deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

TQS: Product specification report

Gabriel Santos [113682], Guilherme Santos [113893],

João Gaspar [114514], Shelton Agostinho [115697]
v2025-05-07

1 Introduction

1.1 Overview of the project

The increased adoption of electric vehicles (EVs) has introduced new infrastructure challenges, such as the fragmentation across different applications (Waze EV, PlugShare, ChargeMap) and charging networks, as well as difficulties in locating available charging stations. **PowerNest** emerged to integrate this ecosystem, offering a unified platform that simplifies station search, reservation, payment, and usage monitoring for both drivers and station operators.

With the platform, drivers can easily find nearby charging stations, book time slots, unlock chargers, and process payments seamlessly, minimizing "range anxiety" and optimizing trip planning. Station operators, on the other hand, use a centralized management dashboard to register new stations, update pricing, monitor station usage, and schedule maintenance, thus improving operational efficiency and user satisfaction.

To ensure that this solution operates safely, reliably, and efficiently, this project adopts rigorous Software Quality Assurance (SQA) practices. From defining testable requirements and user stories with clear acceptance criteria to implementing automated tests, continuous integration (CI/CD), code analysis, and security practices, our aim is to deliver a platform with high reliability, consistent performance, and an outstanding user experience.

1.2 Known limitations

Our application has some critical known limitations, in order to implement and scale our application, we would have to find a way to unite all the different charging stations software, since each one has their proprietary software, therefore, we would need to create an adaptor layer that would connect all the available software with our application. Which would be a big obstacle, but nonetheless would be surmountable. We didn't have enough time to implement payments.

1.3 References and resources

- React - JavaScript library for building UIs.
  https://reactjs.org/

- Spring Boot - Framework for creating Java/Spring applications.
  https://spring.io/projects/spring-boot

- Swagger UI - Tool for generating interactive documentation for REST APIs.
  https://swagger.io/tools/swagger-ui/

- GitHub Actions - CI/CD integrated with GitHub with self-managed runners.
  https://docs.github.com/pt/actions

- Prettier - Opinionated code formatter to maintain style consistency.
  https://prettier.io/

- Docker Compose - Orchestration of multiple Docker containers in a development environment.
  https://docs.docker.com/compose/

- Professor Sérgio's Tutorials - Tutorial provided by the professor for configuring Jira and Xray.

- YouTube - Jira and Xray - Short Xray tutorials.
https://www.youtube.com/@XrayApp/featured

2 Product concept and requirements

2.1 Vision statement

Our system consists of providing a simplified interface for users to book and charge their electric vehicles.
Nowadays, electric cars are at an all time high, hence an increase in demand for public car chargers follows it, therefore it can be frustrating to charge an EV, since, when a user wants to charge their car they have to be lucky to find a free charger, or find a charger that supports fast charging to not waste their time. We aim to fix that problem by allowing users to book a reservation in a charger available in our app, which allows a decrease in entropy in the everyday life of our users and to know exactly where they are charging their cars.
To achieve such a goal, firstly we plan to implement a map that provides our users with valuable information of the stations found around them, such as the number of available chargers, if they are all in use, what are the respective charging power and supported charging connectors.
After that first wave of features, we will start allowing our clients to make bookings to the available chargers to a specified date and time, to allow them to more efficiently plan out their days.
Our next steps would be adding new stations/companies to our application to allow it to scale information wise and availability wise, and add a payment method to our app, which will fix the major problem with current charging stations: Every charging station requires a specific company card for it, such as Galp, EDP, etc.
With our solution, we would unite every charging station out there, simplifying everyone's life.
However, with a payment method comes always a need for receipts and consumption history, to allow our users to take care of their spendings and understand exactly how much they are spending every month/year on their cars.

2.2 Personas and scenarios

Persona 1: Pedro Almeida – Daily EV Driver

- **Age:** 34 years old
- **Profession:** Analyst (works in an office in the city)
- **Biography:** Pedro lives in an apartment and does not have his own garage with a charger, so he depends on public charging stations for his electric vehicle. He is a technology and sustainability enthusiast, having bought an EV to save on fuel and reduce his carbon footprint. He uses the car daily to go to work and take his children to school, making mainly urban trips.
- **Objectives with the use of the platform:** Quickly find an available charging station nearby during his daily routine; minimize the waiting time for charging; keep track of how much energy he spends and consumes per month to control his family budget.

