TQS: Product specification report

Hugo Ribeiro [113402], Eduardo Lopes [103070], Rodrigo Abreu [113626], Joao Neto [113482]
v2025-05-08

1 Introduction

1.1 Overview of the project

This project was developed in the context of the Test and Quality Software (TQS) course, which emphasizes building robust, testable, and maintainable software systems. Throughout the project, we applied principles of agile development, user-centered design, and quality assurance.

Our product, ChargeHub, is a digital platform designed to improve the experience of electric vehicle (EV) drivers and facilitate efficient management for station operators. It aims to simplify the process of discovering, reserving, and managing EV charging sessions, while also providing essential administrative tools. Below lies a comprehensive list of our core functionalities:

Station Discovery
- Search for nearby charging stations based on location.
- Filter stations by charger type, availability, and cost.

Slot Booking & Scheduling
- Reserve a charging slot in advance.
- Receive notifications and booking confirmations.

Charging Session Management
- Start/stop charging sessions using a reservation token.
- View real-time session data including time, cost, and energy used.

Payment Integration
- Review and pay for charging sessions within the app.
- Station operators can access financial reports and track payments.

User Profiles & Charging History
- Track past charging sessions and total energy consumption.

Back Office Operations for Station Operators
- Update station data such as pricing and availability.
- Report user misconduct.
- Admin features include account creation and station assignment.

ChargeHub reflects the TQS course objectives by combining well-structured software design with comprehensive feature testing and quality control measures.

1.2 Known limitations

In the **Station Discovery** module, features like the map view and real-time route tracing to charging stations were excluded from the MVP. Although visualizing stations on a map and guiding users to their destination would have significantly enhanced usability, the development effort required to integrate mapping APIs and manage location data in real time was beyond the scope of the initial release. Similarly, the ability to plan long-distance trips with intermediate charging stops was deferred due to its high complexity and limited necessity for everyday use cases.
Within **Slot Booking & Scheduling**, the system does not currently support operator-defined custom schedules or advanced notification flows. While users receive immediate booking confirmation, reminders such as those scheduled 15 minutes before a reserved slot or alerts about

charger disruptions were not fully implemented. These features were deprioritized in favor of ensuring reliable booking and confirmation workflows.
In the **Charging Management** area, several enhancements were considered but ultimately left out. For instance, users are not yet able to monitor real-time metrics like battery level, cost accumulation, or energy consumption during a session. Additionally, the option to restart a recently stopped session was not included, as it involves maintaining complex session state and reconnection logic. A live dashboard for station operators to monitor charging sessions was also scoped out for future development, as it required significant backend support.
From a **sustainability and user history** standpoint, the platform does not currently track estimated CO_2 emissions saved—an optional feature designed to give users a sense of environmental impact. Similarly, administrative dashboards to analyze platform-wide trends and KPIs were planned but not pursued in this version, as the focus remained on delivering a strong user-facing experience.
Lastly, in the **account and backoffice management** domain, full administrative control over user accounts and a user-led charger feedback mechanism were not implemented. While operators can manage station data and create new accounts, broader moderation and analytics capabilities were reserved for future iterations. Additionally, the ability for users to permanently delete their accounts was not completed in time for the initial release.
In summary, the omitted features reflect a careful balancing of ambition and feasibility. While not part of the current release, they represent valuable directions for future development, ensuring that **ChargeHub** remains a scalable and feature-rich platform as it evolves beyond the scope of the TQS course.

1.3 References and resources

During the development of our project, we consulted several key resources that informed and supported our work. The Portuguese government's fuel price information site, https://precoscombustiveis.dgeg.gov.pt/, served as an example and inspired our approach to presenting and structuring data.
In addition to this example, we used various libraries, frameworks, and APIs to build and enhance our solution. Here are some of the links we found useful for the development of this project:

https://github.blog/enterprise-software/ci-cd/build-ci-cd-pipeline-github-actions-four-steps/
https://docs.github.com/en/actions/about-github-actions/about-continuous-deployment-with-github-actions
https://www.openstreetmap.org/#map=6/40.82/-6.11
https://openrouteservice.org/

2 Product concept and requirements

2.1 Vision statement

ChargeHub is a digital platform aimed at enhancing the experience of electric vehicle (EV) drivers and supporting the operational needs of charging station operators. Our vision is to simplify EV charging by offering a seamless interface for discovering, reserving, and managing charging

sessions, while also empowering station operators and administrators with tools to maintain station data, monitor usage, and ensure service quality.
The core business problem ChargeHub addresses is the growing complexity and fragmentation in accessing EV charging infrastructure. EV drivers often face challenges in locating available chargers, reserving slots, and managing payments. At the same time, station operators lack centralized, easy-to-use tools for managing their stations and ensuring optimal service delivery.

To tackle this, ChargeHub delivers the following key features:

- **Station Discovery**: Locate nearby charging stations filtered by connector type, availability, and cost.
- **Slot Booking & Scheduling**: Reserve charging slots in advance and receive timely notifications.
- **Charging Session Management**: Securely start/stop charging using reservation tokens and monitor charging progress.
- **Payment Integration**: In-app summaries and payments for completed sessions; revenue tracking for station operators.
- **User Profiles & History**: Detailed tracking of past sessions and energy usage over time.
- **Back Office Operations**: Admin and operator dashboards for managing stations, accounts, and reports.
- **Authentication & Account Management**: Profile management, access control, and secure account handling.

Changes to the Initial Plan

During development, a key change occurred in the implementation of the payment system. The original plan was to integrate with **Ifthenpay**, a Portuguese payment service provider. However, due to **compatibility issues with the university firewall and network restrictions**, we were unable to complete the integration reliably in the academic environment.
As a result, we pivoted to using **Stripe**, a widely adopted, developer-friendly payment API. Stripe's comprehensive documentation, robust testing tools, and support for international standards made it an ideal alternative. While the switch required some redesign of our backend integration and payment flow, it ultimately enhanced the system's flexibility and scalability for future deployments.

State of the Art and Concept Development

ChargeHub draws inspiration from leading EV platforms like PlugShare, ChargePoint, and Ionity but differentiates itself by integrating booking, session management, and back-office operations into a single cohesive ecosystem. While many platforms focus primarily on station discovery or payment, ChargeHub emphasizes the full end-to-end experience, from planning a session to resolving station-side issues, making it particularly useful in semi-public and private charging networks.
The concept for ChargeHub was developed iteratively through team discussions and analysis of the existing market (https://precoscombustiveis.dgeg.gov.pt/). Our team adopted a **user-centered design approach**, frequently validating assumptions against common EV use cases. Informed prioritization decisions were made to help shape our initial epics and user stories.

2.2 Personas and scenarios

Personas

EV Driver

Sofia is a thirty-nine-year-old Marketing Consultant based in Lisbon, Portugal. She recently made the switch from a diesel car to an electric vehicle. Though confident in her decision, she wants to ensure it makes long-term financial sense, especially given how often she drives for work.

She frequently travels across city and suburban areas to meet clients and attend events. Despite her marketing background, navigating EV-related logistics is new to her. She often finds herself concerned about where and when she can charge her vehicle and how charging costs compare to the fuel expenses of her previous car.

Sofia is motivated to make smarter decisions about EV ownership. She would benefit from a user-friendly app that shows available charging stations along her route, provides real-time charger availability, and offers expense comparisons. This would give her peace of mind during daily commutes and work trips.

Motivation: Sofia wants to improve her EV experience and be sure her switch from diesel is both eco-friendly and economically viable.

Station Operator

Miguel is a forty-seven-year-old station manager who oversees two Galp service stations in Aveiro, Portugal. He is married and has worked in the fueling industry for more than two decades. Recently, Miguel installed electric chargers at both locations to modernize his business and meet the rising demand for EV infrastructure.

Despite this upgrade, Miguel struggles with making the most of the chargers. He needs real-time insights into charger usage, availability, and performance. His goal is to keep the stations visible and attractive to EV drivers and avoid chargers being offline or underused.

Miguel is looking for a smart management system that lets him track usage patterns, schedule maintenance, and update station status instantly. He also wants a tool to advertise his chargers' availability to drivers nearby.

Motivation: Miguel wants to maximize charger usage, attract more EV drivers, and run his stations efficiently.

Admin

José is a twenty-seven-year-old System Administrator working in Aveiro, Portugal. He is married and plays a key role in managing the user accounts and access permissions for an EV charging platform.

José's work revolves around ensuring that the right people have access to the right tools. He is responsible for creating and managing accounts for station operators, assigning stations to them as needed, and updating platform access to reflect organizational changes. He also oversees EV driver accounts and ensures that user data stays clean, current, and compliant with platform rules.

Although he doesn't deal directly with technical maintenance or infrastructure, José's work is essential for keeping operations organized and secure. He values clarity and simplicity in admin tools, and needs efficient interfaces to carry out tasks quickly and accurately.

Motivation: José wants to manage users and station assignments effectively and ensure the platform reflects the latest organizational setup at all times.

Scenarios

Sofia searches for a charger nearby

- Sofia notices her EV battery is getting critically low while driving to an important client meeting. She opens the app and is greeted by a clean interface showing nearby charging points. The app detects her current location and displays available fast chargers within reach. She quickly filters by availability and connector type, selects the nearest compatible charger, and the app provides real-time navigation to the station.

Sofia checks for chargers along her planned route

- Sofia is planning a work trip from Lisbon to Porto and wants to ensure charging won't disrupt her schedule. She opens the trip planner in the app and enters her destination. The system calculates the best route and overlays charging stations along the way. She taps on a few stations to compare details and estimated costs, helping her plan a smooth and stress-free trip.

Sofia reserves a charger in advance

- Knowing she'll arrive in a busy downtown area during peak hours, Sofia wants to avoid the risk of waiting in line. She searches the app for chargers in the area and sees that only a few are available. She selects a station, reviews the terms, and reserves a slot. Upon arrival, she checks in via the app and begins charging immediately.

José suspends an EV driver account due to policy violation

- José receives an automated alert about repeated misuse of charging sessions by an EV driver, such as overstaying after sessions end. He reviews the user's charging history and confirms multiple violations. Through the admin panel, he temporarily suspends the driver's account.

José adds a new station operator and assigns them their stations

- A new charging station operator joins the platform. José logs into the admin system and creates the operator's account. He selects the appropriate role and access permissions, then assigns the relevant stations. After final review, he activates the profile. The new operator now has access to manage pricing, monitor charger status, and view usage analytics for their assigned stations.

Miguel configures pricing and details for his assigned charging stations

- Miguel has recently been assigned a new set of EV chargers at his Galp stations. He logs into the management portal for the first time to complete the initial setup. He selects each station and inputs key details, such as the type of connectors available, maximum charging power and pricing. Once the configuration is complete, the stations become visible to EV drivers through the platform's app.

2.3 Project epics and priorities

EPIC 1: Station Discovery

User Stories:

- **As an EV Driver**, I want to filter stations by charger type, availability, location, and cost so that I can find stations that match my technical needs and budget.
    - **Priority/Estimate:** High, 2
    - **Acceptance Criteria:**
        - be able to apply filters by charger type
        - be able to apply filters by availability
        - be able to apply filters by location
        - be able to apply filters by cost
        - be able to see only stations matching selected filters

- **As an EV Driver**, I want to search for charging stations near my location so that I can access stations within a short distance from me.
    - **Priority/Estimate:** Medium, 2
    - **Acceptance Criteria:**
        - be able to allow location access
        - be able to see a list of charging stations
        - be able to view stations ordered by proximity

- **As an EV Driver**, I want to see charging stations on a map so I can easily locate them.
    - **Priority/Estimate:** Low, 3
    - **Acceptance Criteria:**
        - be able to grant location access
        - be able to open map view
        - be able to see station markers on the map

- **As an EV Driver**, I want to trace a route to a nearby charging station so that I can visualize the path to it.

