along the route with advance reservation options.

Scenario 3: Station Management

Ana uses GreenEV's admin dashboard to monitor station status, receives maintenance alerts, and analyzes usage statistics for operational optimization.

Scenario 4: Environmental Impact Monitoring

A driver visualizes CO savings compared to combustion vehicles and can share environmental impact statistics on social media.

2.3 Project Epics and Priorities

Epic	**Description**	**Priority**
Station Discovery	Real-time station location with connector types, availability status, and filtering capabilities	High
Reservation System	Slot booking, schedule management, and operator reservation tools	High
Payment and Billing	Credit card processing, transaction management, and financial reporting	Medium
Station Administration	Station registration, maintenance alerts, and operational management	Medium

Table 1: Project Epics and Priorities

3 Domain Model

The core domain entities include: User Table: Manages user accounts including both drivers and charging station operators, with role-based access control through the role field and authentication via email/password credentials

Vehicle: Represents electric vehicles owned by users, storing essential details like brand, model, license plate, and connector type compatibility for charging station matching

Charging Station: Defines physical charging locations operated by users, including geographical coordinates (lat/lon), maintenance tracking, and visual documentation through photo URLs

Charging Spot: Represents individual charging points within each station, specifying technical capabilities like charging velocity, connector types, power output (kW), pricing per kWh, and real-time availability status

Session: Records complete charging transactions linking users, their vehicles, and specific charging spots, tracking session duration (start/end times), associated costs, and unique session identifiers for billing and analytics

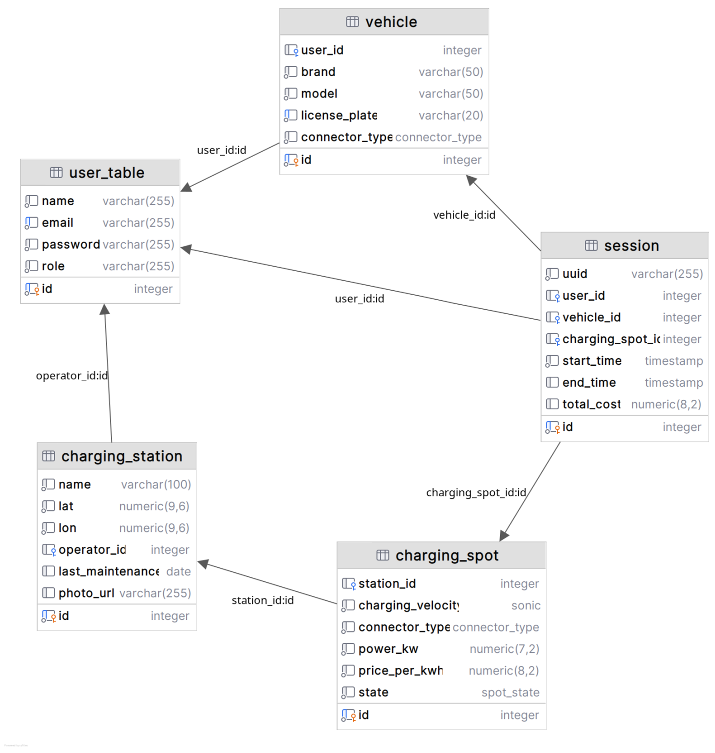

Figure 1: UML Domain Model Diagram

4 Architecture Notebook

4.1 Key Requirements and Constraints

This system has multi role user platforms. Also, it's created from scratch, which means that there are no legacy systems to deal with, so we can design it using modern technologies, and no data is adaptable.

Regarding deployment and different server conditions, the project should work in the same way, no matter the server it is hosted on. Also, it should have remote access, so that clients can buy trips from anywhere in the world.

The backend should require authentication for all data-sensitive operations, */private/some_endpoint*, so that authorization is applied and thus each person gets only access to the data they are meant to get or modify.

An external payment API should be used for making this project available to buy the charging sessions.

4.2 Architecture View

This project is defined by a micro-service architecture, on a Docker environment. It creates a unified experience across multiple environments, which means that the whole project is decoupled from the server(s) it's hosted on. Also, it makes it easy to enable remote access, with a simple port forwarding needed, since the port exposing is part of the Docker solution and not the host.

The backend consists of a Spring Boot app that is connected to a PostgreSQL database through the JDBC (Java Database Connectivity) protocol. The database is connected to a separate Docker network, that only the backend and the database can access, which means that no other container can communicate with it.

