

TQS: Product specification report

Diogo Costa[112714], Bruno Tavares[113372], Francisco Pinto[113763], André Alves[113962]
v2025-06-05

1	Introduction	1
1.1	Overview of the project	1
1.2	Known limitations	1
1.3	References and resources	2
2	Product concept and requirements	2
2.1	Vision statement	2
2.2	Personas and scenarios	2
2.3	Project epics and priorities	2
3	Domain model	3
4	Architecture notebook	3
4.1	Key requirements and constrains	3
4.2	Architecture view	3
4.3	Deployment view	3
5	API for developers	3

1 Introduction

1.1 Overview of the project

We will develop a web application with a Spring Boot backend to manage electric vehicle charging. The project will follow DevOps and Software Quality Assurance practices, including CI/CD, automated testing, and code analysis. It will feature functionalities such as charger search, booking, and payment.

1.2 Known limitations

Graphical analysis for stations, and a way to predict the disponibility of a station in the future. Monitor system health.

2 Product concept and requirements

2.1 Vision statement

Our product focuses on providing a solution for electric vehicle (EV) owners to manage their charging needs more effectively. The ChargerControl app allows users to monitor and control their EV charging process, ensuring that they can charge their vehicles at the most convenient times and rates. The app also provides real-time data on charging status, energy consumption, and cost savings, making it an essential tool for EV owners looking to optimize their charging experience.

2.2 Personas and scenarios

Persona: Intermittent Charger User (Zezinho, 24 years)

Description: Zezinho is an IT Support Technician living in the Aveiro suburbs. Commutes daily to work (approx. 30 km round trip) in his Peugeot e-208, makes weekend trips occasionally, values time efficiency and planning. Comfortable with apps and digital services, uses Waze and MBway regularly.

Goals: Zezinho wants to find available and nearby chargers quickly, especially during commutes or when low on battery. Track charging status and history to manage costs and battery health. Book charging slots in advance to avoid waiting times..

Needs: He requires a user-friendly app interface with a clear map and filtering options, Integrated payment system, and reliable information about charging speed, and estimated waiting time. Support and contact in case of charger malfunction or issues during charging

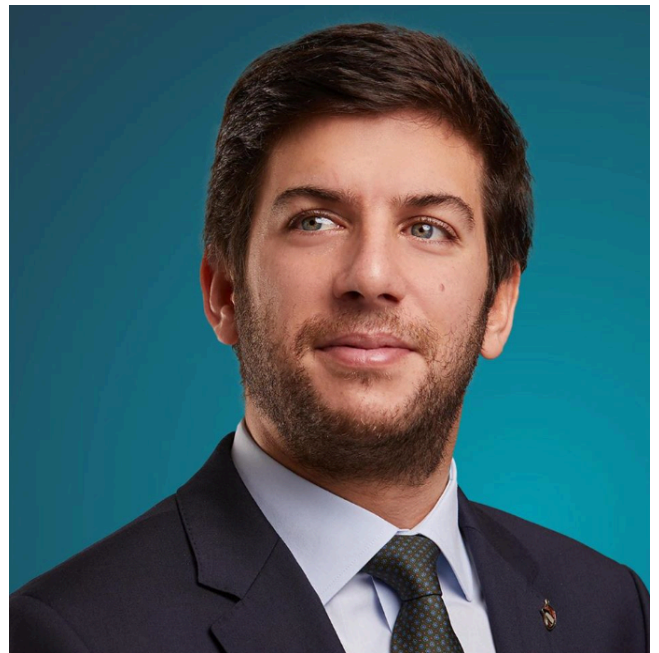


Persona: Station Operator (Chicão, 35 years)

Description: Chicão is a Charging Station Operator & Technician living in the Porto suburbs, he manages several charging points across urban and semi-urban locations, is also responsible for ensuring uptime, reporting to a municipal or private energy provider.

Goals: Chicão wants to register and configure new charging stations quickly and accurately, Monitor the operational status of all assigned stations in real-time, Get alerts or reports about failures, unusual behavior, or excessive idle time

Needs: He requires a centralized dashboard to manage all stations (status, bookings, usage), tools to register new chargers, including metadata like location, type, power output, and availability schedule. Access to Logs and reports for audits, performance reviews, and environmental tracking (e.g., CO₂ savings).