Pedro wants to use **a single, reliable platform** that helps him plan his weekly top-ups without any hassle.

- **Frustrations:** Before **PowerNest,** Pedro was frustrated by having to switch between several apps from different operators to locate an available charging station. It has happened that he has arrived at a station that was available, but when you arrive you find it out of service – a huge waste of time. In addition, he hates having to carry ID cards or register with each charging network separately. Another frustration is not being clear about the cost of recharging until he receives separate invoices later.
- **Specific needs:** Reliable real-time **information** on the availability and operation of the stations; possibility to **filter by type of connector** compatible with his car; simple interface that shows distance and estimated time to the charging point; **automated payment** in the app so he doesn't have to deal with cash or cards on the spot; and access to a **detailed history** of recharges (kWh consumed, amount paid, location and date) to monitor his monthly consumption.

Persona 2: Carla Mendes – Long Distance Traveler with EV

- **Age:** 41 years old
- **Profession:** Sales Representative (regional sales)
- **Biography:** Carla works traveling by car through several cities in the countryside, visiting clients throughout the week. She spends many hours on the road and her electric sedan is her work tool. Because she travels to new places frequently, Carla needs to carefully plan the stops for recharging during her trips, ensuring that she will be able to reach her destinations without delays. She values punctuality in meetings and can't afford to run out of battery midway.
- **Objectives with the use of the platform: Plan efficient travel routes** considering charging points along the way; book a charging time in advance to avoid arriving at a station and finding it busy; optimize her time on the road knowing exactly where and when to stop to charge. Carla also wants to have **payment receipts** organized, as she needs to report the recharge costs for reimbursement by the company.
- **Frustrations:** Carla has already faced the situation of arriving at an electric station in an unknown city and discovering that all the points were occupied or out of operation. This forced her to wait a long time and almost caused her to miss a work appointment. She is frustrated by the lack of integration between networks: having to maintain multiple apps and registrations for different regions.
- **Specific needs: Parking space reservation function** at the charging station, to ensure that when arriving at a certain time there will be a connector available for it; estimation of the **recharging time needed** to achieve the desired autonomy, helping in planning her schedule; integration with **a navigation** system to trace the route to the chosen station; and store detailed receipts for each recharge (date, location, cost, energy) to facilitate accountability to the company.

Persona 3: Luís Fernando – Charging Station Operator

- **Age:** 52 years old
- **Profession:** Entrepreneur (owner of a small electric station business)
- **Biography:** Luís owns two fast charging stations located at strategic points in the city. He entered the EV infrastructure business when he noticed the growth of this fleet and wanted to offer a quality service to drivers, while seeking to monetize his investment in equipment. Luís is not a programmer, but he deals well with computers; He himself monitors the day-to-day operations of his stations.

- **Objectives with the use of the platform: Promote your charging stations** to reach as many drivers as possible through the unified platform; manage the **availability and prices** of his stations (for example, adjust fares during peak hours or offer loyalty discounts); update **alerts of failures or maintenance needs** in an agile way; monitor **financial and operational** performance – how many recharges are made per day, how much energy is being supplied and what is the accumulated billing. In short, Luis wants a centralized tool to **manage and optimize his stations** without complication.
- **Frustrations:** Before using **PowerNest,** Luís depended on customers to call or complain to find out that a station had a problem, which generated dissatisfaction and loss of revenue while the point was unusable. In addition, it felt "invisible" to many EV drivers, since only those who knew its station directly used it; It didn't have a broad platform to make it visible to new users.
- **Specific needs:** An intuitive **back-office** interface where he can **register new stations** and edit information (address, power, connector types, hours of operation, price per kWh); ability to **change the status** of a station (available, under maintenance, occupied) in real time, informing users.

