TQS: Product specification report

Afonso Ferreira [113480], Ricardo Antunes [115243], Tomás Brás [112665]
v2025-04-25

1 Introduction

1.1 Overview of the project

This project is called **EcoCharger** and consists of developing a software platform to help electric vehicle (EV) drivers easily locate, reserve, and use charging stations while providing an efficient back-office management system for operators and administrators.

EcoCharger aims to centralize key services, for example, station discovery, real-time charger availability, slot reservation, charging monitoring, payment processing, and historical usage tracking into a seamless and intuitive application. It is designed to benefit both drivers (providing a better charging experience) and station operators (offering control, monitoring, and maintenance tools).

This project also offers a unique opportunity to apply **testing methodologies**, such as unit testing, integration testing, and system testing, to ensure that EcoCharger meets the highest quality standards. In addition, there will be a strong emphasis on test automation and **quality metrics analysis to continuously monitor** and improve the software's effectiveness.

Therefore, the EcoCharger project not only demonstrates the practical application of the concepts learned in the TQS course but also delivers a valuable solution to enhance the electric vehicle charging experience for users and increase the operational efficiency of charging station services.

1.2 Known limitations

During the development of the application, several relevant functionalities were identified; however, certain technical limitations prevented their implementation at this stage. One of these limitations concerns navigation support: to provide users with route guidance, it would be necessary to implement a navigation system or integrate with third-party services. As a result, users currently need to manually copy the destination address or switch between apps to start navigation. Integration with applications such as Google Maps or Waze would require the use of platform-specific APIs, which in turn would demand additional development time and extensive testing to ensure compatibility across different operating systems (Android/iOS).
Another limitation relates to the charging status updates, which are not provided in real time. Achieving real-time updates would require multiple frequent requests to the server, which could negatively affect performance and data freshness. To overcome this, technologies such as WebSockets or Firebase would be necessary, both of which involve architectural complexity beyond the current project scope.
Moreover, the application requires a constant internet connection to display up-to-date information about station availability, charging point status, and reservation details. Offline functionality is not supported in the current version.
Push notifications also represent a current limitation, as users do not yet receive real-time alerts such as "charging complete" or "reservation confirmed." Enabling this functionality would require integration with an external service such as Firebase Cloud Messaging, as well as token management per device and explicit permission from the user all of which require a separate configuration and security layer not yet in place.

.

1.3 References and resources

https://www.dgeg.gov.pt/
https://www.tomtom.com/
https://www.chargepoint.com/drivers/mobile?srsltid=AfmBOorCp3LJIX-DVVAAN2xVwOetiSSzeXdE2AdxSDwQTIb2RzFSzkHU/
https://swagger.io/

2 Product concept and requirements

2.1 Vision statement

Our application aims to offer a centralized point for charging EV cars, while managing the allocation of different vehicles towards different spots. Additionally, it will serve to make people aware of the differences between EV cars and gas cars, by pointing out differences in prices, and guiding the user during the charging process.

2.2 Personas and scenarios

<table>
<tr><td>Persona:</td><td colspan="3">EV Enthusiast</td></tr>
<tr><td>Photo:</td><td> 
</td><td>Goals and tasks:</td><td>● Monitor the car’s consumption, comparing it with a car that uses gas
● Book a charging station so that he can use it when he arrives
● Check where are the closest charging stations wherever he is</td></tr>
<tr><td>Fictional name:</td><td>Roberto Siqueira</td><td>Environment:</td><td>He completed his law degree a few years back, and quickly got a lot of exposure. With many cases around the country, he needs to drive to a lot of places. He loves his EV, and believes he is helping the planet. He is somewhat accustomed with technology, and his trips are all planned.</td></tr>
<tr><td>Job title/ major responsibilities</td><td>Lawyer</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Demographics</td><td>● 34yo
● Works all around the country</td></tr>
</table>

Scenario:

Roberto Siqueira is a renowned lawyer, and to reach the courts where he works, he often has to travel long distances on the highway which can be a challenge for the battery range of his electric vehicle. Before leaving home, he opens the app and checks the current range of the vehicle. Since it’s a long trip, he reserves a charging spot near the courthouse in advance, ensuring that he can recharge and return home without any issues. Roberto also regularly checks the consumption and savings statistics in the app. He does this often because he enjoys seeing the positive environmental impact of switching from a gas-powered car to an EV. If he needs to charge along the way, he opens the app’s map and selects the most convenient charging station based on location and availability. This kind of planning is part of his routine. He values efficiency and technology, combined with his concern for the environment.

<table>
<tr><td>Persona:</td><td colspan="3">EV Sceptic</td></tr>
<tr><td>Photo:</td><td> 
</td><td>Goals and tasks:</td><td>Figure out if using an EV is really cheaper than a gas-driven car (and more planet-friendly)
Park his car wherever he wants, without the need to book a slot, and pay it afterwards
Be able to pick up his car as soon as the battery is charged</td></tr>
<tr><td>Fictional name:</td><td>Pedro Antunes</td><td>Environment:</td><td>He completed his degree in Physics Engineering a long time ago. He is an enthusiast of technology, and can work around it, but he just prefers gas cars to EV cars. However, his company-provided car ended up being an EV, to his disdain. Now he has to adapt to this situation and be able to use this car without much complications. He likes reliability, so he wants to be able to park around the same place every day, without needing to worry about finding a place to charge his car.</td></tr>
<tr><td>Job title/ major responsibilities</td><td>Factory Coordinator</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Demographics</td><td>56yo
Does the same commute every day</td></tr>
</table>

Scenario:

Pedro Antunes, a 56-year-old factory coordinator, drives the same route every day to his job just outside the city. Although he's always been a fan of technology, he was never convinced by electric vehicles. But his company recently assigned him an EV, and now he's trying to adapt. Every morning, he drives to work and parks in the same spot near the factory. He doesn't want to deal with booking charging slots or navigating complex apps, so he relies on a simple system that

automatically charges his car while he's working. At the end of the day, he gets a notification that his car is ready. He simply unplugs it, drives home, and checks the app once a week to see if it's actually saving him money compared to his old petrol car like the company told him.