Scenario: Zezinho Checks Charging Station Information

Zezinho, is preparing for a trip across the city. Before starting his journey, he wants to find the most convenient charging station to stop at. Zezinho opens the Charger app on his phone and browses the map to explore nearby charging stations. He taps on one of the stations he's considering, and the app redirects him to the station information page. On this page, Zezinho can clearly see useful details such as the total number of chargers available at the station, how many are currently occupied, and how many are free. In addition to real-time data, the app also shows projected availability for the next few hours, allowing Zezinho to make a well-informed decision. The page includes usage statistics and trends, such as the busiest hours and average waiting times, which help Zezinho avoid potential delays. With this information, Zezinho feels confident in choosing the station that best fits his schedule and avoids unnecessary waiting.

Scenario: Station Operator Manages Station Status and Views Statistics

Chicão, a station operator responsible for several charging stations in the city, logs into the Charger admin panel to monitor their performance. From the dashboard, he can see a full list of all stations under his management, each with up-to-date statistics such as usage rates, downtime history, and current availability. While reviewing the data, Chicão notices that one of the stations is experiencing recurring technical issues and needs maintenance. Using the system interface, he selects the affected station and disables it temporarily for the next three hours. As soon as he confirms the action, the system automatically sends a notification to all users who had an active booking at that station, informing them of the temporary closure and suggesting alternative stations nearby. Chicão also has the option to re-enable the station manually if maintenance finishes early. This level of control allows him to take proactive actions based on the station's performance, ensuring better service quality and avoiding user frustration.

2.3 User Stories

Epic : User account management

User Story 1 : User registration

Priority: High

Cost: 2

As a Charger user,

I should be able to create a new account in a system,
so that I can use the system more effectively.

Acceptance Criteria:

The user must be able to access the registration page.

He should be able to input information needed for the registration.

The system must validate the information .

If some information is incorrect , the system should display an appropriate error message.

Epic :User account management

User Story 2:User Login

Priority: High

Cost: 2

As a Charger user,

I should be able to login into the system,
so that I can access essential information.

Acceptance Criteria:

The user must be able to access the login page.
He should be able to enter their register email and password.
The system must validate the information .
If some information is incorrect , the system should display an appropriate error message.

Epic :User analytics

User Story 3:See personal information

Priority: High

Cost: 5

As a Charger user,

I must be able to see my information,
so that I can manage my account.

Acceptance Criteria:

The user must be able to go to their personal page.
The system must show information about the account .
The user can alter account information .
He can see his consumptions, locations of charge, charge duration, money spent and CO2 savings and personal vehicles .

Epic : Account management

User Story 4:Add personal vehicle

Priority: High

Cost: 4

As a Charger user,

I must be able to add a vehicle to the account,
so I can charge the car.

Acceptance Criteria:

In the personal page the user must press the add car button.
He has to fill a form with information about the car.
The user should be able to delete the car.
The user should be able to update information about the car.

Epic : Station information

User Story 5: See station location

Priority: High

Cost: 4

As a Charger user,

I must be able to locate a charging station,
so I can go there and charge the car.

Acceptance Criteria:

The user must access the map in the website.

He can click on a station on the map.

It should show the distance and approximate time of travel.

Epic : Station information

User Story 6: See station Information

Priority: High

Cost: 8

As a Charger user,

I must be able to obtain information about the charging station,
so I can make a decision.

Acceptance Criteria:

The user must be able to access the station page .

It should show information about the charging station like total chargers, occupied chargers and overall statistics.

The system must show this in the current time and future time.

Epic : Booking Options

User Story 7: Book a slot in a station

Priority: High

Cost: 8

As a Charger user,

I must be able to reserve a slot in a charger,
so I can guarantee that he can charge his car.

Acceptance Criteria:

The user must be able to access the station page .

He should be able to press the add reservation.

He can choose a time from a minimum of 2 hours and a maximum of a month.

He should be able to pay.

Epic : Payment system

User Story 8: Define a method of payment

Priority: High

Cost: 4

As a Charger user,

I must be able to choose my method of payment,
so I can best suit my situation.

Acceptance Criteria:

The user must be able to access his personal page .

In the payment options he should be able to choose between pay per use or subscription options.

If he chooses pay per use, every time he uses a charging station he must pay .

If he chooses a subscription, he has various options of wattage per month with some discount .

Epic : System management

User Story 9: Station management

Priority: High

Cost: 4

As a Station Operator,

I must be able to see station statistics,
so I can make decisions on what to do.

Acceptance Criteria:

The Station Operator must be able to see all stations .

He must see information about the stations.

He must be able to disable a station or enable it.

If he disables a station, the system must send a message to every user that has a booking in it to inform them.