Persona 4: Ana Rodrigues – Charging Station Maintenance Technician

- **Age:** 29 years old
- **Profession:** Electrotechnical Technician (employee of a charging infrastructure company)
- **Biography:** Ana works in the technical support team that keeps dozens of charging stations operational throughout the region. Their day-to-day task involves remotely monitoring the status of equipment and also going to the field to perform repairs or preventive maintenance. She is trained in electrical systems and also has the IT notion to deal with station management software. Ana understands that, for drivers to trust the charging network, it is crucial that the equipment is working perfectly and that any problems are resolved quickly.
- **Objectives with the use of the platform: Remotely monitor the status** of each charging station under your responsibility in one place; **notify** when a point has an error or goes offline, so that he can act before too many users are affected; put a station in **maintenance mode** via the system, preventing new users from trying to use it while it is being repaired;
- **Frustrations:** Ana's hands were tied when the old tools didn't show enough detail – sometimes a station stopped working, with clients complaining on social media. The lack of a centralized dashboard forced it to check several systems: one for the online status of the machines, the other for technical records. She also got frustrated when she needed to schedule preventive maintenance and had no way to block reservations at that time, causing conflict with a driver who arrived to load. Every minute of an inactive station without warning represented angry drivers and damage to the company's image.
- **Specific needs: Unified operations dashboard** that lists all stations and their current status (available, under maintenance, occupied); possibility to **insert notes or signs** in the stations (e.g. "maintenance scheduled today 2pm-4pm") visible both internally and to users, ensuring transparency; **technical history** of each station, including electrical data and past incidents, to aid in the analysis of recurring faults

User Stories

1. **As an electric vehicle driver, I want to search for nearby charging stations** to easily find where to recharge my car when the battery is low.
    a. **Acceptance criteria:**
        i. Since I am logged in to the app and allow access to my location, when I open the search screen I should see on an **interactive map** the available charging stations nearby, with a clear indication of which ones are **free or occupied** in real time.
        ii. The system should allow me **to filter** the stations by criteria such as charging speed (slow/fast) displaying only the points that meet my preferences.
        iii. When selecting a specific station on the map or list, I must view details such as address, distance, number of spaces, power offered, fares and reviews from other users.
2. **As an electric vehicle driver, I want to book a time slot at a charging station in advance** to ensure that I will have a charging point available when I arrive, especially at peak times or long trips.
    a. **Acceptance criteria:**
        i. In the interface of the chosen station, there should be the option to **schedule a time**: I enter the desired start date and time (among the available times) and the estimated duration of the session or amount of load desired.
        ii. The system should not allow double booking at the same point/time – that is, it **prevents overbooking**. If I try to book an already busy slot, I'll receive a warning and the ability to choose another time or station.
        iii. When confirming the booking, I should receive an **immediate confirmation** with the details (station, time booked, confirmation code or QR code) and the booking should be in my history/agenda in the app.
3. **As an electric vehicle driver, I want to unlock the charger and start recharging through the app** to start charging my car hassle-free when arriving at the reserved or unoccupied station.
    a. **Acceptance criteria:**
        i. When I arrive at the station, I should be able to select my active reservation (or the desired free point) in the app and activate the **"Start Charging" function**, which remotely unlocks the connector corresponding to my space.
        ii. If I have a reservation, the system must validate that I am within the reserved time slot and, if so, allow the session to start (if I arrive too early or too late, I may need to adjust or notify unavailability). If I'm using a walk-in point, the app should check if the point is free and then release it immediately.
        iii. During the recharge, I want to see in the app the **real-time data** of the session: energy (kWh) already supplied, accumulated cost so far, and elapsed time. There should be the option to **log out** of the app when I want to finish recharging; by doing so, the charger should be locked again and my results recorded.