<table>
<tr><td>Persona:</td><td colspan="3">Busy Person That Happens To Have an EV</td></tr>
<tr><td>Photo:</td><td> 
</td><td>Goals and tasks:</td><td>Charge the car and pay for it, with minimal usage of any interface
Book a slot in the nearest supermarket without the need to pick it herself</td></tr>
<tr><td>Fictional name:</td><td>Luísa Laranjeiro</td><td>Environment:</td><td>She has been a teacher for a number of years by now. She has a family composed of 2 small children, which she needs to take to school and bring back home every day. Her schedule is too busy to be able to track when and where she can charge her car, and as such she usually struggles to find a charging station where she needs it when she needs it. She is quite uncomfortable with technology, and does not deal well with most platforms.</td></tr>
<tr><td>Job title/ major responsibilities</td><td>Primary School teacher</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Demographics</td><td>43yo
Has 2 children to take care of</td></tr>
</table>

Scenario:

Luísa Laranjeiro, a 43-year-old primary school teacher, lives in Coimbra but drives daily to Aveiro to teach. Before leaving, she opens the EcoCharger app and checks the charging stations near the school and the supermarket since she usually does the shopping after she finishes her job for the day. She selects the most convenient charging station near the supermarket and books a slot automatically, without needing to navigate through multiple menus. The app confirms her

reservation and handles the payment setup in advance. After a full day of teaching and managing her two children, Luísa heads to the supermarket. When she has done all shopping she unplugs the car and drives home with a fully charged vehicle.

<table>
<tr><td>Persona:</td><td colspan="3">Administrator</td></tr>
<tr><td>Photo:</td><td> 
</td><td>Goals and tasks:</td><td>• Ensure that users and operators are using the platform for its intended usage
• Manage permissions for different users (depending on their use of the platform)
• Approve addition / removal of charging stations</td></tr>
<tr><td>Fictional name:</td><td>David Rodrigues</td><td>Environment:</td><td>He has taken a degree in management a long time ago. He has experience in management of resources, user management, and is familiar with technologies. He is methodical and likes things to be organized.</td></tr>
<tr><td>Job title/ major responsibilities</td><td>Manage and maintain the product</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Demographics</td><td>• 39yo
• EcoCharger Administrator</td></tr>
</table>

Scenario:

David Rodrigues, a 39-year-old EcoCharger administrator, logs into the platform to check recent activity. He has been informed that some of the workers stopped working on a determined charging station so he removes them and adds the new workers assigning them the right permissions. He validates a request to add three charging stations and flags an old, inactive one for removal. Focused on keeping the system organised and efficient, David ensures everything runs smoothly before logging off for the day.

<table>
<tr><td>Persona:</td><td colspan="3">Charging Station Operator</td></tr>
<tr><td>Photo:</td><td> 
</td><td>Goals and tasks:</td><td>Add company’s charging stations to a centralized platform where it can be shown to EV drivers
Update assigned charging stations statuses and information
Check the usage of charging stations, like when and how much it’s being used</td></tr>
<tr><td>Fictional name:</td><td>Agostinho Rocha</td><td>Environment:</td><td>He currently works at a charging station company that wants to expose their business to drivers in the country. He is somewhat experienced with administrative platforms but needs everything organized and ready to access in just a few clicks.</td></tr>
<tr><td>Job title/ major responsibilities</td><td>Manage assigned Charging Stations</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Demographics</td><td>40yo
Installs Charging Operations</td></tr>
</table>

Scenario:

Agostinho Rocha, a 40-year-old charging station operator, starts his day by updating his company’s charging stations on the EcoCharger platform. He adds a newly installed station, checks the status of the existing ones, and reviews usage reports to see which locations are getting the most traffic. Since tomorrow is going to be a holiday he decides to change the available schedules for a specific station. With just a few clicks, he keeps everything organized and up-to-date to help EV drivers across the country.

2.3 Project epics and priorities

Epic 1: Management of Charging Stations and Charging Points

As a Driver, I want to see a map with all charging stations and their charging points, **so that I** can choose a suitable station near my location to charge my car.

Acceptance Criteria:

- Given a Driver wants to see charging stations on the map to select a charging point to charge their car,
- When the driver opens the application, they can see the available charging stations near their location.
- Then, when they choose one of them, the system shows the available charging points at that station.

As a Driver, I want to see the real-time status (available/busy) of each charging point at a station, **so that I** can ensure that I choose a station with available charging points to charge my car.

Acceptance Criteria:

- Given a Driver wants to find available charging points so they can park the car for charging
- When the driver selects a charging station from the application map
- Then they can see all charging points at that station and their availability (available/busy)
- And they can choose an appropriate charging point to charge their car

As an Administrator, I want to add charging stations, **so that I** can make the new available charging stations available to the users.

Acceptance Criteria:

- Given an Administrator has access to the charging stations list
- When the Administrator fills out the form to add a new charging station with its availability information
- Then the new station should be added to the list and become immediately available to users.

As an Administrator, I want to add or remove charging stations, **so that I** can remove nonexistent or unavailable charging stations from the website.

Acceptance Criteria:

- Given an Administrator has access to the charging stations list
- When the Administrator chooses to remove a charging station through the management form or interface
- Then the selected station should be removed from the list and no longer available to users.

As a BackOffice Operator, I want to edit information related to a charging station and charging points, **so that clients** always have access to up-to-date information about charging stations and charging points.

Acceptance Criteria:

- Given a BackOffice Operator wants to update information about a charging station in the application
- When the BackOffice Operator edits information regarding that charging station through a form in the application
- Then the system immediately updates the information and the updated information becomes available to all users.

Epic 2 - Management of Charging Point Reservations

As a Driver, I want to reserve a charging point at a specific time slot, **so that I** can save time and ensure a charging point will be available when I arrive.