    - **Priority/Estimate:** Optional, 4
    - **Acceptance Criteria:**
        - be able to select a charging station
        - be able to select “Get Directions”
        - be able to view a route from current location to station on map

- **As an EV Driver**, I want to see the closest charging stations when planning a trip, so that I can see where I should stop to charge my vehicle.
    - **Priority/Estimate:** Optional, 5
    - **Acceptance Criteria:**
        - be able to enter trip start and end points
        - be able to start trip planning
        - be able to see stations near the route in list
        - be able to see stations near the route in map
        - be able to view distance from departure point to the stations

---

EPIC 2: Slot Booking & Scheduling

User Stories:

- **As an EV Driver**, I want to book a charging slot in advance so that I can ensure availability.
    - **Priority/Estimate:** High, 3
    - **Acceptance Criteria:**
        - be able to select a charging station
        - be able to choose a time slot and confirm booking
        - be able to reserve the selected time slot
        - be able to prevent double-booking of that slot
        - be able to receive booking confirmation with charger type, time, and token

- **As a Station Operator**, I want to set custom schedules for my charging stations, so that I can control when each station is available for use.
    - **Priority/Estimate:** Optional, 4
    - **Acceptance Criteria:**
        - be able to open schedule management interface
        - be able to define available time slots for each charger
        - be able to define unavailable time slots for each charger

- **As an EV Driver**, I want to receive notifications 15 minutes before my booked time starts and when it starts, so that I am reminded to arrive at the charging station on time.
    - **Priority/Estimate:** Optional, 2
    - **Acceptance Criteria:**
        - be able to receive a notification 15 minutes before booking starts
        - be able to receive a notification at the start time of the booking

- **As an EV Driver**, I want to receive a notification if a charger I booked becomes disabled before my scheduled time, so that I can make alternative arrangements in advance.
    - **Priority/Estimate:** Low, 2
    - **Acceptance Criteria:**

    - be able to receive notification if charger becomes unavailable before the scheduled time
    - be able to receive prompt with options to rebook at another station or cancel booking

---

EPIC 3: Charging Management

User Stories:

- **As an EV Driver**, I want to use my reservation token in the app to unlock the charger when I arrive, so that I can start my charging session securely and only during my reserved slot.
    - **Priority/Estimate:** High, 3
    - **Acceptance Criteria:**
        - be able to input reservation token in the app
        - be able to remotely unlock the corresponding charger
        - be able to begin the charging session after unlocking

- **As an EV Driver**,I want to see real-time charging status including battery percentage, current cost, and energy consumed, so that I can monitor and manage my charging session according to my needs and budget.
    - **Priority/Estimate:** Optional, 4
    - **Acceptance Criteria:**
        - be able to open session details in the app
        - be able to view live battery percentage
        - be able to view real-time cost
        - be able to view energy (kWh) consumed

- **As a Station Operator**, I want to monitor all active and historical charging sessions, so that I can ensure operational visibility, detect issues, and analyze usage patterns
    - **Priority/Estimate:** Optional, 4
    - **Acceptance Criteria:**
        - be able to view current charging sessions with user info, charger status, duration, and energy consumed
        - be able to access past charging sessions
        - be able to apply filters to historical data

- **As an EV Driver**, I want to stop the charging session from the app at any time, so that I can charge only the amount I need and control the cost.
    - **Priority/Estimate:** High, 2
    - **Acceptance Criteria:**
        - be able to select “Stop Charging” in the app
        - be able to stop power delivery immediately
        - be able to view a session summary with cost, time, and energy used

- **As an EV Driver**, I want to restart the charging session from the app if I stop it accidently, so that I can continue charging without needing to make a new reservation.

    - **Priority/Estimate:** Optional, 2
    - **Acceptance Criteria:**
        - be able to select “Restart Charging” in the app
        - be able to resume charging using the same reservation

---

EPIC 4: Payment Integration

User Stories:

- **As an EV Driver**, I want to receive the payment summary in the app when my charging session ends, so that I can easily review and pay for the charging cost directly through the app.
    - **Priority/Estimate:** High, 4
    - **Acceptance Criteria:**
        - be able to view payment summary when session ends
        - be able to see total amount due
        - be able to see energy consumed
        - be able to see charging duration
        - be able to select a payment method
        - be able to complete payment within the app

- **As a Station Operator**, I want to access financial reports per charging session and verify their payment status, so that I can monitor revenue and ensure that all sessions have been correctly billed and paid.
    - **Priority/Estimate:** Medium, 4
    - **Acceptance Criteria:**
        - be able to open financial reporting section
        - be able to view each charging session’s payment status
        - be able to view amount charged and payment timestamp

---

EPIC 5: User Profiles & Charging History

User Stories:

- **As an EV Driver**, I want to view my charging history and total energy consumption over time, so that I can track my usage patterns and manage my charging behavior more efficiently.
    - **Priority/Estimate:** Medium, 4
    - **Acceptance Criteria:**
        - be able to open the "Charging History" section
        - be able to view past charging sessions with dates and locations
        - be able to view energy consumed (kWh), session duration, and cost

- **As an EV Driver**,I want to track my estimated CO_2 savings based on my charging activity, so that I can understand the environmental impact of using electric vehicles instead of fuel-based alternatives.

  - **Priority/Estimate:** Low, 4
  - **Acceptance Criteria:**
    - be able to access the sustainability section in user profile
    - be able to view estimated CO_2 emissions avoided in kg or tons

- **As an Admin**, I want to view aggregated user statistics and charging behavior trends, so that I can analyze platform usage and optimize the service for users.
  - **Priority/Estimate:** Optional, 5
  - **Acceptance Criteria:**
    - be able to open analytics or reporting section
    - be able to view total users
    - be able to view average energy per session and peak usage times
    - be able to view total CO_2 saved across the platform

EPIC 6: Backoffice Operations

User Stories:

- **As a Station Operator**, I want to update my station's information such as availability and pricing, so that users always see accurate data when booking a charging slot.
  - **Priority/Estimate:** Medium, 3
  - **Acceptance Criteria:**
    - be able to access a station’s details
    - be able to update availability, pricing, and charger information
    - be able to save changes and reflect them immediately in the user-facing app

- **As an Admin,** I want to create station operator accounts, so that new employees can manage charging stations assigned to them.
  - **Priority/Estimate:** High, 3
  - **Acceptance Criteria:**
    - be able to fill in required details for a new operator account
    - be able to allow operators to access the platform with appropriate permissions
- **As an Admin,** I want to assign or remove stations from specific station operators, so that I can manage responsibilities across the network effectively.
  - **Priority/Estimate:** High, 3
  - **Acceptance Criteria:**
    - be able to open an operator’s profile in the admin dashboard
    - be able to assign or remove stations
    - be able to update operator’s access based on assigned stations

- **As an Admin**, I want to manage all user accounts, including EV drivers and station operators, so that I can enforce platform rules and support operations effectively.
  - **Priority/Estimate:** Low, 4
  - **Acceptance Criteria:**
    - be able to view list of all users in the admin dashboard

        - be able to search for specific accounts
        - be able to edit, deactivate, or delete user accounts

- **As an Admin**, I want to view system-wide reports and statistics, so that I can monitor platform performance and usage trends.
    - **Priority/Estimate:** Optional, 4
    - **Acceptance Criteria:**
        - be able to access the analytics section
        - be able to view KPIs like total sessions, revenue, energy delivered, and user growth

- **As a Station Operator**, I want to report users for misconduct such as not paying or damaging equipment, so that the platform can take appropriate action.
    - **Priority/Estimate:** Medium, 3
    - **Acceptance Criteria:**
        - be able to access a problematic session from the operator dashboard
        - be able to click “Report User”
        - be able to provide a reason and submit the report to the admin team

- **As an EV Driver**, I want to report chargers I use, so that other drivers can benefit from my feedback.
    - **Priority/Estimate:** Optional, 2
    - **Acceptance Criteria:**
        - be able to visit the charger’s detail page after a session
        - be able to submit a report or comment about the charger

---

EPIC 7: Authentication & Account Management

User Stories:

- **As a user of the platform,** I want to be redirected to the correct dashboard based on my role after login, so that I can access the appropriate features and information for my responsibilities.
    - **Priority/Estimate:** High, 4
    - **Acceptance Criteria:**
        - If I log in as an EV Driver, I should be redirected to the EV Driver dashboard.
        - If I log in as a Station Operator, I should be redirected to the Station Operator dashboard.
        - If I log in as an Admin, I should be redirected to the Admin dashboard.
        - The system must identify my role accurately upon login.
        - The redirection must occur automatically and quickly after successful authentication.

- **As an EV Driver,** I want to update my account information, so that my profile stays accurate and secure.
    - **Priority/Estimate:** Medium, 3
    - **Acceptance Criteria:**
        - be able to navigate to the profile settings

        - be able to edit personal details
        - be able to save updated information

- **As an EV Driver,** I want to permanently delete my account, so that I can stop using the service and remove my personal data from the platform.
    - **Priority/Estimate:** Optional, 2
    - **Acceptance Criteria:**
        - be able to choose the option to delete my account
        - be required to confirm the deletion
        - have my account removed from the platform
        - be automatically logged out after deletion

- **As an EV Driver,** I want to register on the platform with my personal details, so that I can access EV charging services.
    - **Priority/Estimate**: High, 2
    - **Acceptance Criteria:**
        - be able to select the “Create an account” option
        - be able to enter personal details (e.g., name, email, password, etc.)
        - have my account created successfully
        - be able to log in after registration

3 Domain model

For our domain model, we tried to keep it as simple as possible without compromising any functionalities or requirements of our product. There are two main entities: **Client** (normal users of our application) and **Staff**. Regarding the Staff, there are two types of roles: **Staff.OPERATOR** (responsible for a station) and **Staff.ADMIN** (responsible for adding the new employees to the database with the corresponding roles, and adding new stations).
Which **Charger** is associated with a **Station**, and as a set of attributes necessary to assure a good use of them.
It is possible to book slots (**Booking**) and start a **ChargingSession** using the associated token to unlock the charger. The **Payment** entity helps the station operator validate if the client actually paid for the charging session, and allows the admin to create statistics relative to the usage of the chargers and the preferred paying methods.
The User can add a **Vehicle** to their account to facilitate the charger's choosing process (based on voltage, connector type, and charging type). There is also a **Notification** entity used to remind/notify the user when their charging session is about to start.
The domain model is presented in the image below:

The domain model for our MVP (Minimum Viable Product) has been designed with simplicity in mind, ensuring all essential functionalities and requirements are covered without introducing unnecessary complexity, just as you can see below:

4 Architecture notebook

4.1 Key requirements and constraints

Several critical concerns related to interoperability, responsiveness, platform diversity, and real-time operations shape this system's design. The following architectural characteristics were identified as essential to guide the software architecture and ensure it supports users' needs, operational requirements, and future extensibility.

The system supports multiple user-interfacing platforms, including:

- A web-based frontend built with ReactJS and LeafletJS for interactive map rendering.
- Potential integration with mobile platforms or lightweight embedded displays in the future.

To support this, the architecture is logically separated into a frontend and backend through standard HTTP REST APIs, ensuring loose coupling between presentation and business logic. This allows UI interfaces to evolve independently of backend logic, while reusing the same API services across platforms.

Some system components, such as booking confirmation or charging status, require real-time updates. This is achieved through backend services that track and manage data such as notifications and session progress. The Notification Module, composed of a controller, service, and repository, is responsible for delivering system messages and alerts and supporting real-time feedback mechanisms.

The system interfaces with physical infrastructure, particularly charging stations and possibly booking kiosks. These are abstracted through backend services like the Charge Service and Station Service, which isolate hardware-specific interactions from the rest of the business logic. This approach allows the software to remain hardware-agnostic and easily adaptable to changes in device models or communication protocols.

Given the involvement of multiple user roles (clients, staff, and admin), the architecture incorporates strict authentication and authorization mechanisms. These controls ensure that users can only access actions and data appropriate to their role. For example:

- Staff can update station statuses and charging sessions.
- Clients can book stations and view their vehicle data.

Security is embedded at both the API endpoint level and within business logic, minimizing the risk of unauthorized access or misuse.