4. **As an electric vehicle driver, I want to track my consumption and recharge history on a personal dashboard** to understand my spending, my energy usage, and better plan my future recharges.
    a. **Acceptance criteria:**
        i. In my profile or dashboard, I should have access to a **detailed history of all charging sessions** carried out via the platform: listing date and time, location (season), duration, energy consumed (kWh) and cost paid in each session.
        ii. The system should present useful **aggregate statistics**, such as total kWh consumed in the last month, total spending on recharges in a period, average spending per session, and even how many CO_2 emissions were avoided by using electricity (an extra environmental data, if available).
        iii. I want to be able to **export or share** a report of my history (e.g. to send to the company in the case of enterprise use) and filter the information by period (last week, 6 months, current year, etc.). In addition, associated payment information (receipts) must be available for each top-up.
5. **As a charging station operator, I want to register a new station on the platform** so that it is visible to drivers and I can start offering my charging services through the unified system.
    a. **Acceptance criteria:**
        i. The platform must provide a **registration interface** where I enter the details of my station: location (address and GPS coordinates), number of charging points, types of connectors available at each point, power (kW) of each charger, operating hours and tariff (price per kWh or per time).
        ii. After submitting this information, the system must **validate basic data** (e.g. check that the coordinates match the address provided, that all required fields have been filled in correctly). There may be an approval process by the **PowerNest team,** if necessary, but once approved, the station should appear to all users in searches and maps.

6. **As a charging station operator, I want to edit the details of an existing station** so that I can keep its information up to date
    a. **Acceptance criteria:**
        i. The operator can select any of their registered stations and open an “Edit Station” form.
        ii. The form displays current station data pre-populated in the fields.
        iii. The operator can modify any attribute: tariff, operating hours, connector availability, or station status.
        iv. The system validates changes (e.g., prices must be positive numbers, hours must be valid ranges).
        v. Upon saving, updates are applied immediately and reflected in both the admin dashboard and the driver app’s search results.
        vi. A change log entry is recorded for auditing (timestamp, operator ID, fields changed).

7. **As a maintenance technician, I want to mark a station as "under maintenance" temporarily** so that drivers are informed of unavailability during repairs, avoiding attempts to use it while working on site.
    a. **Acceptance criteria:**

i. From the admin panel, I should be able to **change the status** of a specific charging point (or the entire station) to "Maintenance" or "Unavailable" by adding, if possible, an explanatory note (e.g., *"Preventive maintenance from 2 pm to 4 pm today").*
ii. When the maintenance status is activated, drivers viewing that station in the app should see it clearly indicated as unavailable for that period (for example, a different icon or "On Maintenance" message) and **prevent new** reservations or logins for the duration of the maintenance.
iii. Any driver who has an upcoming active booking at that station during the scheduled period should receive an **immediate warning/alert** about the conflict. (E.g.: "Your 3pm reservation has been canceled due to maintenance. Please reschedule at another station.") – This criterion ensures that no one is caught by surprise. As soon as I, as a technician, resolve the issue and clock in as available again, the system resumes its normal use.

8. **As an electric vehicle driver, I want to pay for the recharge directly through the app** to complete the process in a practical way and receive receipts automatically, without needing additional interactions.
   a. **Acceptance criteria:**
      i. At the end of a charging session, the app must calculate the **amount to pay** based on the station rate (per kWh or minute, as defined) and present a summary of the charge before confirming the payment.
      ii. The system must support **integrated payment methods**: for example, use of a previously registered credit card, account debit, or digital wallet (such as Apple Pay/Google Pay, if applicable). The driver must be able to choose or register the payment method of their choice only once, and then the next transactions occur automatically using that saved method.
      iii. After the payment is successfully processed, the user must receive a **digital receipt** in the app itself, containing details of the transaction: station, date/time, energy consumed, amount charged, and payment method used. If the payment fails for some reason, the system must notify the user immediately and offer alternatives (try another method, or forward payment at the station if available).

2.3 Project epics and priorities

2.3.1 Iteration 2

1. **EP01 - Station Discovery & Filtering**
   Search nearby stations on a map, filter by connector/speed/network, see real-time availability.
2. **EP02 - Booking & Reservation Management**
   Booking interface with time slot selection, conflict detection, reservation confirmation.

2.3.2 Iteration 3

1. **EP03 - Charging Session Execution**
   Unlock charger, validate time, show real-time charging data, stop charging session.
2. **EP04 - Station Registration & Editing**
   Operators can register
   Operators can edit stations with required details.
3. **EP05 - Maintenance Mode & Availability**
   Maintenance tech can mark stations as under maintenance, notify users, and restore availability.

2.3.3 Iteration 4

1. **EP06 - Consumption History & Dashboard**
   Charging session history, export, dashboard with stats.
2. **EP07 - In-App Payment & Receipt**
   Payment processing, stored payment methods, receipts.