Acceptance Criteria:

- Given a Driver wants to make a reservation for a specific time slot
- And that time slot is available to the user
- When the Driver selects the desired time slot, the system shows the available charging points for that time
- Then the driver selects the most appropriate charging point and, if required, completes the payment to ensure the charging point is reserved and available when the Driver arrives, preventing scheduling conflicts.

As a BackOffice Operator, I want to set and manage the schedules for all charging stations and charging points **so that I** can ensure the system works properly and users have access to their reserved charging slots.

Acceptance Criteria:

- Given a BackOffice Operator wants to define the availability for a specific charging point,
- When they select the charging point,
- Then they should be able to set an overall operating schedule (working hours) and, within this schedule, define specific time slots during which the charging station is available for users.

- Given a BackOffice Operator wants to define the operating hours for a charging station,
- When they select the charging station,
- Then they should be able to define the station's overall working time (opening and closing hours), ensuring that all associated charging points follow or inherit this availability by default unless individually configured.

As a BackOffice Operator, I want to remove a reservation from a charging port **so that I** can cover for unexpected situations that require temporarily or permanently shutting down the charging station

Acceptance Criteria:

- Given a BackOffice Operator needs to manage existing reservations,
- When they view a reservation,
- Then they should be able to cancel it, ensuring that users are notified accordingly, and if it is cancelled, the user is refunded

As a BackOffice Operator, I want to modify a reservation for a charging point **so that I** can ensure the system works properly and users have access to their reserved charging slots.

Acceptance Criteria:

- Given a BackOffice Operator needs to modify existing reservations,
- When they view a reservation,
- Then they should be able to modify it, ensuring that users are notified accordingly

---

Epic 3 - Management of App Users

As a Driver, I want to register my car in the application **so that** I can use it at charging stations and receive accurate insights about energy usage, CO_2 savings, and cost efficiency.

Acceptance Criteria:

- Given a Driver wants to register their vehicle for use in charging sessions,
- When they log into the app and navigate to the "My Vehicles" section and fill out the form with the car's details (e.g., brand, model, battery capacity, energy efficiency),
- Then the vehicle is saved to their profile and becomes available for selection during the charging process

As an Administrator, I want to add specialized operators to the system **so that I** can assign them to charging stations.

Acceptance Criteria:

- Given an Administrator wants to insert a new operator into the system,
- When the Administrator fills the form to add a new operator into the system,
- Then the operator's account is created with a password that is sent to their email, forcing them to change it on the first login

As an Administrator, I want to assign specialized operators to service stations **so that I** can ensure that maintenance is carried out if necessary and that charging points are correctly operated by clients.

Acceptance Criteria:

- Given a charging station and an operator are available in the system,
- When the Administrator assigns the charging station to the operator through the assignment form,

- Then the operator should be linked to that station and gain the necessary permissions to manage and maintain it

---

Epic 4 - Management of Charging Results

As a Driver, I want to monitor the real-time charging status of my car **so that I** can track the progress of the charging process.

Acceptance Criteria:

- Given a Driver wants to see the status of their charging session
- When they initialize the charging process
- Then they should be able to view real-time information about the charging status, including the current charging time, progress, energy consumed and estimated cost

As a Driver, I want to view the savings compared to combustion vehicles during the charging process **so that I** can analyze costs and understand the advantages of using an electric vehicle.

Acceptance Criteria:

- Given a Driver wants to analyze the cost savings during the charging session
- When they view the charging session details
- Then they should see the comparison between the cost of charging and the average cost of fueling a combustion vehicle for the same distance.

As a Driver, I want to view the history of my charging stations, **so that** I can see how much I've charged my car and how much money I've been spending.

Acceptance Criteria:

- Given a Driver wants to view their charging session history,
- When they access the charging history section,
- Then they should see a list of all past charging sessions along with details such as cost, duration, and energy consumption for each session.

As a Driver, I want to review a specific charging session of my car, **so that I** can get detailed insights into that particular session.

Acceptance Criteria:

- Given a Driver wants to review a specific charging session,
- When they select a particular session from the history list,
- Then they should be able to view detailed information, including the time the session started and ended, the total energy consumed, and the cost incurred.

As a Driver, I want to view estimated charging time and savings when I select a charging point, **so that** I can make an informed decision based on my car's characteristics and plan my charging session accordingly.

Acceptance Criteria:

- Given a Driver has a vehicle registered in their profile,
- When they click on a specific charging point on the map,

- Then the app displays an estimate of: Total charging time to full capacity (based on the car's battery and charging power), Energy cost for a full charge or session duration, CO_2 savings and money saved compared to a combustion vehicle

As a BackOffice Operator, I want to access all statistics related to charging activities at charging stations, **so that I** can monitor performance, usage, and efficiency trends.

Acceptance Criteria:

- Given a BackOffice Operator wants to access charging activity statistics,
- When they log in to the system and navigate to the statistics dashboard,
- Then they should be able to view an overview of all charging activities, including total usage, number of sessions, and total energy consumed.

Epic 5 - Management of Payments

As a Driver, I want to complete payments at service stations using either the app's wallet or direct debit from a card, **so that I** can easily pay for my charging sessions and ensure my vehicle is charged.

Acceptance Criteria:

- Given a Driver wants to pay for a charging session,
- When they initiate the payment process at the service station,
- Then they should be presented with the option to pay using either the app's wallet or a direct debit from a linked card.

As a Driver, I want to be able to enter a code on the charging point monitor, so that I can quickly start the car charging process.

Acceptance Criteria:

- Given a Driver wants to initiate the charging process,
- When they open the charging provider's mobile app,
- Then they should see a unique numeric or alphanumeric code associated with their session or vehicle.

- Given the Driver has received a code from the mobile app,
- When they enter that code into the charging station monitor,
- Then the system should recognize the code and automatically initiate the car charging process.

Epic 6 - Integration with Third-Party Services

As a Third-Party Service I want to access the list of available charging stations through the API, **so that I** can integrate this information into my own system and allow my users to find charging stations.

Acceptance Criteria:

- Given that the third party has permission to use the API,
- When the third party makes a request to get the charging stations,
- Then they receive an updated list of all available stations and charging points with their respective information.