There are one frontend service: the main portal, adapted to the staff and client. All of them are implemented in React, using the Next.js tool.

All these containers have no exported ports due to security and CORS blocking. To circumvent this, a nginx-powered reverse proxy exposes ports 80 and 8080 (only in development builds), and maps requests to the respective containers. It allows further access control, as well as secure protection to the upstream servers.

Backend and frontend communicate through a REST API, using HTTP underneath.

CI/CD workflows are used for automatic testing, integration and deployment through GitHub Actions, which leads to better code quality and lower downtime. Check section 4.3 for more regarding deployment.

The technology stack includes:

- **Backend**: Spring Boot (Java)
- **Frontend**: Next.js (React with shadcn)
- **Database**: PostgreSQL
- **Monitoring**: Grafana + Prometheus
- **Load Testing**: K6
- **Web Server**: Nginx

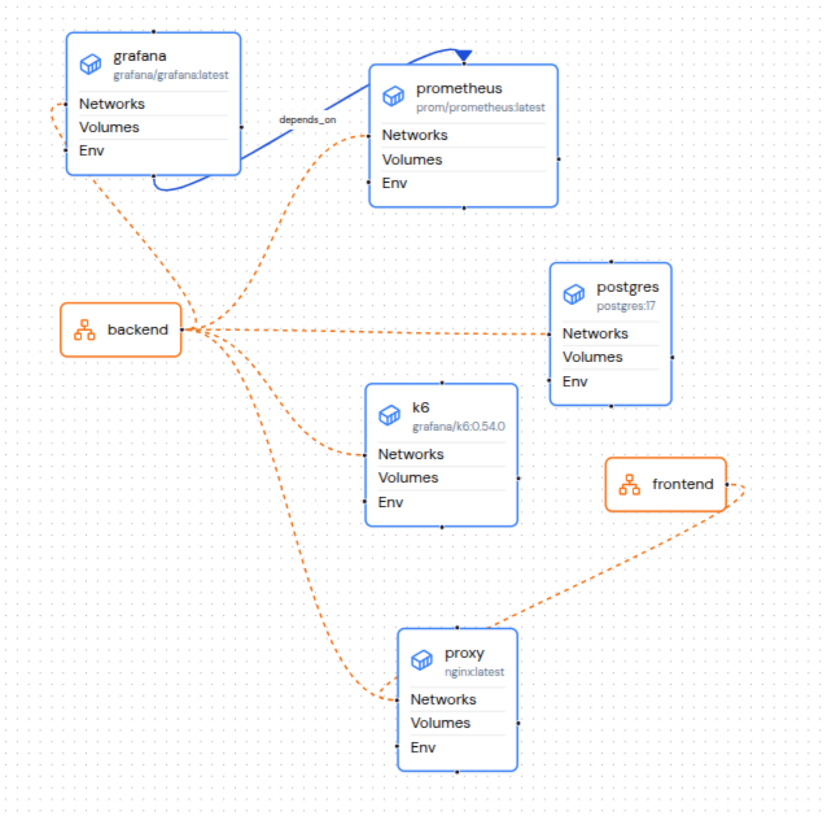

Figure 2: System Architecture Diagram

4.3 Deployment Architecture

The deployment architecture is pretty much the same as the development one, since both use Docker containers in a similar environment.

However, some environment variables are changed, such as secrets, and production builds are used instead of development ones. This is defined in production-specific Dockerfile and Docker Compose schemas, with a deployment as simple as running `docker compose -f compose.prod.yaml up -d` in the terminal.

5 API for Developers

Documentation regarding the API can be reached through a Swagger instance that is embedded in the Spring Boot app and implements the OpenAPI protocol. Additionally, it is crucial to mention the two unique users whose existence is provided by the default data:

User	**Email**	**Password**
admin	admin@admin.com	password123
user	user@user.com	password123

Table 2: Default Users

6 Project Resources

Resource	**Location**
Git Repository	https://github.com/GreenEV-Project
Issue Tracking	Project Issues
API Documentation	Live API Docs
CI/CD Pipeline	GitHub Actions
Monitoring Dashboard	Grafana at http://deti-tqs-20.ua.pt/grafana

Table 3: Project Resources

References

[1] Spring Boot Framework. *https://spring.io/projects/spring-boot*

[2] Next.js Documentation. *https://nextjs.org/docs*

[3] PostgreSQL Official Site. *https://www.postgresql.org*