3 Domain model

Our problem defines a necessity of a system that manages charging stations, therefore we need one that allows a user to make reservations in a specific charging station, and that allows a manager to manage their stations.
Therefore, as functional requirements, we need a system that allows:
For Owners:

- Create charging stations in their companies
- Create accounts to manage a specific charging station

For Managers

- Changing status of a charging station
- Cancel reservations made
- See reservations made
- Edit timetables of charging stations

For Users

- Create an account
- Add a car to their account
- See charging stations in a within a selected area
- See the timetables of charging stations and their current status, such as availability.
- Book a slot in a given car to a given charging station at a specific date and time
- Cancel existing bookings
- Pay for a booking with credit card

And finally Charging Stations to unlock only to users that booked a slot or if a charging station is not booked for that time slot.

Our solution can be achieved with the following entities:

- **Users**, which will be the main actor of our system and that could be a normal user, a charging station worker or an owner of a company;
- **Companies**, composed of Charging Stations and their owner/managers;
- **Stations**, that belong to a Company and has Reservations;

- **Charger**, which represent each specific charger in a station which allows each charging station to have a status;
- **Reservations**, made for a specific charging station for a set date and time.
- **Receipt**, useful to give a user a history of previous purchases.

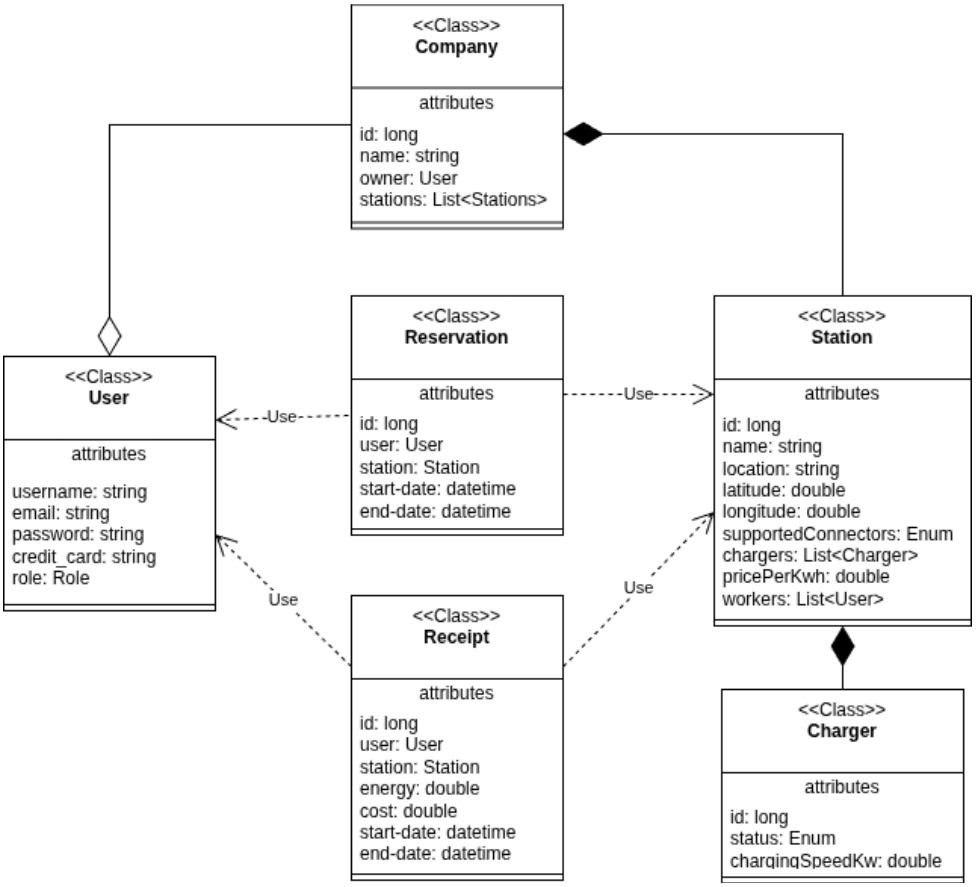

Figure 1. Class diagram of our system

4 Architecture notebook

4.1 Key requirements and constraints

Our solution has one external system, an external payment system to process payments and one hardware dependency, the charging stations themselves.
We need an architecture that answers both the requests of the users and that receives/provides information required for proper integration between our system and the charging stations. This architecture must allow for different charging station providers, in order to allow an aggregation of different companies into our system with a minimal adaptation. <Verificar esta parte com o grupo>
As non-functional requirements, our system is expected to have managers with an account consistently open during their time tables. As such, we must offer a system that is focused on availability, maintaining a high uptime with minimal downtime in case of failures, this can be achieved with a containerized application that restarts on failure for minor issues. For other extreme cases where a restart is not enough, a manual check-in of the system may be necessary.

We also expect a system that is able to handle a great number of concurrent requests. Especially for critical user interactions such as allowing users to charge their car even if our system is passing through complications. However, in an initial phase, the scalability doesn't need to be a critical issue, as such we chose to follow a monolithic architecture as it is able to answer our problems efficiently with a small technical debt in an initial lifecycle of our system.

4.2 Architecture view

Our solution, that accommodates these necessities, is based on three main monolithic building blocks: the backend, that encapsulates the business logic behind our solution, and the frontend that presents itself as a bridge between the user/manager and our system, and, as well, the charging stations and our solution.
The backend consists of three main layers:

- **Controller layer**, that receives/answers requests to be processed, and passes them into the service layer.
- **Service Layer**, where all the business logic is implemented, consists of 5 services:
  - **User Service**, to manage the users in the system
  - **Company Service**, to manage the companies
  - **Station Service**, to manage the stations and their reservations.
  - **Payment Service**, to serve as a bridge between our system and an external payment service, to process reservation payments
  - And finally an **Auth Service**, to simply allow registrations/login/logout into our solution.
- **Repository Layer**, where all the repositories are defined creating an ORM Layer with our relational database.