As a Third-Party Service I want to access real-time availability information of charging points through the API**, so that I** can display the availability to my users in real-time.

Acceptance Criteria:

- Given that the third party is requesting information about a specific charging point,
- When the third party makes a request for availability,
- Then they receive the availability information (whether it is available or occupied) for the requested charging point.

As a Third-Party Service I want to make a reservation for a charging point on behalf of my users,**so that they** can ensure the point will be available when they need it.

Acceptance Criteria:

- Given that the third party is integrating the reservation functionality
- When the third party makes a reservation request for a charging point on behalf of the user
- Then the reservation is confirmed, and the charging point is reserved for the requested time slot.

As a Third-Party Service I want to access the charging history of my users through the API,
so that I can provide detailed reports on the usage of charging points and car efficiency.

Acceptance Criteria:

- Given that the third party has permission to access user data
- When the third party requests the charging history of a user
- Then they receive detailed information about past sessions, including cost, duration, and energy consumed.

The user stories highlighted in red are the ones that were not implemented; all the others were fully implemented.

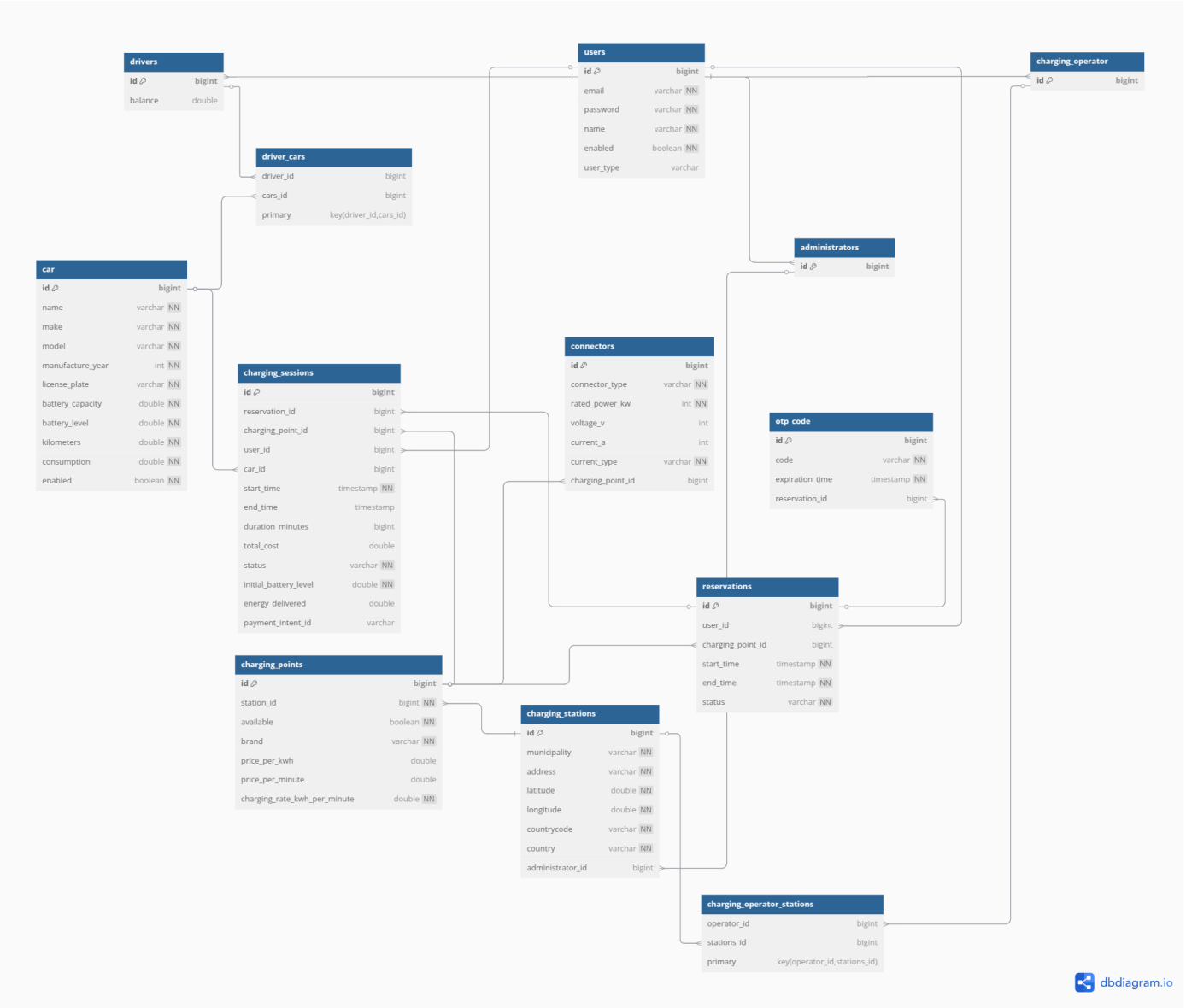
3 Domain model

In this domain, the key information concepts managed are **User**, **ChargingStation**, **ChargingPoint**, **Connector**, **Car**, **Reservation**, and **ChargingSession**. These concepts reflect the core logic of an electric vehicle (EV) charging network and how users interact with its infrastructure.

A **User** can be one of three roles: *Driver*, *Administrator*, or *ChargingOperator*. Each role has specific responsibilities. For instance, a *Driver* owns one or more **Cars**, which can be used to initiate **Reservations** for charging.

A **ChargingStation** is managed by Charging Operators and all administrators and contains one or more **ChargingPoints**, each of which represents a physical outlet available for vehicle charging.

Reservations are created by *Drivers* to book a specific **ChargingPoint** at a defined time window. These reservations may then lead to **ChargingSessions**, which represent actual usage, capturing details such as energy consumed, session duration, and cost.



4 Architecture notebook

4.1 Key requirements and constraints

We identified a lot of key architectural drivers and characteristics that must be considered for designing and implementing the system effectively.