Operating in domains such as mobility infrastructure or healthcare, the system must perform reliably even under heavy usage. The modular backend, coupled with efficient database interactions via Spring Data JPA and a reactive frontend, is designed to scale both vertically and horizontally as needed. Critical paths such as booking, session tracking, and notifications are optimized for low latency and high throughput, ensuring responsiveness even during peak load conditions.

To ensure that the system can be deployed consistently across different environments, the entire application (backend, frontend, database) is containerized using Docker. Docker containers encapsulate the environment, dependencies, and runtime configuration, allowing the system to run identically on local developer machines, staging servers, or production infrastructure. This significantly reduces the risk of deployment issues and simplifies environment management.

This architecture provides a solid foundation for delivering a robust, scalable, and extensible system. It balances user-facing interactivity, backend performance, security, and flexibility for future growth and platform expansion.

4.2 Architecture view

The system uses a client-server architecture, composed of a modular backend, a rich client-side frontend, and integrations with external services. A frontend interface communicates with backend services over HTTP, while the backend handles all business logic and data access responsibilities. An external routing and map provider supports geolocation and mapping needs. A payment gateway handles transactions securely. All components are logically decoupled and follow a clear separation of concerns.

The backend consists of several logical modules, each responsible for a specific functional domain. All modules follow a controller-service-repository pattern and communicate through internal calls. The main modules are:

- **Booking Module** - The Booking Module is responsible for managing all operations related to user bookings of chargers on a specific station. This module ensures that users can securely and efficiently reserve services within the system.
- **Staff Module** - The Staff Module handles operations related to system staff or administrative users. This module ensures that administrative functions are isolated and governed by strict access rules.
- **Client Module** - The Client Module manages all functionality related to application users (i.e., clients who interact with the system to book and charge vehicles). This module plays a central role in enabling end-users to access and utilize the system effectively.
- **Station Module** - The Station Module governs the behavior and status of charging stations. This module integrates closely with the map services to support location-based queries and decisions.
- **Notification Module** - The Notification Module is dedicated to sending real-time alerts, status updates, or confirmations to clients and staff. This module is essential for maintaining user awareness and timely communication within the system.
- **Charging Module** - The Charging Module handles the full lifecycle of charging sessions. This module integrates tightly with both the station and client modules to ensure secure, efficient, and trackable energy usage.

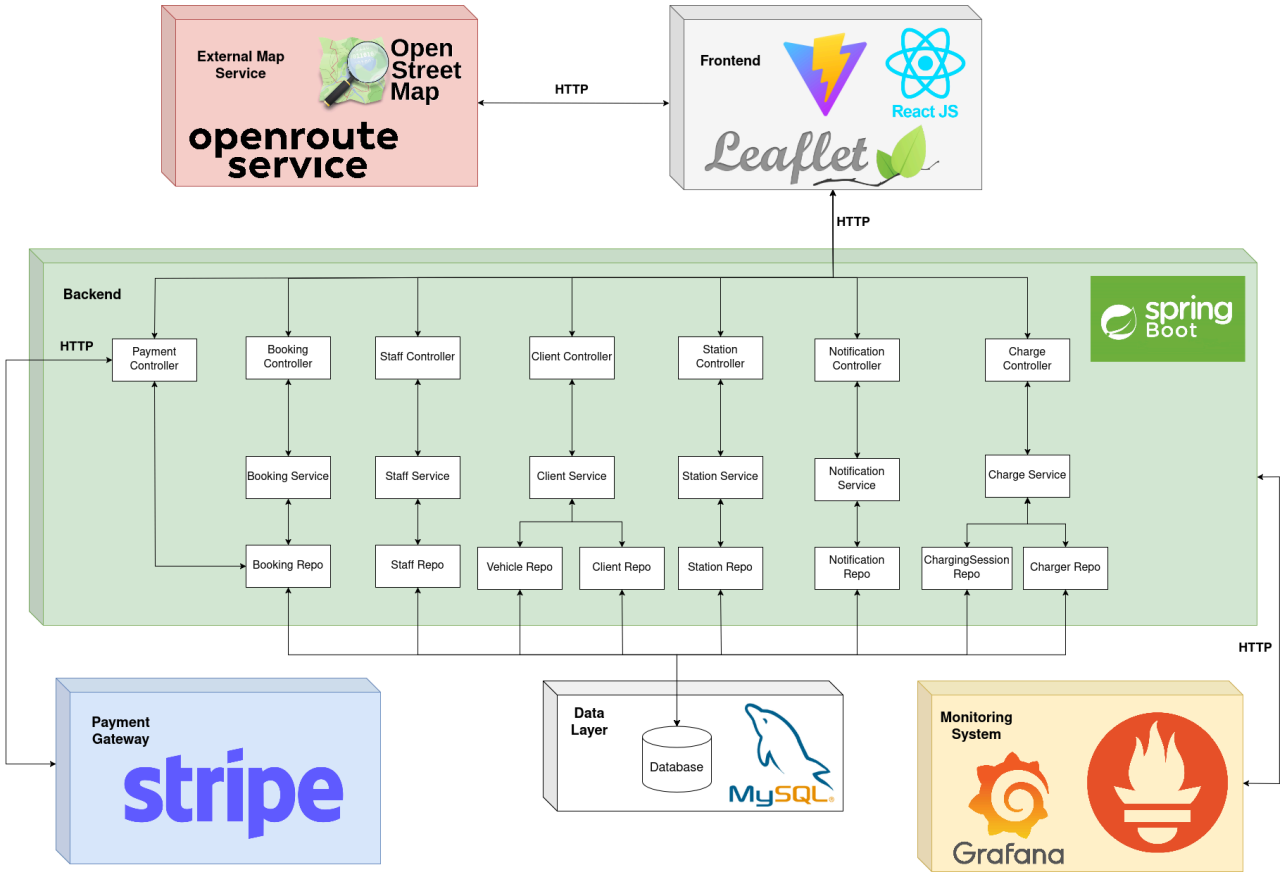
All backend modules interact with a central relational database via repository interfaces. The data model includes clients, vehicles, bookings, station information, staff, notifications, and charge sessions. This centralized data layer ensures consistency and referential integrity across components (all data is persisted in this single instance).

The frontend is a web-based interface that supports different types of users. The interface integrates with **Leaflet** for interactive maps, fetching real-time geospatial data from an external map provider.

Regarding external services, our app uses two:

- **Map Service**: Integrates with OpenRouteService and OpenStreetMap to offer real-time route data, distance calculations, and map tiles (HTTP APIs fetch map data and routes from OpenRouteService).

- **Stripe**: Supports online payments via secure HTTP communication with Stripe's payment gateways. Stripe's HTTP APIs enable secure transmission of payment transactions, offering robust encryption, fraud prevention, and global payment processing capabilities.



4.3 Deployment view

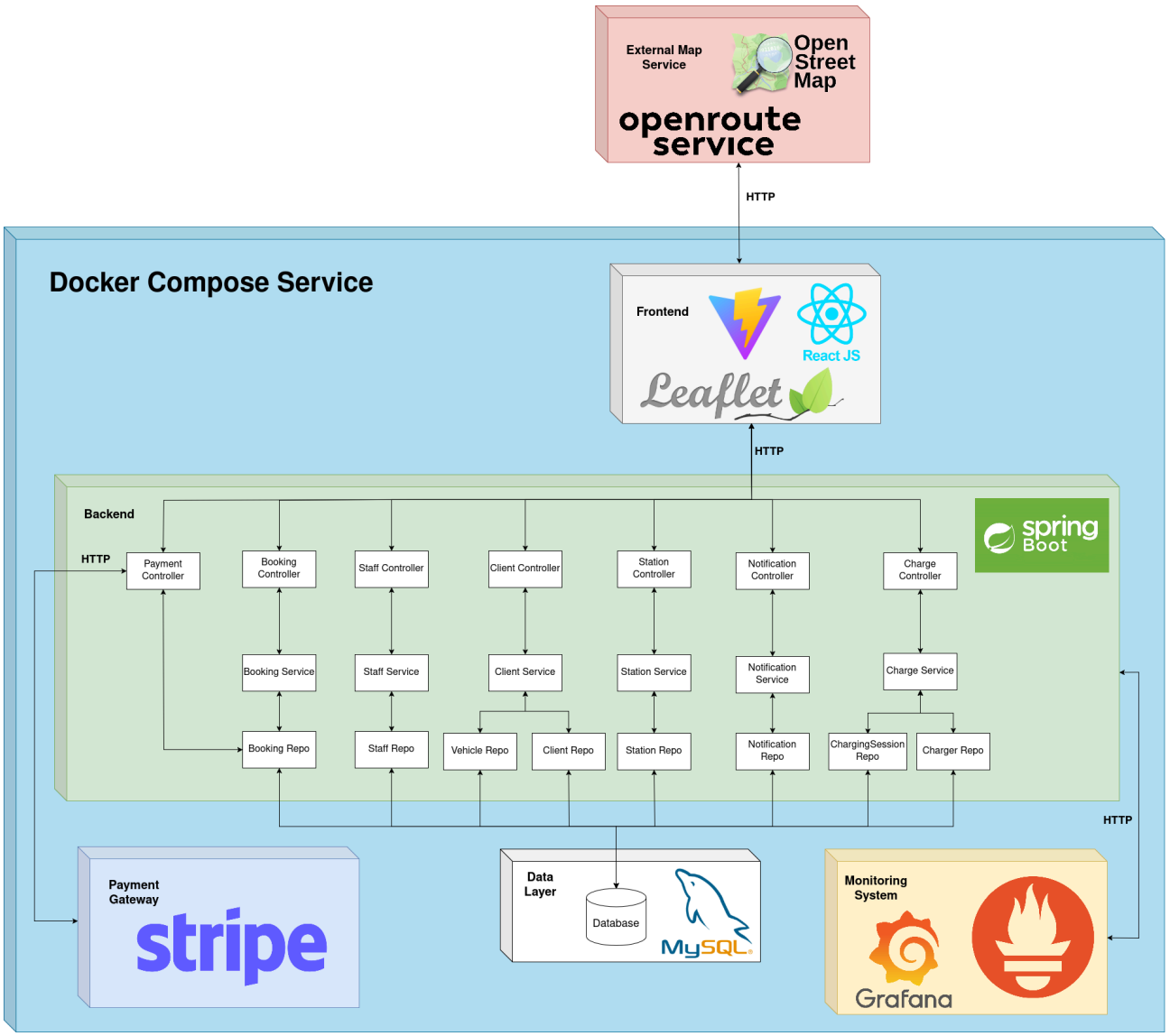
To simplify deployment and ensure consistency across development and production environments, the system is fully containerized using Docker. Each functional component of the system is deployed in its own isolated container, coordinated via Docker Compose, allowing for scalable, reproducible, and environment-agnostic deployments.