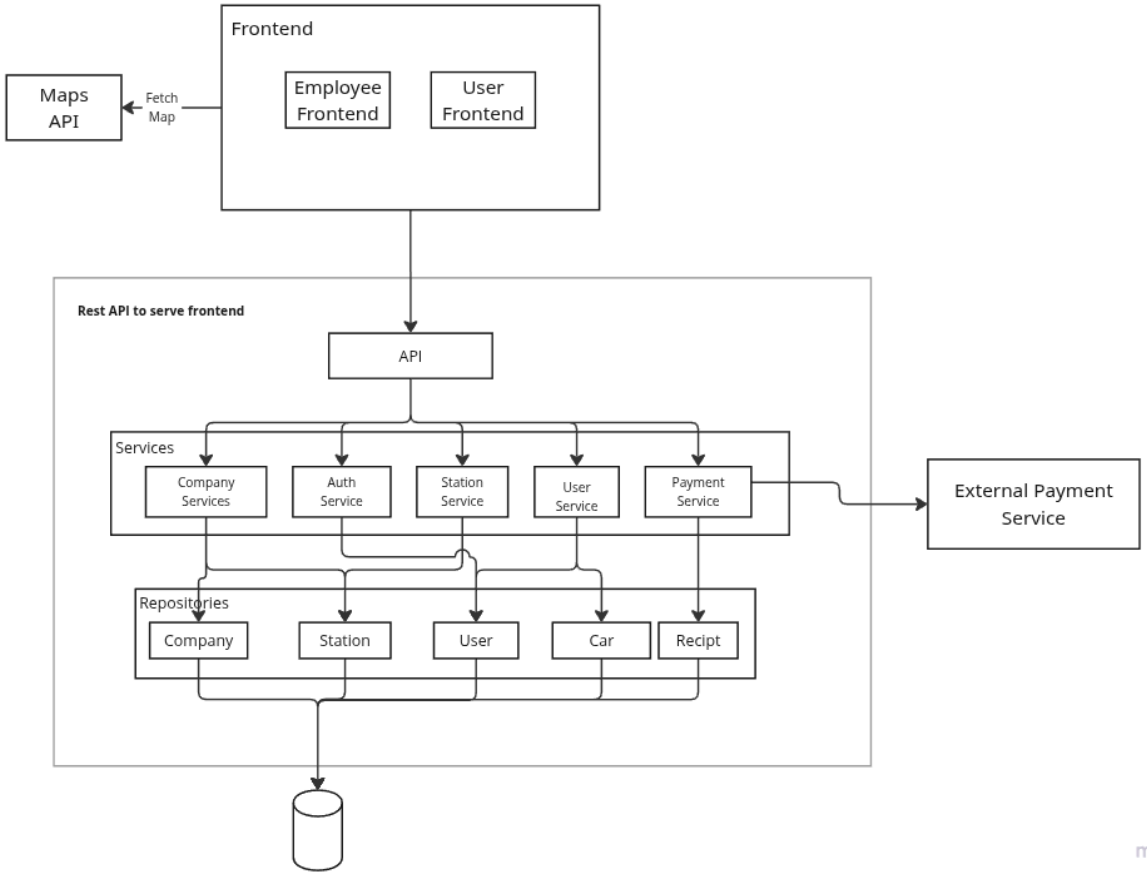

Figure 2. Monolithic architecture intended for our system

With our analysis, we achieved this monolithic architecture, which, as expected will allow us to create a MVP and to start up quickly our application into an usable one, nonetheless, in the future, it may need to be massively revamped, depending on the number of users of the application to ensure the performance and quality criteria imposed.