- First, we need to take into account that users will use the app mostly through a mobile interface, particularly Drivers, who will frequently interact with the system on the go. This implies that the application must be:

- Responsive and mobile-optimized to support various screen sizes and mobile browsers.
- Capable of delivering fast-loading interfaces even under unstable mobile network conditions.

The system will also need integration with several external services, so we can develop features dependent on external APIs. For example:

- Route Planning and Geolocation: By integrating with OpenStreetMap the app can easily provide users with the real location of the nearby charging stations.

- Payment Processing: Integration with Stripe allows users to add funds to their wallet, subscribe to charging plans, or pay per session. This simplifies financial operations while offloading compliance and security management (e.g., PCI-DSS) to a trusted provider.

- Fuel Price and CO2 Comparison: With the DGE Gasoline API, the system can calculate environmental benefits such as CO2 saved or money saved by using electric charging versus traditional gasoline, providing users with added value through insights and incentives.

- Visualization and Analytics: Using Chart.js in the frontend enables real-time visual representation of user statistics, charging history, station utilization, and performance data. This will help both drivers and back-office operators make informed decisions by visualizing data in the form of graphics.

For secure authentication and authorization, we will use Jwt tokens which the user will need to have to make personal API requests.

We will need role-based access control (RBAC) for managing privileges across Drivers, BackOffice Operators, and Administrators.

To ensure consistent deployment across various environments, the entire application including the backend, frontend, and database is packaged using **Docker containers**. These containers bundle all necessary dependencies, configurations, and runtime environments, so the application behaves the same whether it's running on a developer's local machine, a staging server, or in production. This approach greatly reduces deployment-related issues and simplifies the process of managing different environments.

4.2 Architecture view

Our architecture is focused on a modular, layered architecture style using the logical decomposition of responsibilities. It separates concerns into distinct functional modules, so the system can be easily maintained and scale. promoting maintainability, scalability, and extensibility.

It’s composed of the **following key** layers and components:

1. **User Interface**

    - This layer is responsible for user interaction. It will be developed in ReactJS and will work as the user interface for drivers, administrators, and back-office operators.

- It communicates with the backend through RESTful API calls using *axios* and integrates third-party libraries such as **OpenStreetMap** for maps and **Chart.js** for data visualization.

2. **Charging Slot Interface Container**
   - This layer corresponds to the interface that will be operating in the charging slots. It will show a 6 digit code for the user to start the charging process

3. **API Layer (REST Controllers)**
   - This layer exposes endpoints that the frontend interacts with.
   - Each controller handles specific domains and delegates logic to the appropriate service layer components.
   - The controllers are:
     - **AdministratorController -** endpoints for managing administrator operations
     - **AuthenticationController -** Handles login, registration, password management, and OTP verification.
     - **ChargingPointController**
       Manages endpoints related to charging points – status updates, availability, etc.
     - **ChargingSessionController -** starts, stops, and monitors charging sessions for users.
     - **ChargingStationController -** Provides endpoints for managing charging stations and charging slots
     - **DriverController -** Handles operations for driver profiles, vehicle data, wallet management
     - **ReservationController -** Manages booking of charging slots, scheduling, and cancellations.

4. **Service Layer (Business Logic)**
   - Contains the core business logic and orchestrates workflows between the controllers, models and the external services
   - The services in this layer are:
     - **AuthenticationService** – Manages login, registration, and user roles.
     - **ReservationService** – Handles booking of charging slots.

- **ChargingSessionService** – Starts and stops charging sessions.
- **DriverService** – Manages driver profiles and wallet.
- **AdministratorService** – Manages operators and admin tasks (BackOffice)
- **ChargingStationService** – Controls charging station settings and status.
- **ChargingPointService –** Manages individual charging points (part of station infrastructure)
- **StripeService** - Manages connection with the Stripe External Payment tool

5. **Repository Layer**

The Repository Layer is responsible for accessing and managing data in the database. It serves as an abstraction over the persistence mechanism, allowing the Service Layer to interact with data without needing to write raw SQL.

Using Jakarta Persistence API (JPA), each repository interfaces with the corresponding entity class and provides standard CRUD operations.

The repository interfaces are:

- **AdministratorRepository.java**
  Handles database operations related to administrator accounts and their roles.

- **CarRepository.java**
  Manages persistence for driver car details such as model, battery capacity, etc.

- **ChargingOperatorRepository.java**
  Handles data related to charging station operators or service providers.

- **ChargingPointRepository.java**
  Manages CRUD operations for individual charging points (physical connectors).

- **ChargingSessionRepository.java**
  Stores and retrieves data about charging sessions, such as start/end times and energy consumed.

- **ChargingStationRepository.java**
  Deals with charging station entities including location, status, and configurations.

- **DriverRepository.java**
  Manages driver profiles, including personal info, vehicle associations, and wallet data.