The system is divided into multiple Docker containers, each responsible for a specific subsystem:

- **Backend Container** - Runs the Spring Boot backend application, exposing its RESTful API over HTTPS. It handles business logic, database interaction, and external system integration.
- **Frontend Container** - Hosts the React.js web frontend (built using Vite), which provides the user interface for clients and staff to interact with the system.
- **Database Container** - A MySQL container (as shown in the diagram) acts as the persistent data layer for all application modules. It is accessed internally by the backend via a secure network.
- **Map Services** - While not deployed internally, the frontend makes HTTP calls to OpenStreetMap and OpenRouteService, which are considered external dependencies and accessed over the public internet.
- **Payment Gateway** - The system integrates with external payment providers, namely Stripe, which are not deployed internally but are securely accessed by the backend over

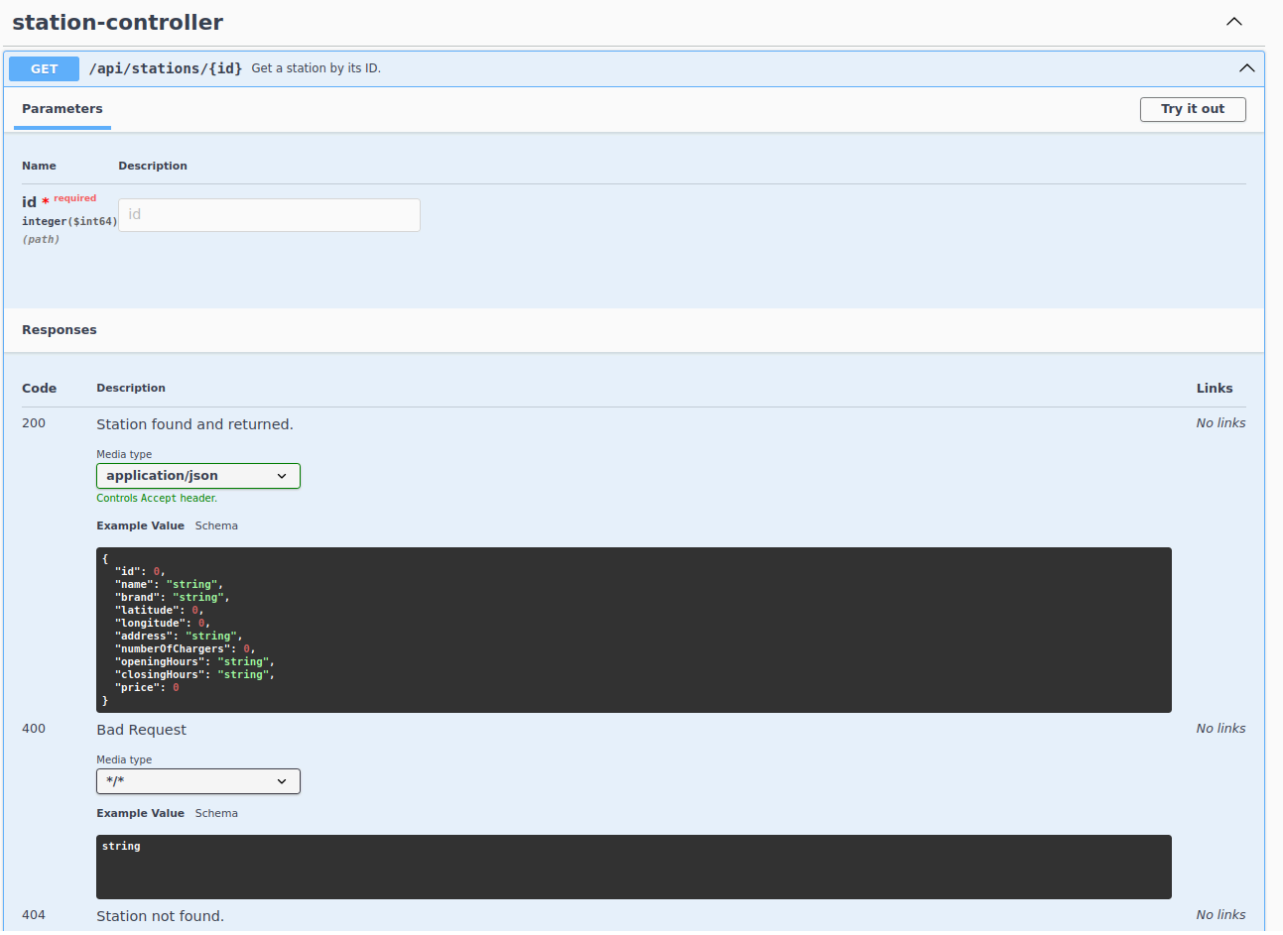
HTTP. These gateways handle payment authorization, transaction processing, and status feedback. The Payment Controller within the backend communicates with these gateways during charging sessions to initiate and validate payments, ensuring a seamless and secure payment experience for end users.

This containerized structure ensures the system can be easily deployed to any infrastructure, whether cloud-based, on-premise, or hybrid. The separation of roles, clean API boundaries, and containerized deployments also prepare the system for CI/CD automation, load balancing, and high availability configurations in the future.



5 API for developers

The API for developers follows best practices for RESTful API design, providing a resource-oriented interface to the core functionalities of the application. Standard HTTP methods (GET, POST, PUT) are used to perform operations on these resources, ensuring compatibility and ease of use for client-side developers. Authentication is handled through secure token-based mechanisms, and error handling adheres to standard HTTP status codes with informative response messages. Comprehensive documentation of all available endpoints, request/response formats, and usage examples is provided through an interactive Swagger interface, enabling developers to explore and test the API in real time. This approach ensures that integration is straightforward and well-supported throughout the development lifecycle.
This is possible to access from here (only inside the UA network or with VPN).



PUT `/api/stations/{id}` Update an existing station by ID.

Parameters Try it out

Name	Description
id * required integer($int64) *(path)*	id

Request body required — application/json

Example Value | Schema

```
{
  "name": "string",
  "brand": "string",
  "latitude": 0,
  "longitude": 0,
  "address": "string",
  "numberOfChargers": 0,
  "openingHours": "string",
  "closingHours": "string",
  "price": 0
}
```

Responses

Code	Description	Links
200	Station updated successfully. Media type: application/json Controls Accept header. **Example Value** | Schema	*No links*

```
{
  "id": 0,
  "name": "string",
  "brand": "string",
  "latitude": 0,
  "longitude": 0,
  "address": "string",
  "numberOfChargers": 0,
  "openingHours": "string",
  "closingHours": "string",
  "price": 0
}
```

400	Invalid input data. Media type	*No links*
404	Station not found.	*No links*

GET `/api/stations` Get a list of all stations.

Parameters Try it out

No parameters

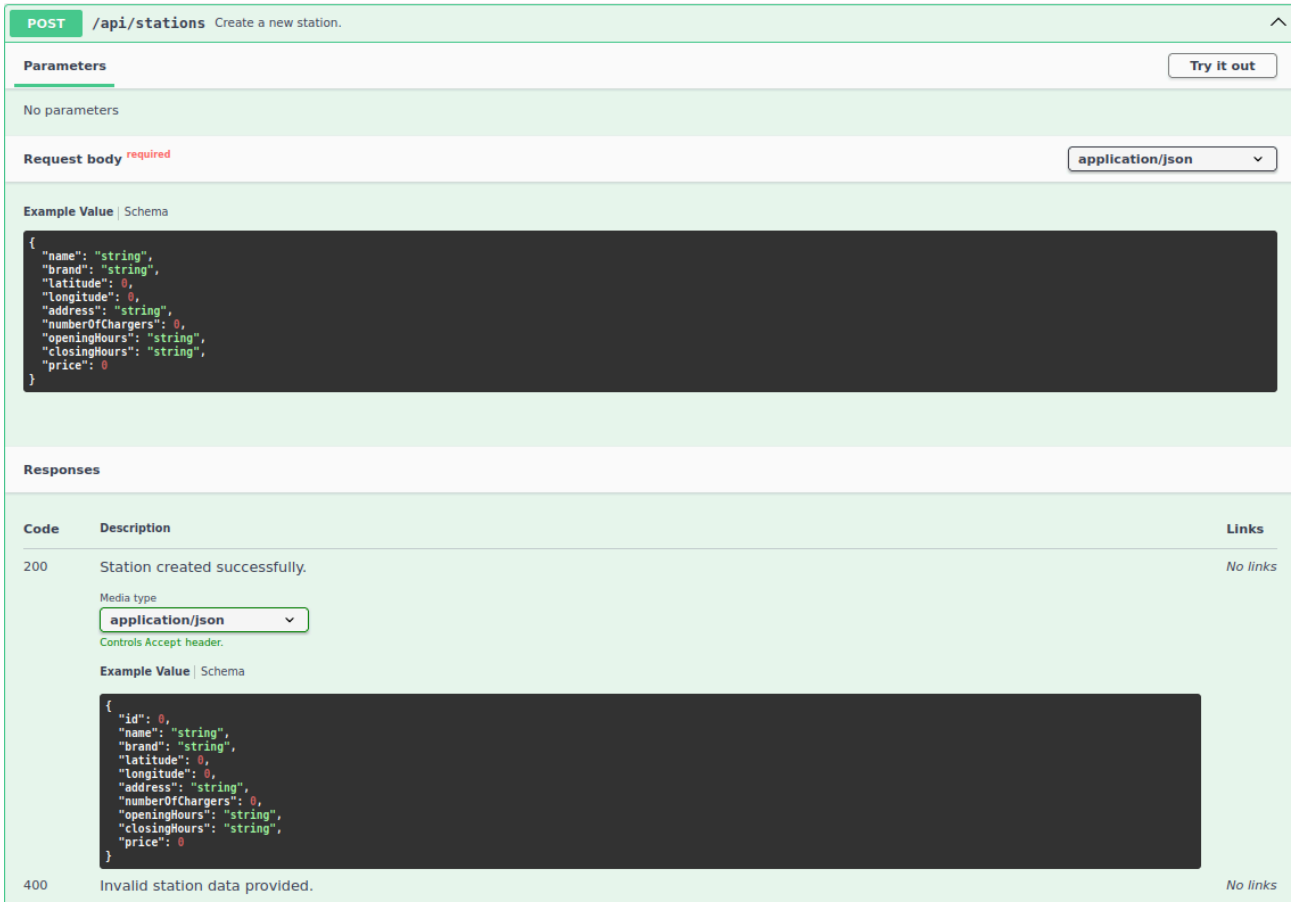
Responses

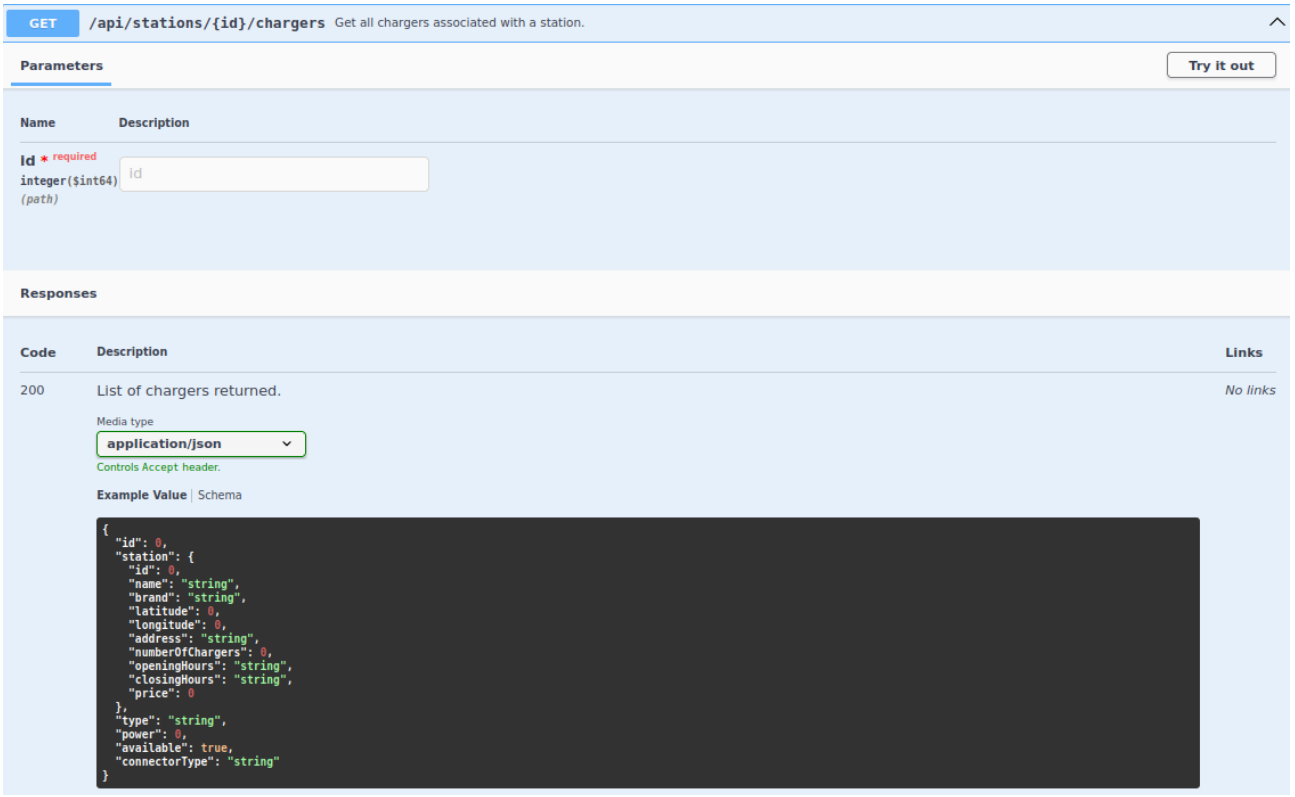
Code	Description	Links
200	List of stations retrieved successfully. Media type: application/json Controls Accept header. **Example Value** | Schema	*No links*