He can also choose to disable the station for a determined time

Epic : System management

User Story 10: Add new Station

Priority: High

Cost: 4

As a Station Operator,

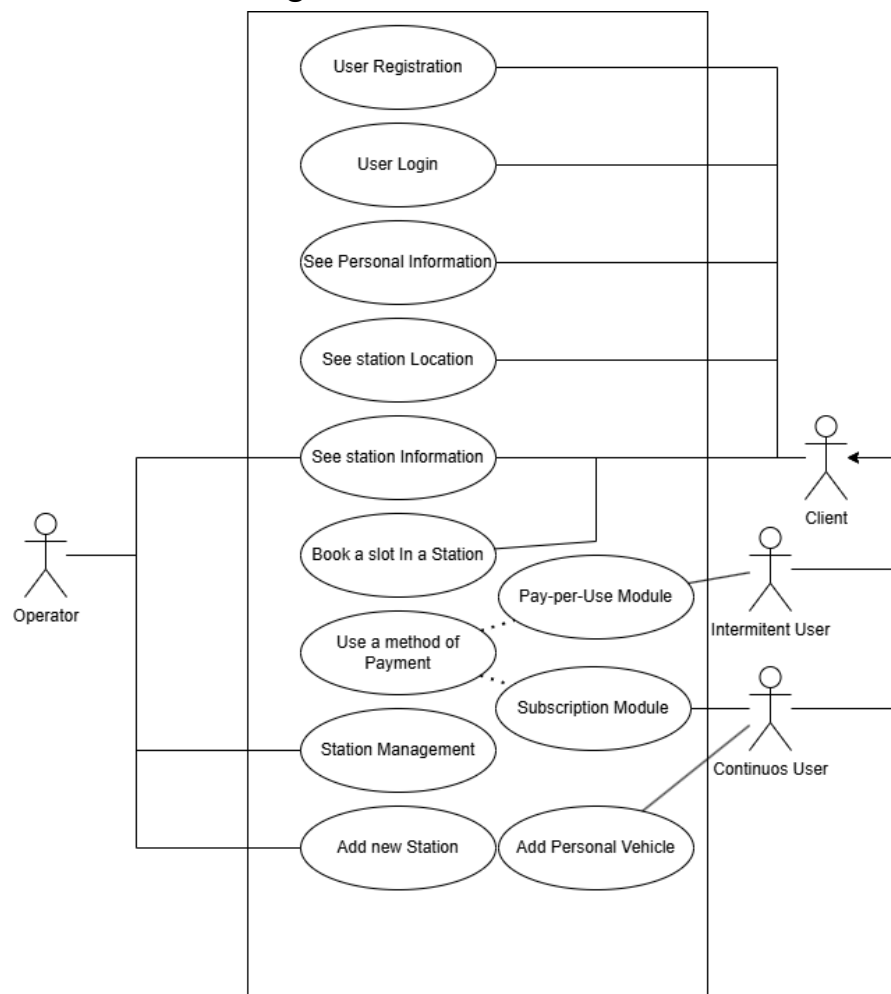
I must be able to add a new station to the system,
so the clients have more options.

Acceptance Criteria:

The Station Operator must go to the add station page.

He must fill a form with station information.

Use Cases Diagram:



2.4 Project epics and priorities

Project Epics:

- User account management
- User analytics
- Station information
- Booking Options
- Payment system
- System management

Iterations:

Iteration 0:

- Define the product concept: Personas, main scenarios, Epics and User Stories.
- Team resources setup: code repository, documents space.
- Begin backlog usage.

Iteration 1:

- Define system architecture.
- Define the SQA tools and practices.
- Create CI Pipeline.
- Start working on the product specification report.
- Create a prototype of the UI.

Iteration 2:

- Develop the main User stories.
- Define testing strategy.
- Create the QA manual.
- Start working on the API

Iteration 3:

- Set up the CD pipeline.
- Continue the development of the user stories.
- Continue various tests implementation.

Iteration 4:


- Complete user stories implementation.

Complete QA manual.
Final deployment.

Example of a Issue:

Add personal vehicle

+ Add

 Apps

Descrição

As a Charger user,



I must be able to add a vehicle to the account,

so I can charge the car.
















Acceptance Criteria:

- In the personal page the user must press the add car button.
- He has to fill a form with information about the car.
- The user should be able to delete the car.
- The user should be able to update information about the car.