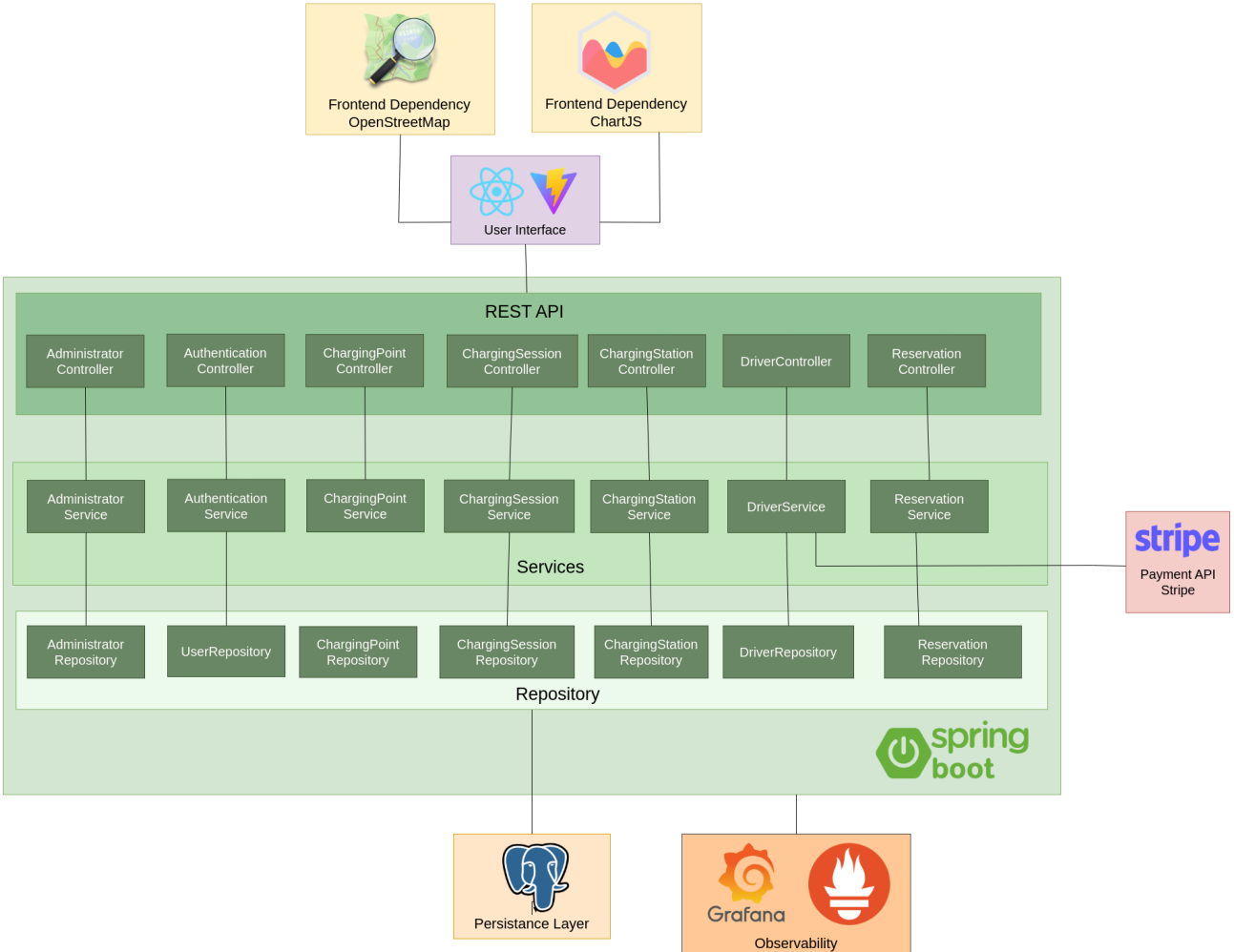
- **OTPCodeRepository.java**
  Handles temporary one-time codes used for authentication or verification.

- **ReservationRepository.java**
  Manages booking and reservation of charging slots for drivers.

- **UserRepository.java**
  Stores generic user account data, possibly shared across admins, drivers, and operators.

6. **Integration Layer (External APIs)**
    - This layer manages communication with external systems such as:
        - **Stripe** for payment processing,
        - **OpenStreetMap** for maps
    - These integrations are abstracted to ensure loose coupling and easy substitution if needed.
7. **Persistence Layer (Database Access)**
    - Handles data storage and retrieval using a relational database.
    - All entities such as users, bookings, charging sessions, and stations are managed here.
8. **Observability Layer**
    - This layer is responsible for monitoring, logging, and visualizing the state and performance of the application.

This layered structure allows the development team to work on features independently, and isolate bugs efficiently. Besides that, since each module communicates through well-defined interfaces, it supports future enhancements like microservice decomposition.



4.3 Deployment view

To ensure consistency across development, testing, and production environments, the system will be deployed using five main Docker containers. Each container encapsulates a specific part of the system:

- **User Interface Container**
  - Responsible for: Serving the user interface **(React + Vite).**
  - Provides the web application to end users.
  - Will make the API requests requested by users in the interface by communicating with the backend via **HTTP requests**
  - Runs on port 5000

- **Backend Container**

  - Runs the Spring Boot application, exposing its RESTful API over HTTP. It will deal with all the business logic, communication with the database
  - Makes requests to the external services
  - Runs on port 8080