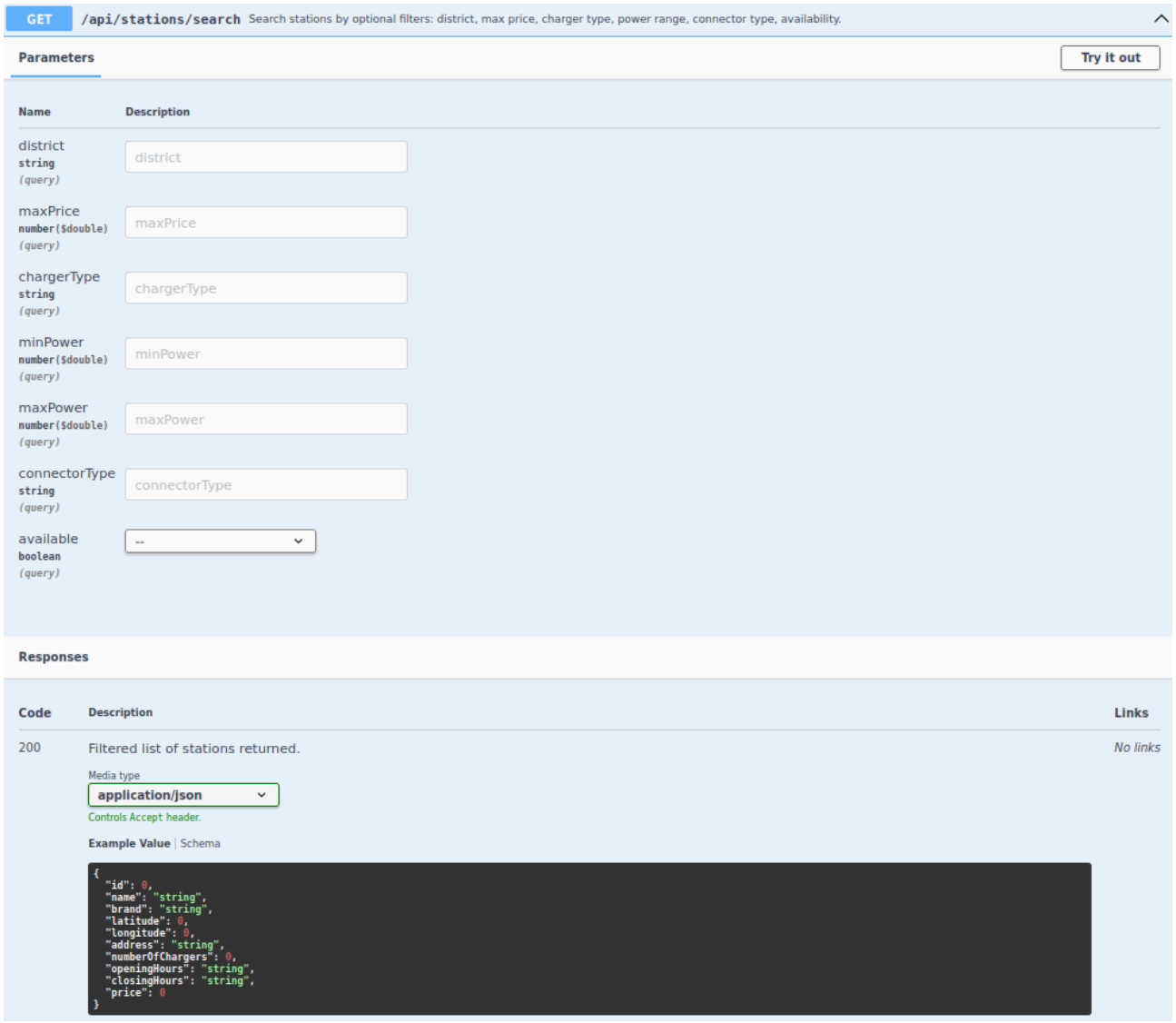
```
{
  "id": 0,
  "name": "string",
  "brand": "string",
  "latitude": 0,
  "longitude": 0,
  "address": "string",
  "numberOfChargers": 0,
  "openingHours": "string",
  "closingHours": "string",
  "price": 0
}
```

400	Bad Request Media type: */* **Example Value** | Schema	*No links*

```
string
```


POST
/api/stations
Create a new station.
Parameters
Try it out
No parameters
Request body
required
application/json
Example Value | Schema
{
"name": "string",
"brand": "string",
"latitude": 0,
"longitude": 0,
"address": "string",
"numberOfChargers": 0,
"openingHours": "string",
"closingHours": "string",
"price": 0
}
Responses
Code
Description
Links
200
Station created successfully.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"name": "string",
"brand": "string",
"latitude": 0,
"longitude": 0,
"address": "string",
"numberOfChargers": 0,
"openingHours": "string",
"closingHours": "string",
"price": 0
}
400
Invalid station data provided.
No links


GET
/api/stations/{id}/chargers
Get all chargers associated with a station.
Parameters
Try it out
Name
Description
Id
required
integer($int64)
(path)
id
Responses
Code
Description
Links
200
List of chargers returned.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"station": {
"id": 0,
"name": "string",
"brand": "string",
"latitude": 0,
"longitude": 0,
"address": "string",
"numberOfChargers": 0,
"openingHours": "string",
"closingHours": "string",
"price": 0
},
"type": "string",
"power": 0,
"available": true,
"connectorType": "string"
}


GET
/api/stations/search
Search stations by optional filters: district, max price, charger type, power range, connector type, availability.
Parameters
Try it out
Name
Description
district
string
(query)
district
maxPrice
number($double)
(query)
maxPrice
chargerType
string
(query)
chargerType
minPower
number($double)
(query)
minPower
maxPower
number($double)
(query)
maxPower
connectorType
string
(query)
connectorType
available
boolean
(query)
--
Responses
Code
Description
Links
200
Filtered list of stations returned.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"name": "string",
"brand": "string",
"latitude": 0,
"longitude": 0,
"address": "string",
"numberOfChargers": 0,
"openingHours": "string",
"closingHours": "string",
"price": 0
}

charger-controller

GET `/api/charger/{id}` Retrieve charger details by its ID.

Parameters | Try it out

Name	Description
id * required integer($int64) *(path)*	id

Responses

Code	Description	Links
200	Charger found and returned successfully. Media type: application/json Controls Accept header. **Example Value** | Schema	*No links*

```
{
  "id": 0,
  "station": {
    "id": 0,
    "name": "string",
    "brand": "string",
    "latitude": 0,
    "longitude": 0,
    "address": "string",
    "numberOfChargers": 0,
    "openingHours": "string",
    "closingHours": "string",
    "price": 0
  },
  "type": "string",
  "power": 0,
  "available": true,
  "connectorType": "string"
}
```

PUT `/api/charger/{id}` Update an existing charger.