Child work items

 ...  +

100% Done

Type	Chave	Resumo	Prioridade	Responsável	Estado
	SCRUM-59	Create Controller	 Medium	 André	CONCLUÍDO ▾
	SCRUM-60	Create Service	 Medium	 André	CONCLUÍDO ▾
	SCRUM-61	Create Repository	 Medium	 André	CONCLUÍDO ▾
	SCRUM-62	Develop FrontEnd	 Medium	 Francisco Pinto	CONCLUÍDO ▾
	SCRUM-63	Add Endpoint	 Medium	 Bruno Tavares Meixedo	CONCLUÍDO ▾

Example of a Sub-Issue:

| 11

Create Controller

+ Add

Concluído ▾

✓ Concluído



⚡ Improve work item

Descrição

Adicione uma descrição...

Details

Responsável

André

Assign to me

Etiquetas

None

Principal

SCRUM-10 Add personal vehicle

Sprint

None +1

Story point estimate

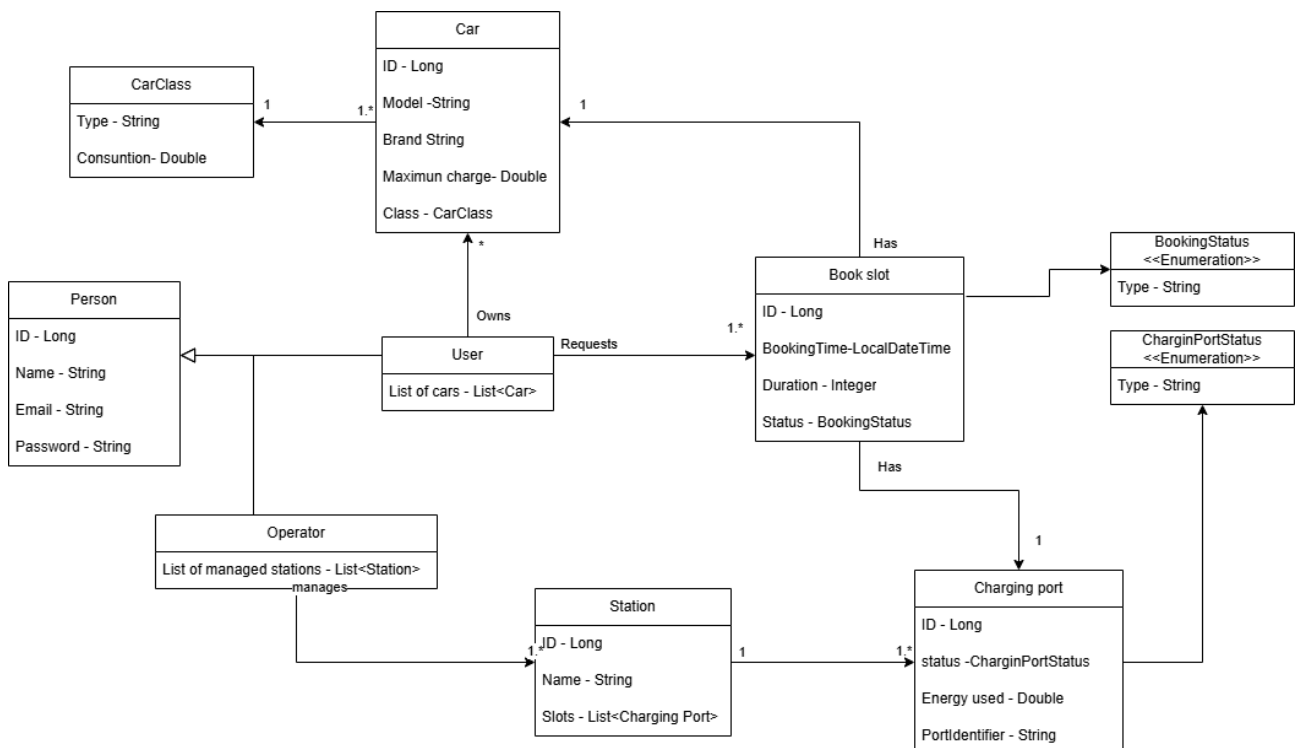
None

Priority

None

Versões de correção

Nenhum



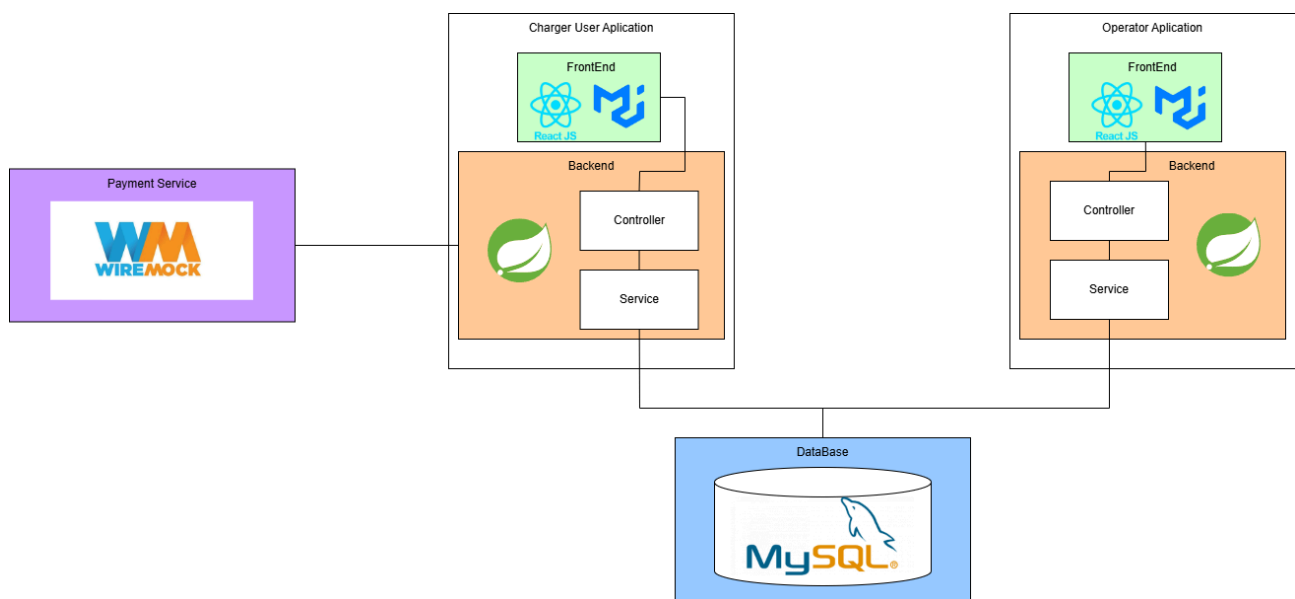
3 Domain model

4 Architecture notebook

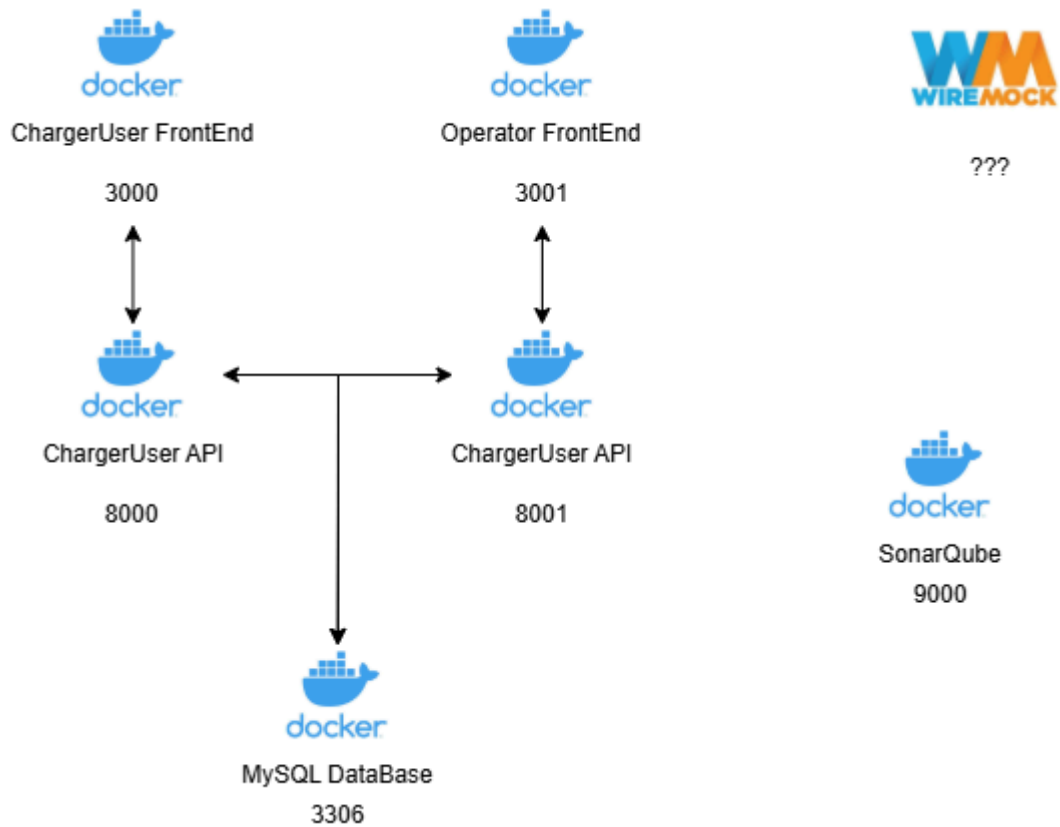
4.1 Key requirements and constraints

- The system must be capable of dealing with several users at once.
- The system must have an authentication system to distinguish user types.
- The different users must have a different platform.
- The system must be user-friendly.
- The system is integrated with a payment system.

4.2 Architecture view



4.3 Deployment view



5 API for developers

API for User APP:

Cars APIs for managing cars			^
GET	/apiV1/cars/{carId}	Get a car by its ID	v
PUT	/apiV1/cars/{carId}	Update an existing car	v