The following Sequence Diagrams explain the lifecycle of the main actions available in our application, searching stations and booking a charger, and the usage of one of the chargers associated with our application:

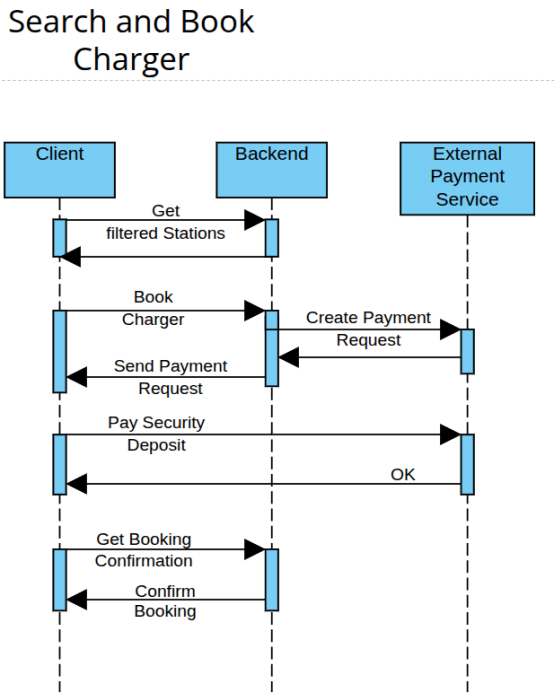

Figure 3. Booking a charger

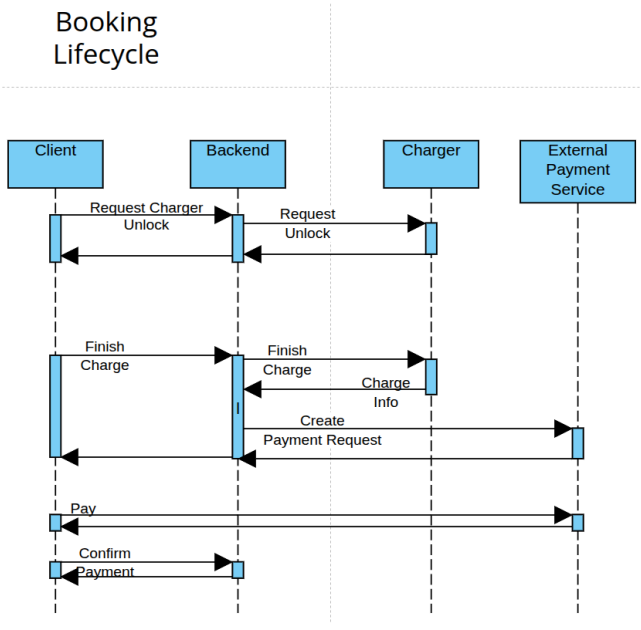

Figure 4. Unlock and using a Charger

In Figure 4., we show that we expect the users to pay a security deposit in order to prevent abuse of our system.