Parameters | Try it out

Name	Description
id * required integer($int64) *(path)*	id

Request body required — application/json

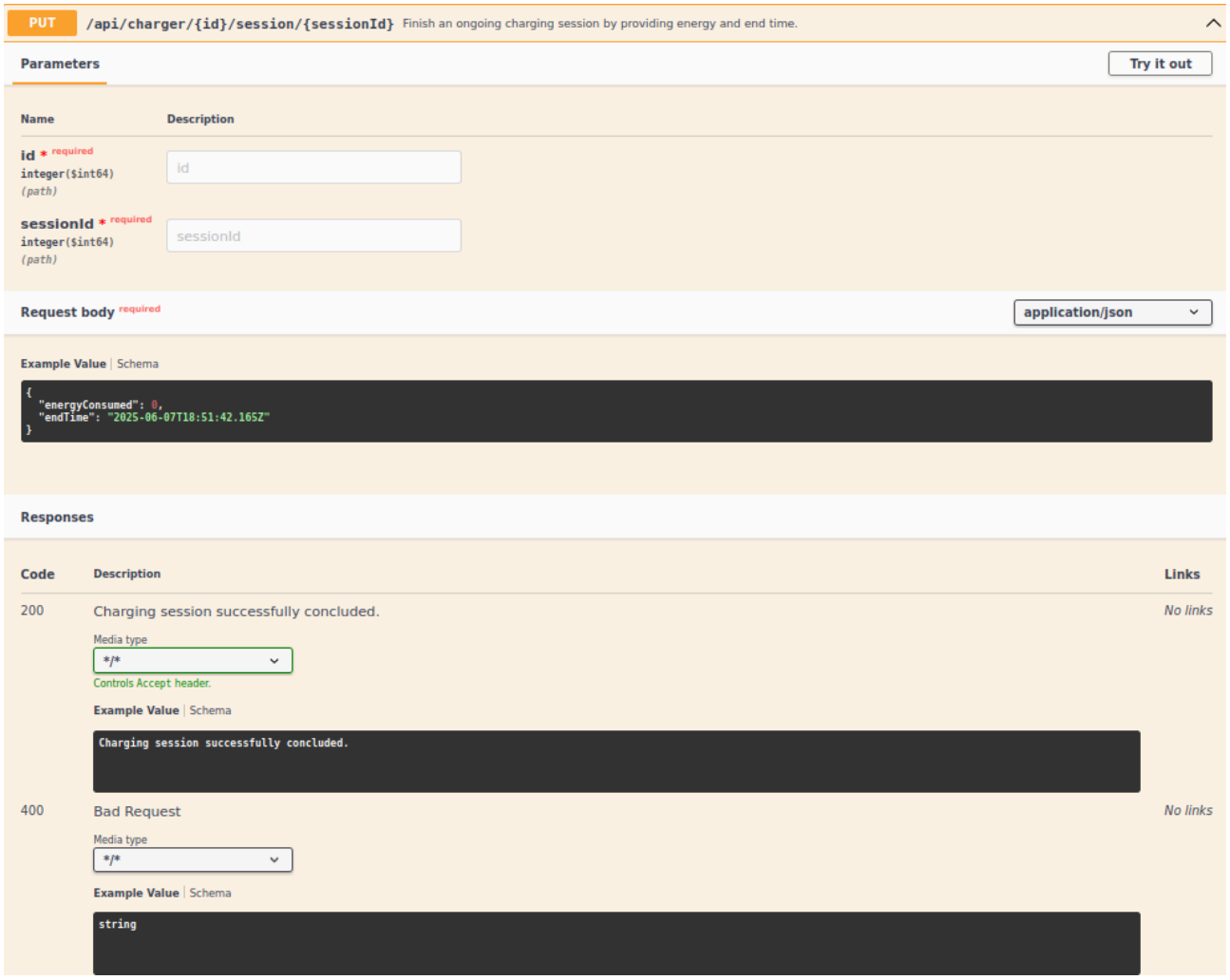
Example Value | Schema

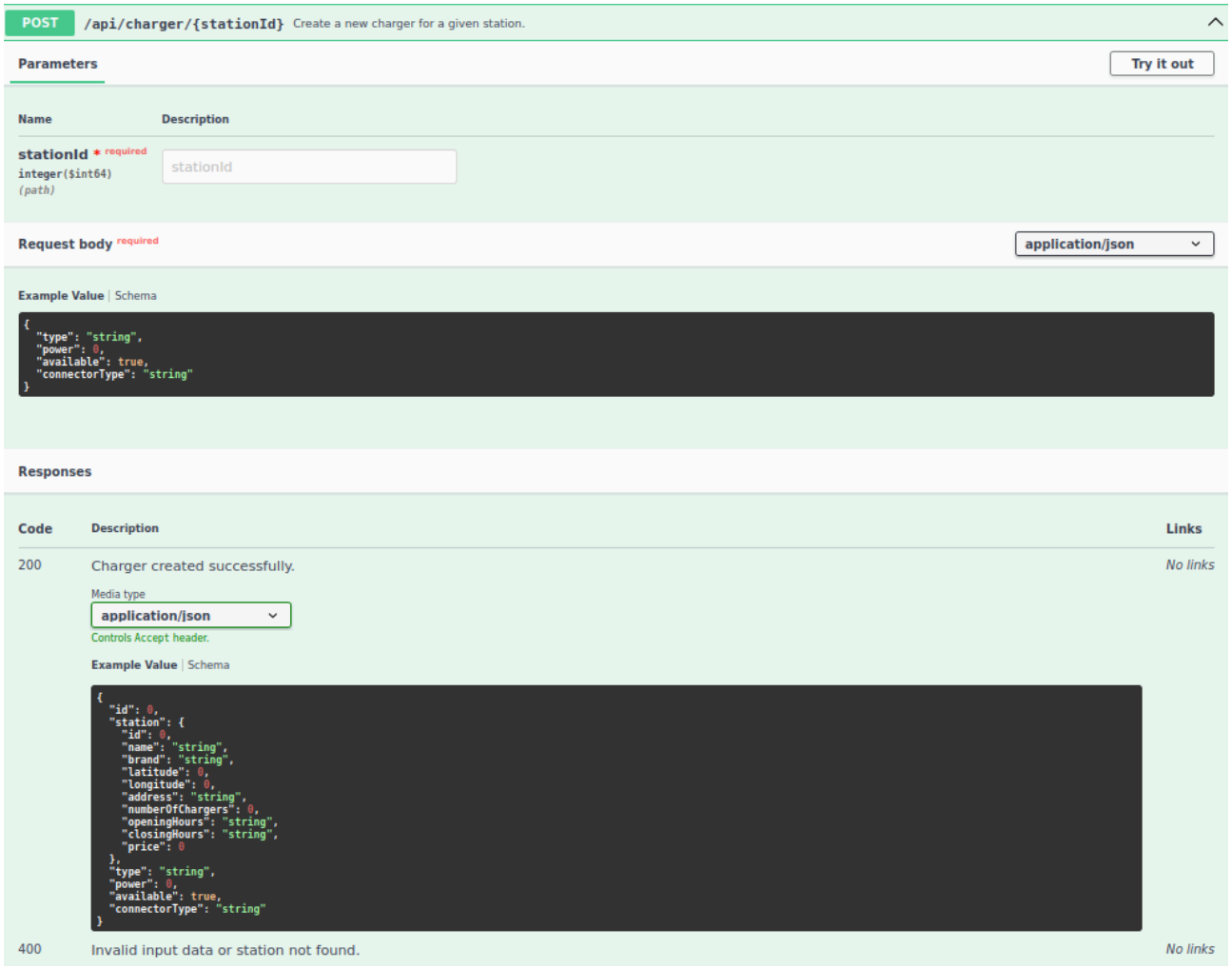
```
{
  "type": "string",
  "power": 0,
  "available": true,
  "connectorType": "string"
}
```

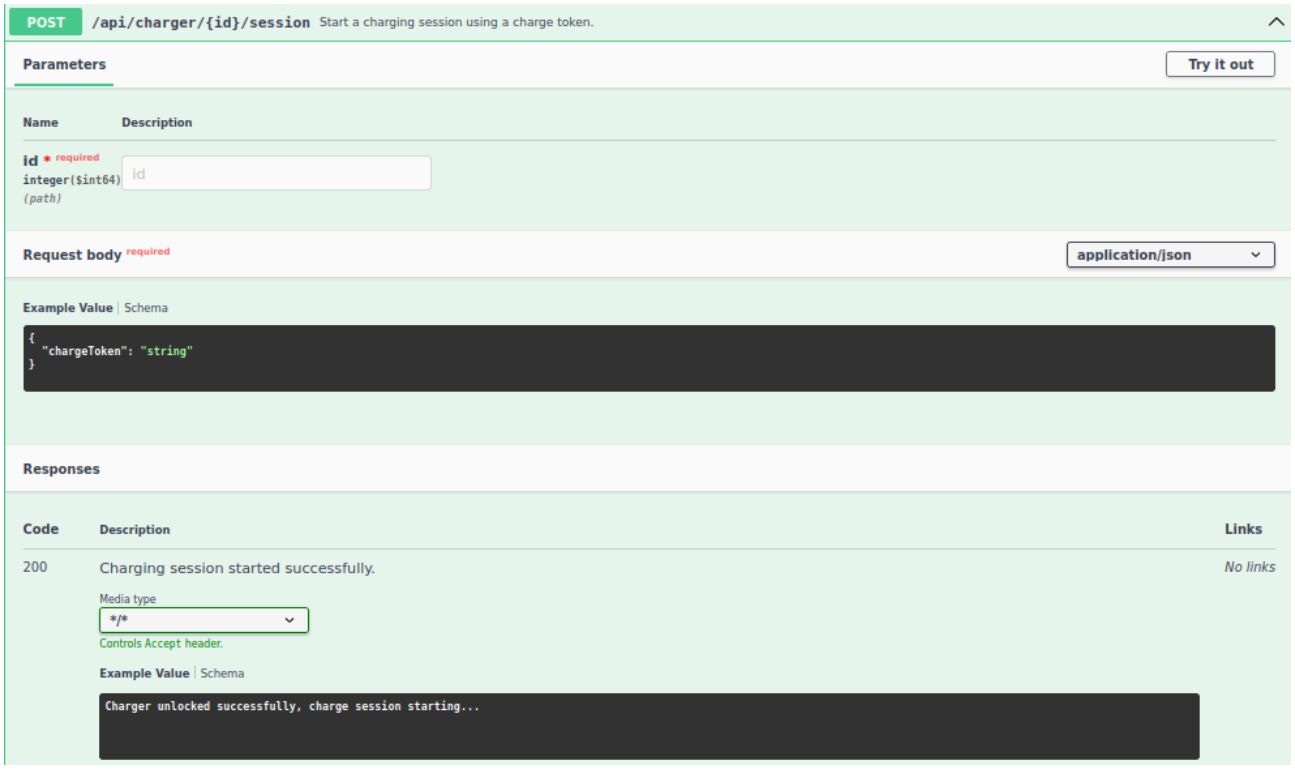
Responses

Code	Description	Links
200	Charger updated successfully. Media type: application/json Controls Accept header. **Example Value** | Schema	*No links*

```
{
  "id": 0,
  "station": {
    "id": 0,
    "name": "string",
    "brand": "string",
    "latitude": 0,
    "longitude": 0,
    "address": "string",
    "numberOfChargers": 0,
    "openingHours": "string",
    "closingHours": "string",
    "price": 0
  },
  "type": "string",
  "power": 0,
  "available": true,
  "connectorType": "string"
}
```


PUT
/api/charger/{id}/session/{sessionId}
Finish an ongoing charging session by providing energy and end time.
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
id
sessionId * required
integer($int64)
(path)
sessionId
Request body required
application/json
Example Value | Schema
{
"energyConsumed": 0,
"endTime": "2025-06-07T18:51:42.165Z"
}
Responses
Code
Description
Links
200
Charging session successfully concluded.
No links
Media type
/
Controls Accept header.
Example Value | Schema
Charging session successfully concluded.
400
Bad Request
No links
Media type
/
Example Value | Schema
string


POST /api/charger/{stationId} Create a new charger for a given station.
Parameters
Try it out
Name
Description
stationId * required
integer($int64)
(path)
stationId
Request body required
application/json
Example Value | Schema
{
"type": "string",
"power": 0,
"available": true,
"connectorType": "string"
}
Responses
Code
Description
Links
200
Charger created successfully.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"station": {
"id": 0,
"name": "string",
"brand": "string",
"latitude": 0,
"longitude": 0,
"address": "string",
"numberOfChargers": 0,
"openingHours": "string",
"closingHours": "string",
"price": 0
},
"type": "string",
"power": 0,
"available": true,
"connectorType": "string"
}
400
Invalid input data or station not found.
No links


POST /api/charger/{id}/session Start a charging session using a charge token.
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
id
Request body required
application/json
Example Value | Schema
{
"chargeToken": "string"
}
Responses
Code
Description
Links
200
Charging session started successfully.
No links
Media type
/
Controls Accept header.
Example Value | Schema
Charger unlocked successfully, charge session starting...

staff-controller

POST `/api/staff/operator` Create a new operator staff account.

Parameters | Try it out

No parameters

Request body required | application/json

Example Value | Schema

```
{
  "name": "string",
  "age": 120,
  "number": "950881874",
  "mail": "string",
  "password": "PU{y$t9~y+SGWg*:Yf?Ux@l!>L#V.-EwT%yq9e$",
  "address": "string"
}
```

Responses

Code	Description	Links
200	Operator account created successfully. Media type: */* Controls Accept header. **Example Value** | Schema `Operator account created successfully.`	*No links*

POST `/api/staff/operator/assign-station` Assign a station to an operator.

Parameters | Try it out

No parameters

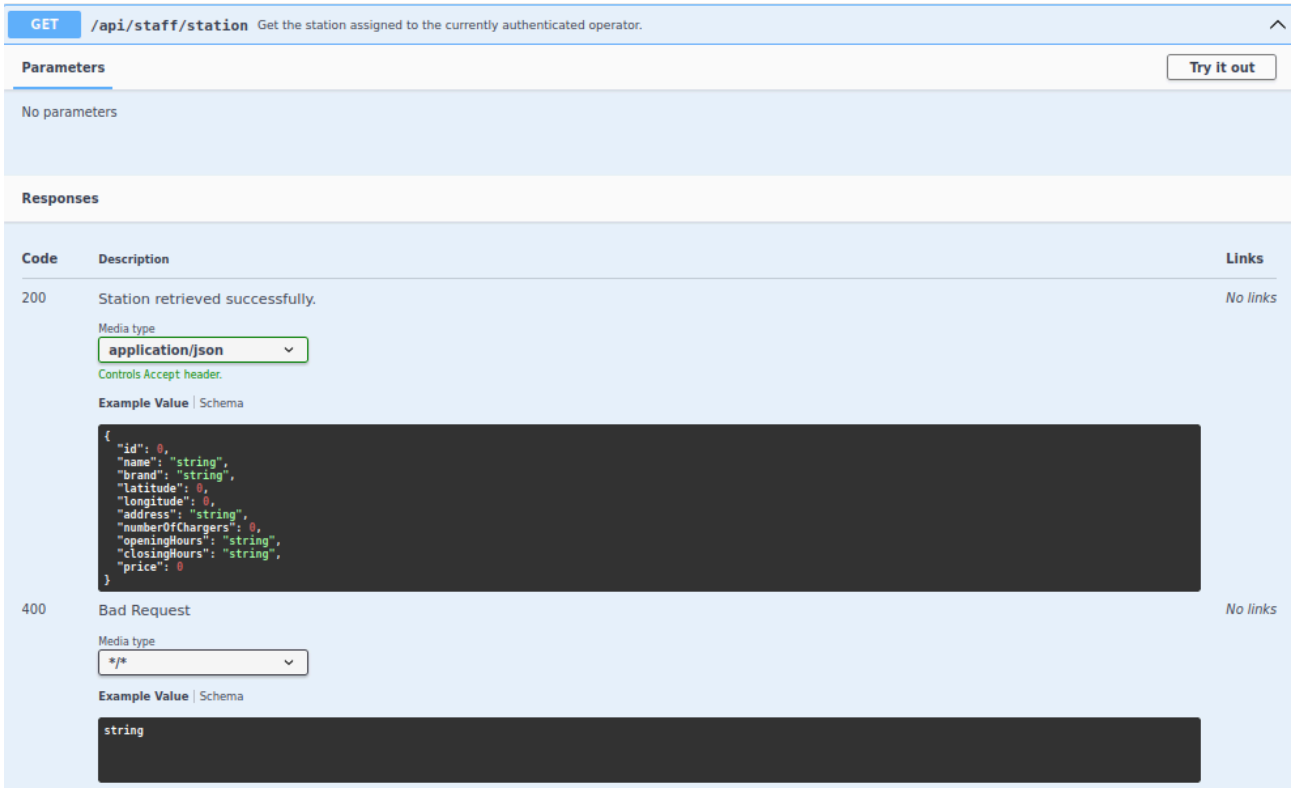
Request body required | application/json