DELETE	/apiV1/cars/{carId}	Delete a car by its ID	v
GET	/apiV1/cars/user/{userId}	Get all cars for a specific user	v
POST	/apiV1/cars/user/{userId}	Add a new car to a user	v
GET	/apiV1/cars/all	Get all cars	v
Charging Ports APIs for managing charging ports			^
GET	/apiV1/chargingports/station/{stationId}	Get all charging ports for a station	v
POST	/apiV1/chargingports/station/{stationId}	Create a new charging port for a station	v
GET	/apiV1/chargingports/status/{status}	Get all charging ports by status	v
GET	/apiV1/chargingports/station/{stationId}/stats/energy	Get total energy used by all charging ports for a station	v
DELETE	/apiV1/chargingports/{portId}	Delete a charging port by its ID	v

User Authentication APIs for user registration and login			^
POST	/apiV1/user/register	Register a new user	▼
POST	/apiV1/user/login	Login an existing user	▼
GET	/apiV1/user/all	Get all users	▼

Bookings APIs for managing bookings			^
PUT	/apiV1/bookings/{id}/status	Update booking status	▼
POST	/apiV1/bookings	Create a new booking	▼
GET	/apiV1/bookings/{id}	Get booking by ID	▼
DELETE	/apiV1/bookings/{id}	Cancel a booking	▼
GET	/apiV1/bookings/user/{userId}	Get bookings by user ID	▼
GET	/apiV1/bookings/station/{chargingPortId}	Get bookings by station ID	▼
GET	/apiV1/bookings/station/{chargingPortId}/range	Get bookings by station ID and time range	▼

Stations APIs for managing stations			^
GET	/apiV1/stations/{id}	Get station by ID	▼