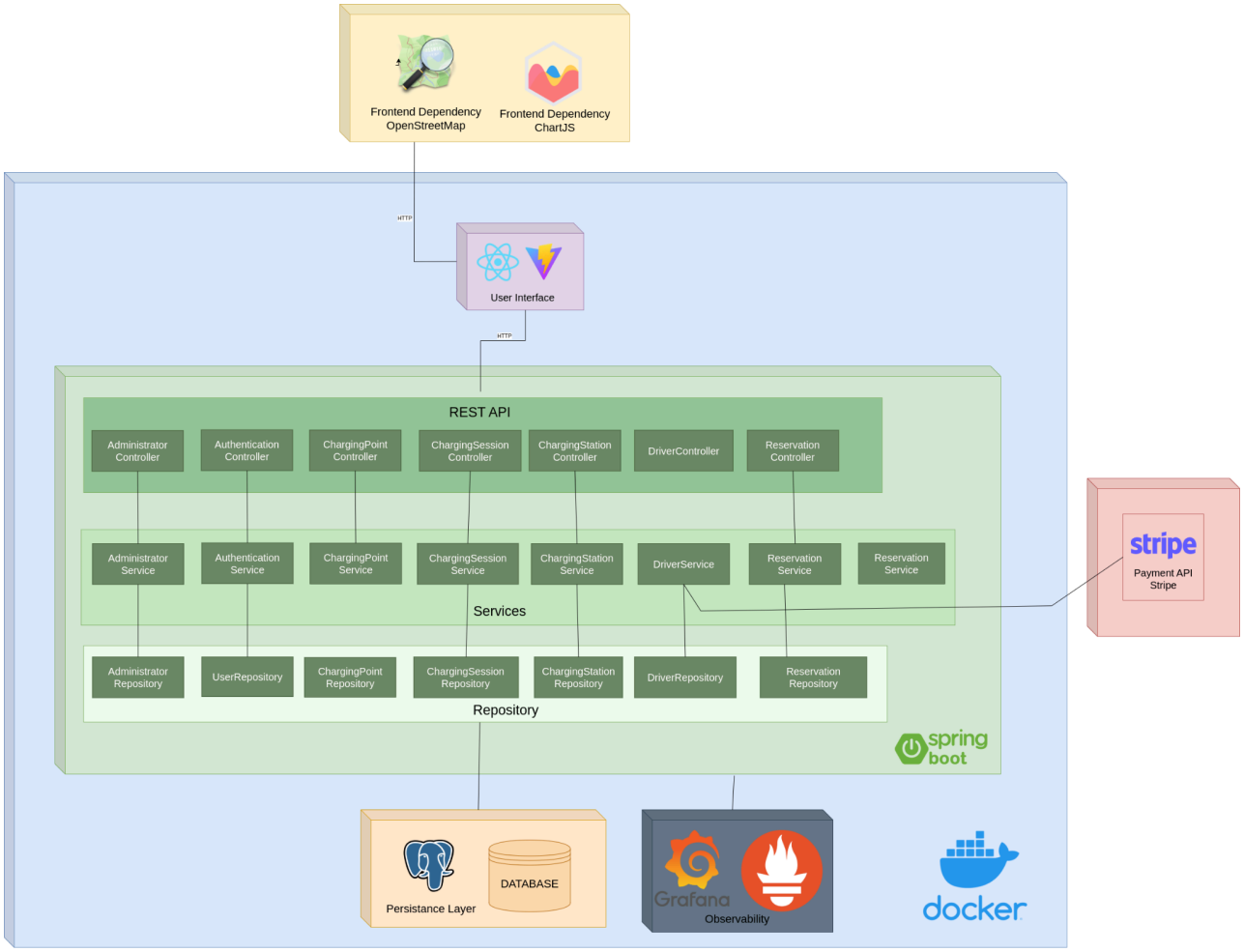
- **Database Container**

  - Runs the persistence database (PostgreSQL)
  - Will store all the application data like users, bookings, etc..
  - Not accessible from the outside, only from the backend container.

- **Monitoring Container – Prometheus**
  - **Responsible for:** Collecting metrics from the backend application and infrastructure.
  - Monitors the health and performance of all services through configured endpoints (e.g., Spring Boot's /actuator/prometheus).
  - Periodically scrapes metrics and makes them available for analysis and visualization.
  - Runs on port 9090

- **Visualization Container – Grafana**
  - **Responsible for:** Visualizing monitoring data collected by Prometheus.
  - Connects to Prometheus as a data source and renders dashboards that display application metrics (e.g., CPU usage, request latency, DB connections).
  - Used by developers/operators to monitor system health in real time.
  - Runs on port 3000 and provides a web interface to access pre-configured or custom dashboards.

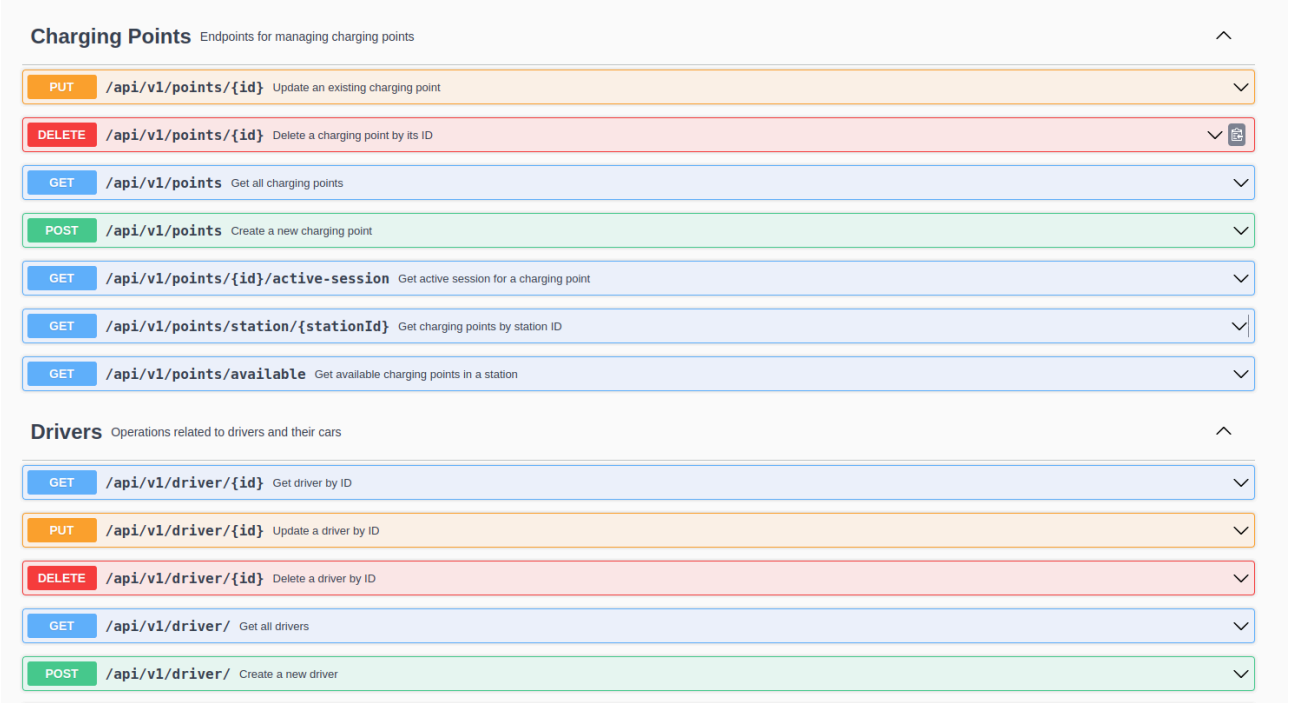
These will be the Docker Containers that will be running on the VM.