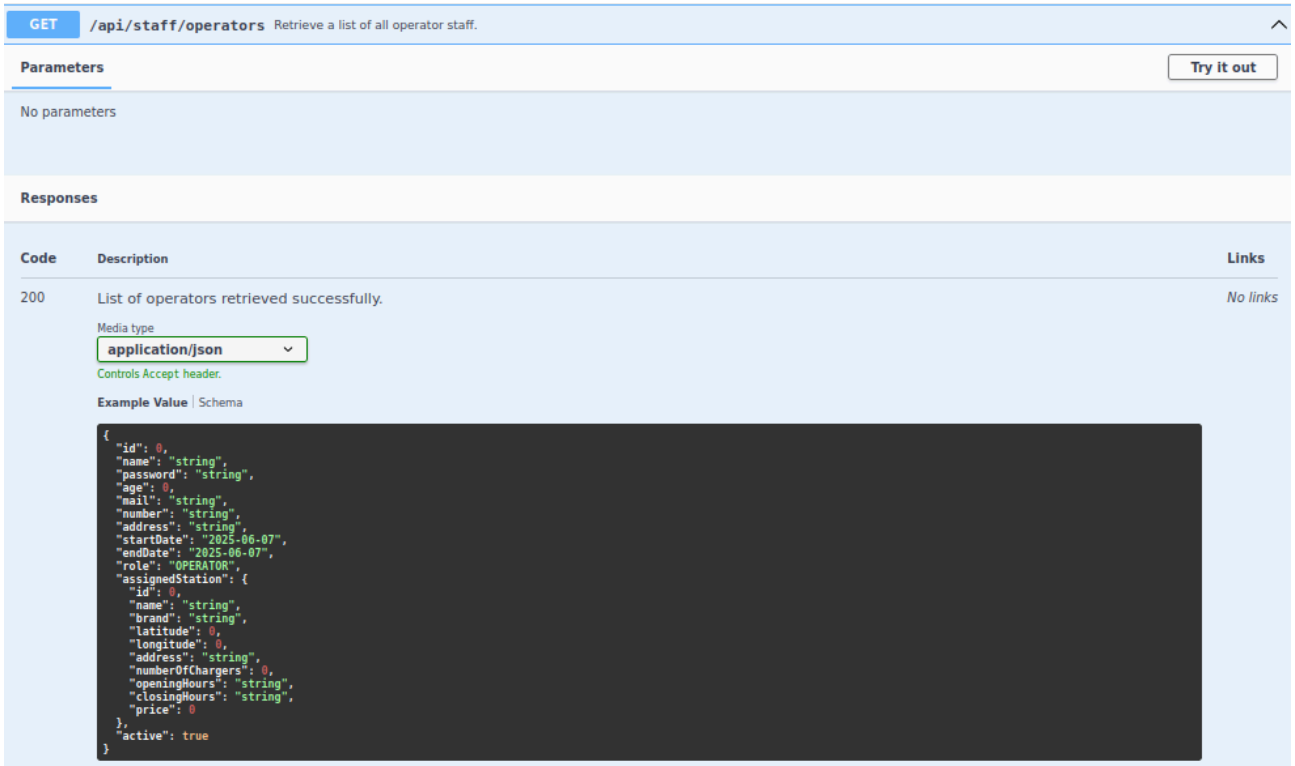
Example Value | Schema

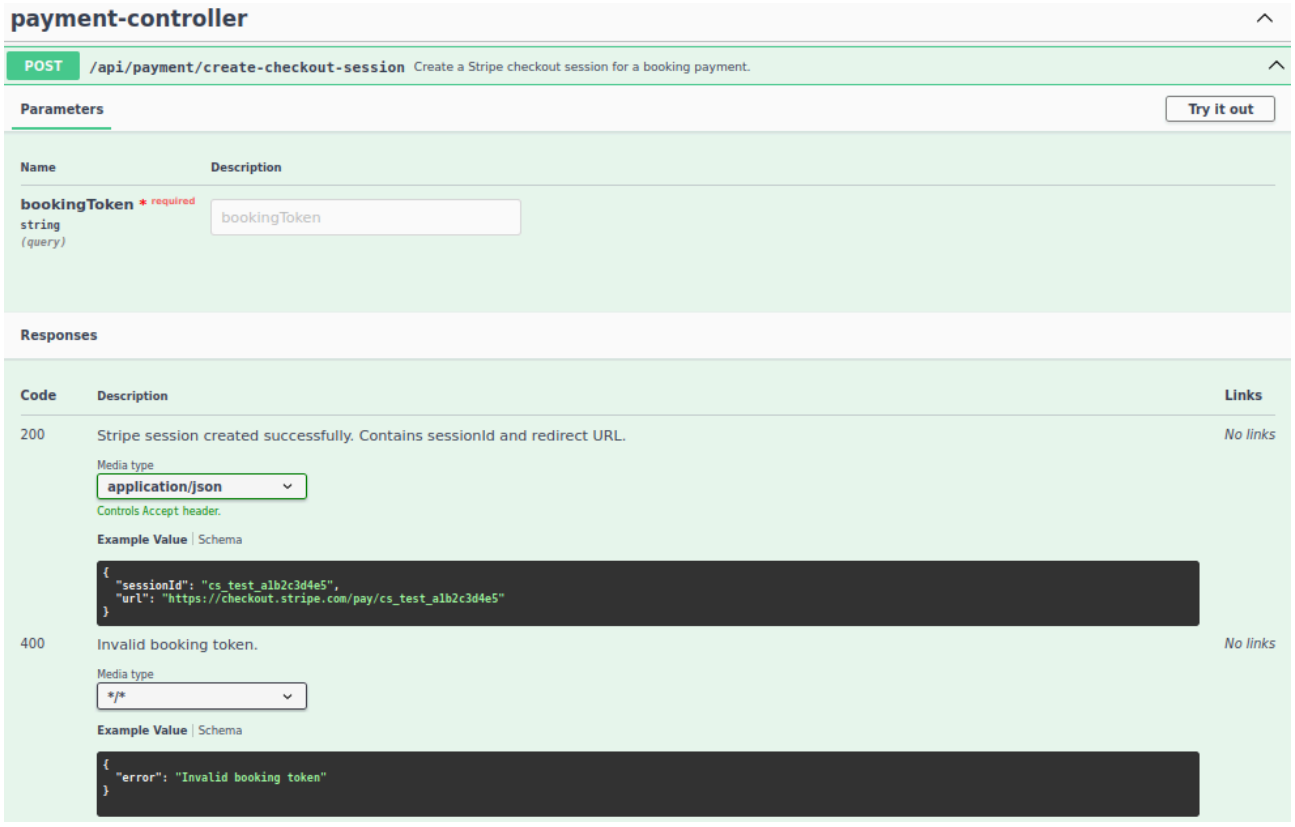
```
{
  "operatorId": 0,
  "stationId": 0
}
```