PUT	/apiV1/stations/{id}	Update an existing station	▼
DELETE	/apiV1/stations/{id}	Delete a station by ID	▼
GET	/apiV1/stations	Get all stations	▼
POST	/apiV1/stations	Create a new station	▼
GET	/apiV1/stations/name/{name}	Get station by name	▼

API for Operator App:

Stations APIs for managing charging stations			^
GET	/apiV1/stations/{id}	Get a station by ID	▼
PUT	/apiV1/stations/{id}	Update a charging station	▼
DELETE	/apiV1/stations/{id}	Delete a station by ID	▼
GET	/apiV1/stations	Get all charging stations	▼
POST	/apiV1/stations	Create a new charging station	▼
GET	/apiV1/stations/available/{available}	Get stations by availability status	▼
Charging Ports APIs for managing charging ports			^
GET	/apiV1/chargingports/station/{stationId}	Get all charging ports for a station	▼
POST	/apiV1/chargingports/station/{stationId}	Create a new charging port for a station	▼
GET	/apiV1/chargingports/status/{status}	Get all charging ports by status	▼
GET	/apiV1/chargingports/station/{stationId}/stats/energy	Get total energy used by all charging ports for a station	▼
DELETE	/apiV1/chargingports/{portId}	Delete a charging port by its ID	▼

6 Links VM

- 6.1 UserApp: <http://192.168.160.7:3000>
- 6.2 OperatorApp: <http://192.168.160.7:3001/operator>
- 6.3 SonarQube: <http://192.168.160.7:9000>
- 6.4 Swagger UserApp: <http://192.168.160.7:8080/swagger-ui/index.html>
- 6.5 Swagger OperatorApp: <http://192.168.160.7:8081/swagger-ui/index.html>