In Figure 3., we show that the final payment is only made after the user finishes charging, with a potential security reimbursement if the final payment comes short of it.

4.3 Deployment view

Figure 5. refers to our deployment diagram, where we have a frontend made in React, a backend built on spring boot and a PostgreSQL database to store all the data from our system.

To have observability over our system, we implemented two main services, Nagios and Prometheus. The latter is responsible for fetching important statistical data such as the number of calls for each endpoint, etc. that is supported by Grafana to create a user interface to allow us to see those statistics over time which allow us to quickly glance up and understand cases for failures, hence improving maintainability of the system.
The former is used as a health check for all our services, in order to catch downtimes as fast as possible since it sends an email notification to us, the developers, when such events are detected, hence it improves the downtimes of our system as a whole.

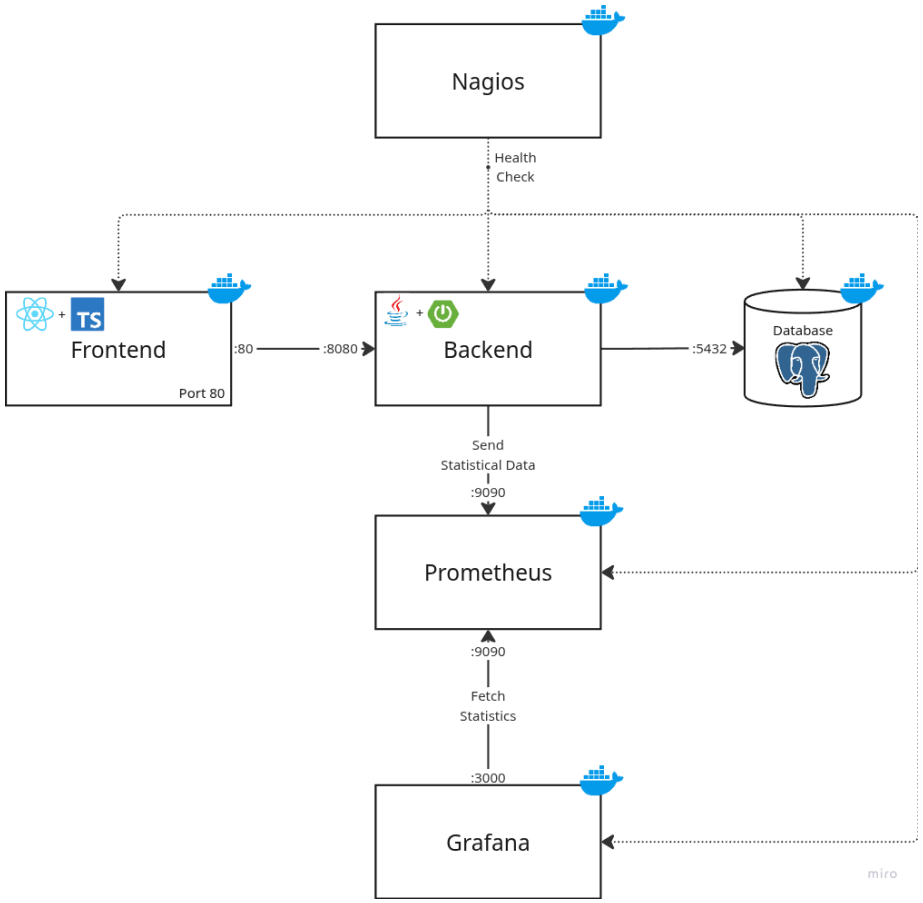

Figure 5. Deployment Diagram

5 API for developers

Our API, accessible through http://deti-tqs-09.ua.pt:8080/api/**/*, is divided by the four main entities of our system:

- Stations: Where stations can be created, updated and deleted or fetched.
- Users: Which users can use to register, login and fetch the user data associated to the sent token (/me endpoint)
- Reservations: Where users can make and remove reservations to chargers
- Companies: A simple endpoint that only presents to programmers an endpoint to fetch the companies registered in our app.

For more information, our API documentation can be accessed through http://deti-tqs-09.ua.pt:8080/swagger-ui/index.html