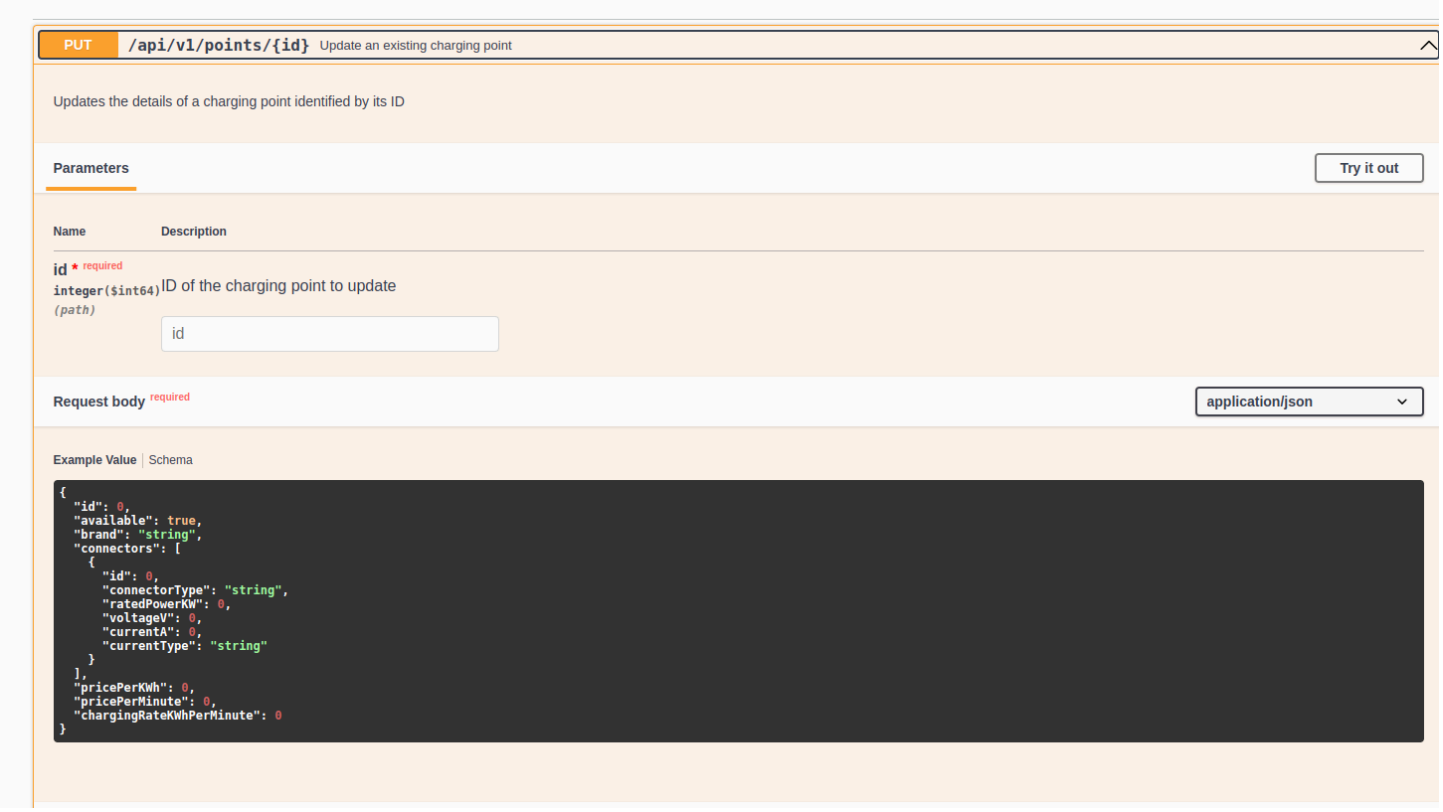


5 API for developers

The API documentation was hosted with Swagger and can be accessed through this link: http://deti-tqs-19.ua.pt:8080/swagger-ui/index.html

All the endpoints are **organized and follow standard REST principles**. Each one uses the appropriate HTTP methods and status codes to make its purpose easy to understand. Most of them receive data in JSON format, which is then validated before being processed. The responses are also well-structured, giving users clear feedback. Thanks to the use of annotations, the API is fully documented in Swagger, which makes it easy to explore and test directly through the interface.


Charging Points
Endpoints for managing charging points
PUT /api/v1/points/{id} Update an existing charging point
DELETE /api/v1/points/{id} Delete a charging point by its ID
GET /api/v1/points Get all charging points
POST /api/v1/points Create a new charging point
GET /api/v1/points/{id}/active-session Get active session for a charging point
GET /api/v1/points/station/{stationId} Get charging points by station ID
GET /api/v1/points/available Get available charging points in a station
Drivers
Operations related to drivers and their cars
GET /api/v1/driver/{id} Get driver by ID
PUT /api/v1/driver/{id} Update a driver by ID
DELETE /api/v1/driver/{id} Delete a driver by ID
GET /api/v1/driver/ Get all drivers
POST /api/v1/driver/ Create a new driver


PUT /api/v1/points/{id} Update an existing charging point
Updates the details of a charging point identified by its ID
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
ID of the charging point to update
id
Request body required
application/json
Example Value | Schema
{
"id": 0,
"available": true,
"brand": "string",
"connectors": [
{
"id": 0,
"connectorType": "string",
"ratedPowerKW": 0,
"voltageV": 0,
"currentA": 0,
"currentType": "string"
}
],
"pricePerKWh": 0,
"pricePerMinute": 0,
"chargingRateKWhPerMinute": 0
}

6 References and resources

Resource	URL / Location
Git Repository	https://github.com/TQS-EcoCharger/TQS_EcoCharger/tree/master
Jira Project	https://ua-deti-alof.atlassian.net/jira/software/projects/ET/boards/2
QA Dashboard	https://sonarcloud.io/summary/new_code?id=TQS-EcoCharger_TQS_EcoCharger&branch=develop
CI/CD Pipeline	https://github.com/TQS-EcoCharger/TQS_EcoCharger/tree/master/.github
Link to website	http://deti-tqs-19.ua.pt:5000/