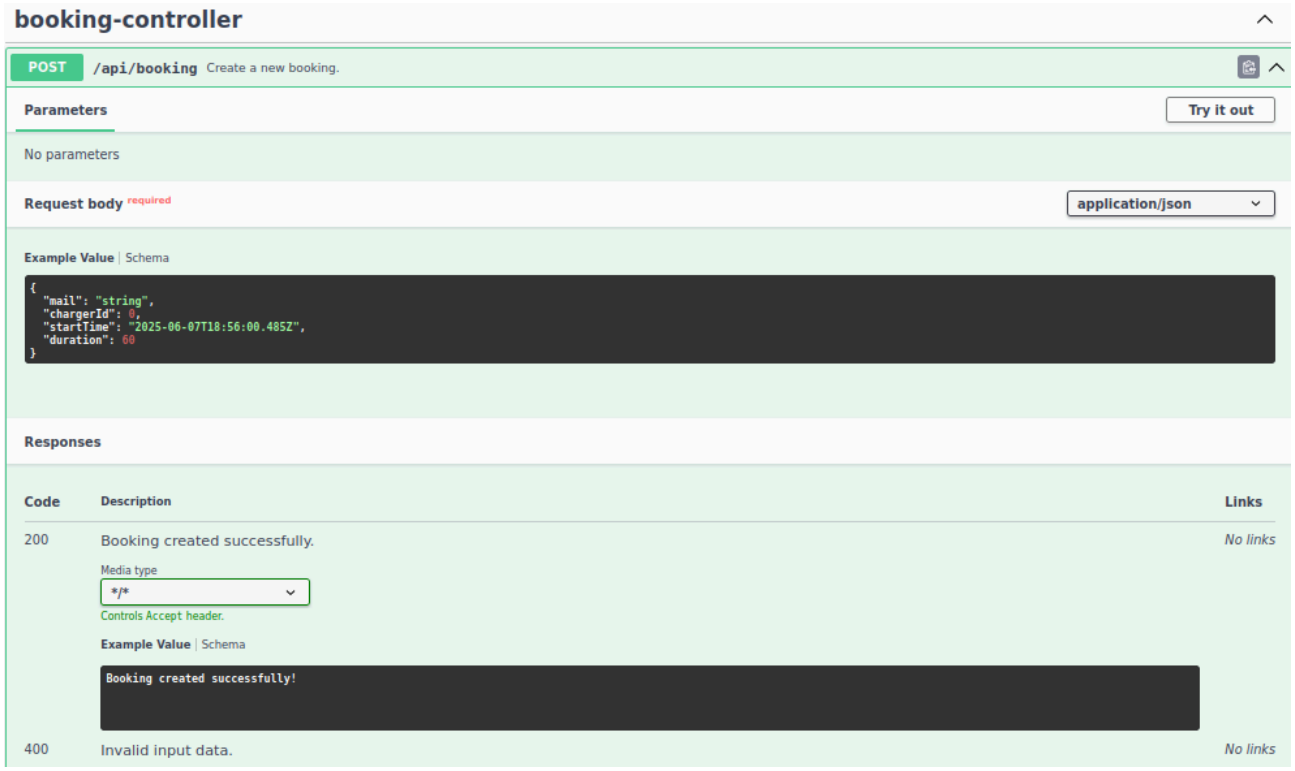
Responses

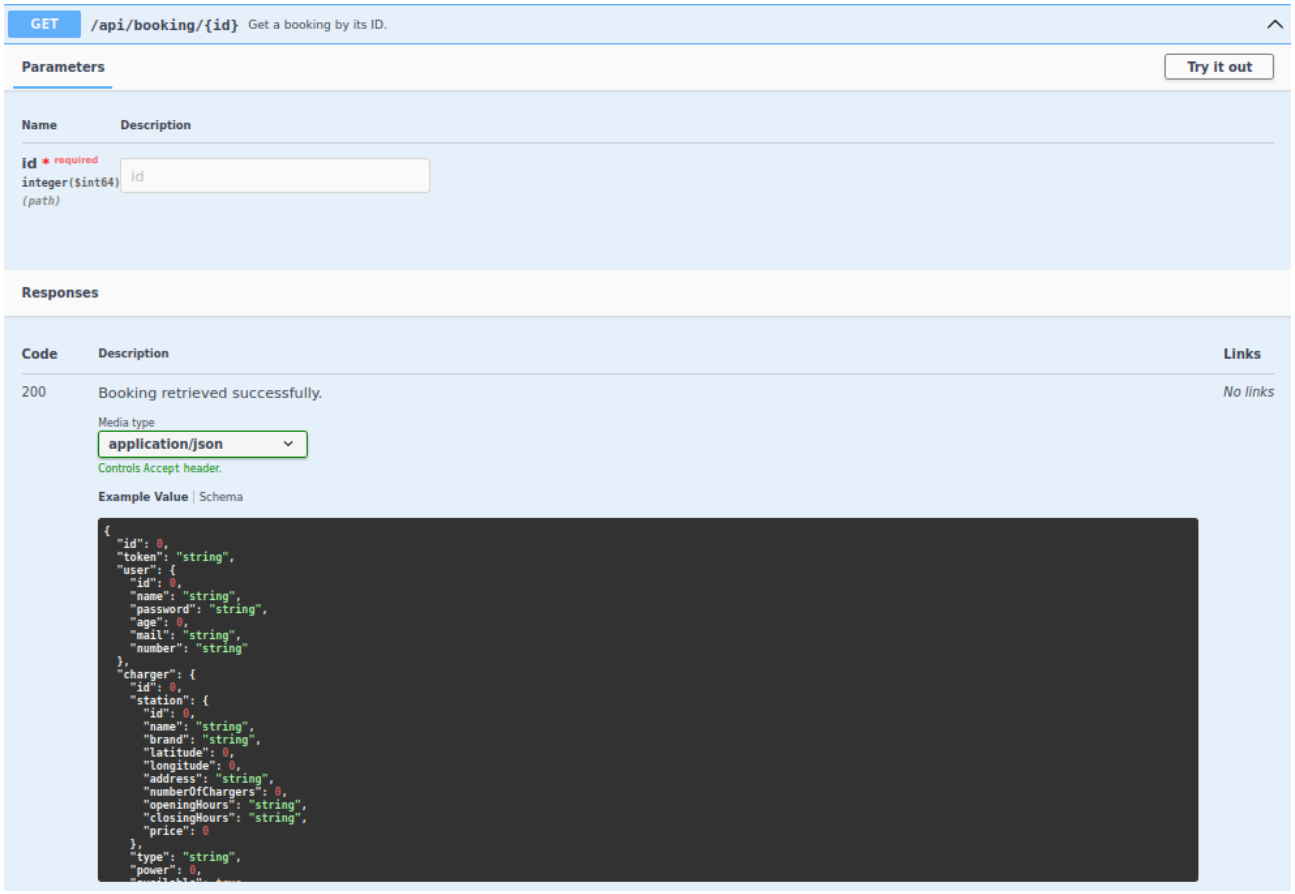
Code	Description	Links
200	Station assigned to operator successfully. Media type: */* Controls Accept header. **Example Value** | Schema `Station assigned to operator successfully.`	*No links*

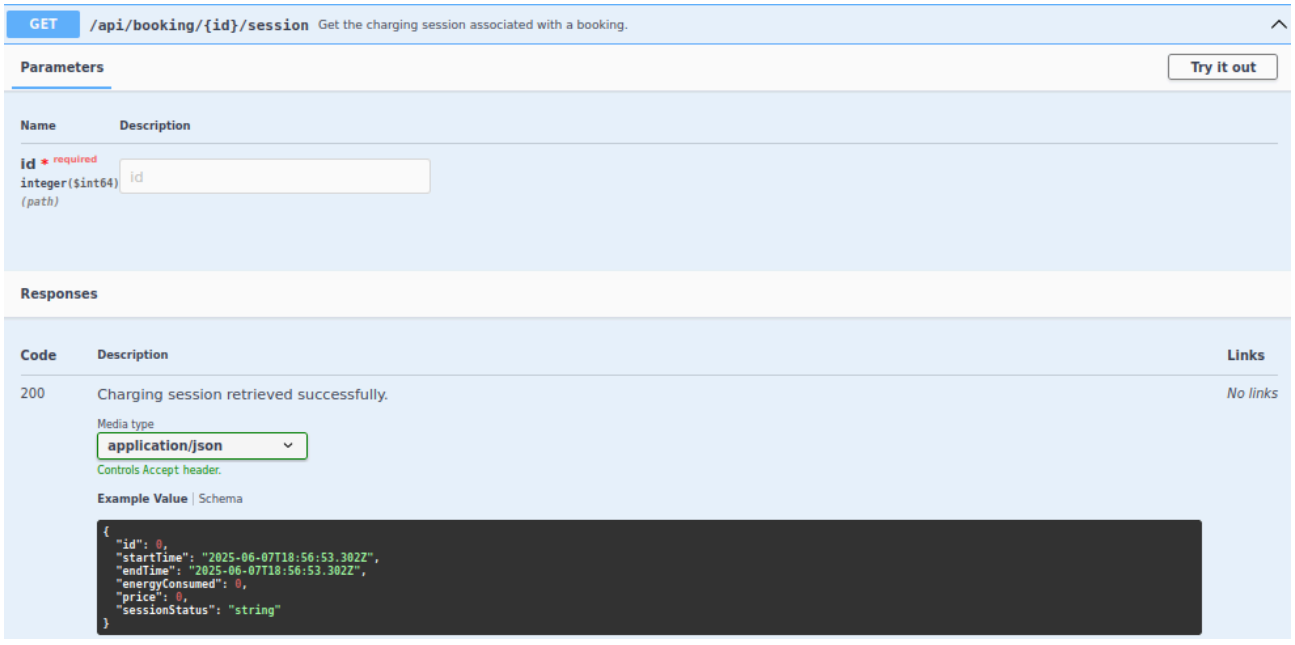

GET
/api/staff/station
Get the station assigned to the currently authenticated operator.
Parameters
Try it out
No parameters
Responses
Code
Description
Links
200
Station retrieved successfully.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"name": "string",
"brand": "string",
"latitude": 0,
"longitude": 0,
"address": "string",
"numberOfChargers": 0,
"openingHours": "string",
"closingHours": "string",
"price": 0
}
400
Bad Request
No links
Media type
/
Example Value | Schema
string

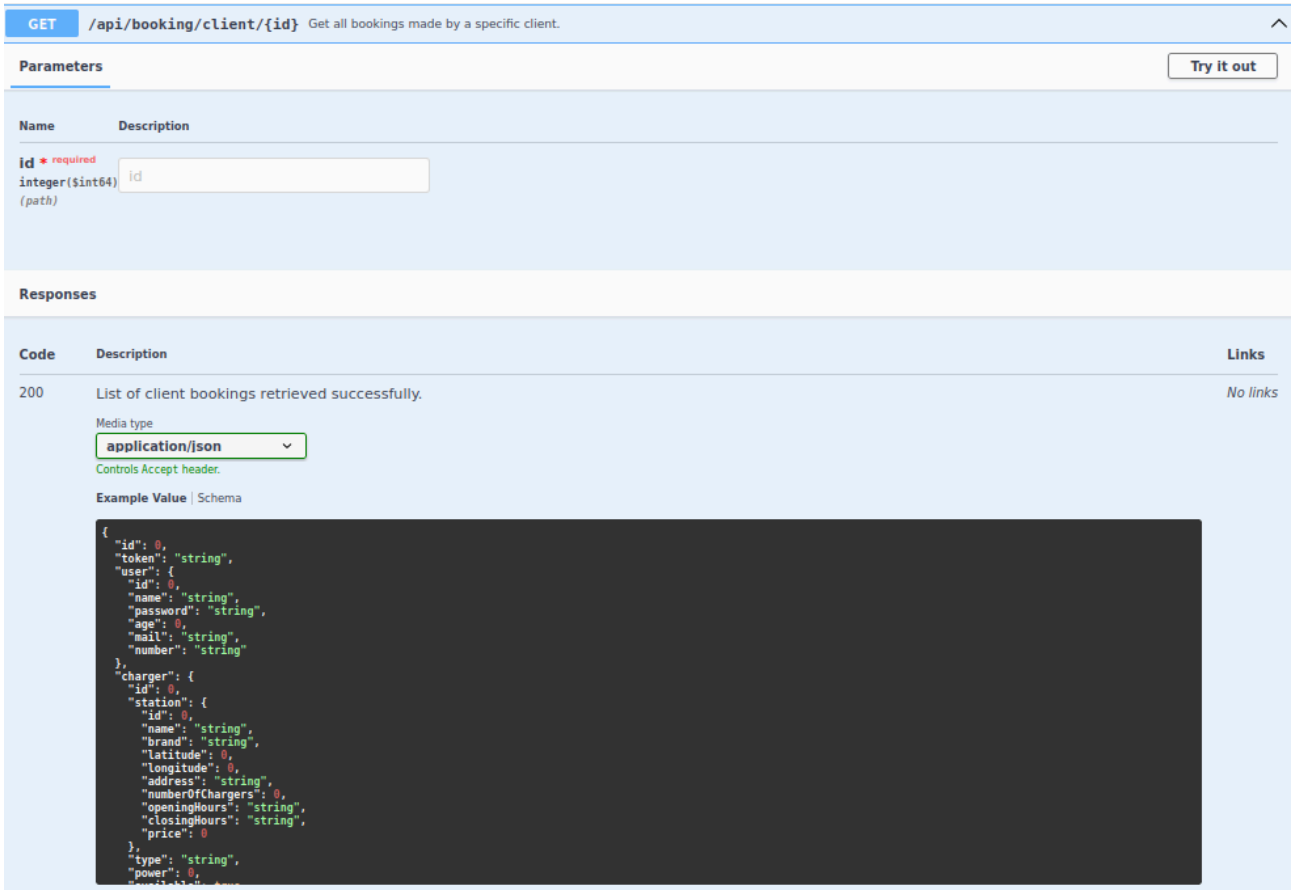

GET
/api/staff/operators
Retrieve a list of all operator staff.
Parameters
Try it out
No parameters
Responses
Code
Description
Links
200
List of operators retrieved successfully.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"name": "string",
"password": "string",
"age": 0,
"mail": "string",
"number": "string",
"address": "string",
"startDate": "2025-06-07",
"endDate": "2025-06-07",
"role": "OPERATOR",
"assignedStation": {
"id": 0,
"name": "string",
"brand": "string",
"latitude": 0,
"longitude": 0,
"address": "string",
"numberOfChargers": 0,
"openingHours": "string",
"closingHours": "string",
"price": 0
},
"active": true
}


payment-controller
POST /api/payment/create-checkout-session Create a Stripe checkout session for a booking payment.
Parameters
Try it out
Name
Description
bookingToken * required
string
(query)
bookingToken
Responses
Code
Description
Links
200
Stripe session created successfully. Contains sessionId and redirect URL.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"sessionId": "cs_test_a1b2c3d4e5",
"url": "https://checkout.stripe.com/pay/cs_test_a1b2c3d4e5"
}
400
Invalid booking token.
No links
Media type
/
Example Value | Schema
{
"error": "Invalid booking token"
}


booking-controller
POST /api/booking Create a new booking.
Parameters
Try it out
No parameters
Request body required
application/json
Example Value | Schema
{
"mail": "string",
"chargerId": 0,
"startTime": "2025-06-07T18:56:00.485Z",
"duration": 60
}
Responses
Code
Description
Links
200
Booking created successfully.
No links
Media type
/
Controls Accept header.
Example Value | Schema
Booking created successfully!
400
Invalid input data.
No links


GET /api/booking/{id} Get a booking by its ID.
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
Links
200
Booking retrieved successfully.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"token": "string",
"user": {
"id": 0,
"name": "string",
"password": "string",
"age": 0,
"mail": "string",
"number": "string"
},
"charger": {
"id": 0,
"station": {
"id": 0,
"name": "string",
"brand": "string",
"latitude": 0,
"longitude": 0,
"address": "string",
"numberOfChargers": 0,
"openingHours": "string",
"closingHours": "string",
"price": 0
},
"type": "string",
"power": 0,


GET /api/booking/{id}/session Get the charging session associated with a booking.
Parameters
Try it out
Name
Description
id * required
integer($int64)
(path)
id
Responses
Code
Description
Links
200
Charging session retrieved successfully.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"startTime": "2025-06-07T18:56:53.302Z",
"endTime": "2025-06-07T18:56:53.302Z",
"energyConsumed": 0,
"price": 0,
"sessionStatus": "string"
}



GET /api/booking/client/{id} Get all bookings made by a specific client.

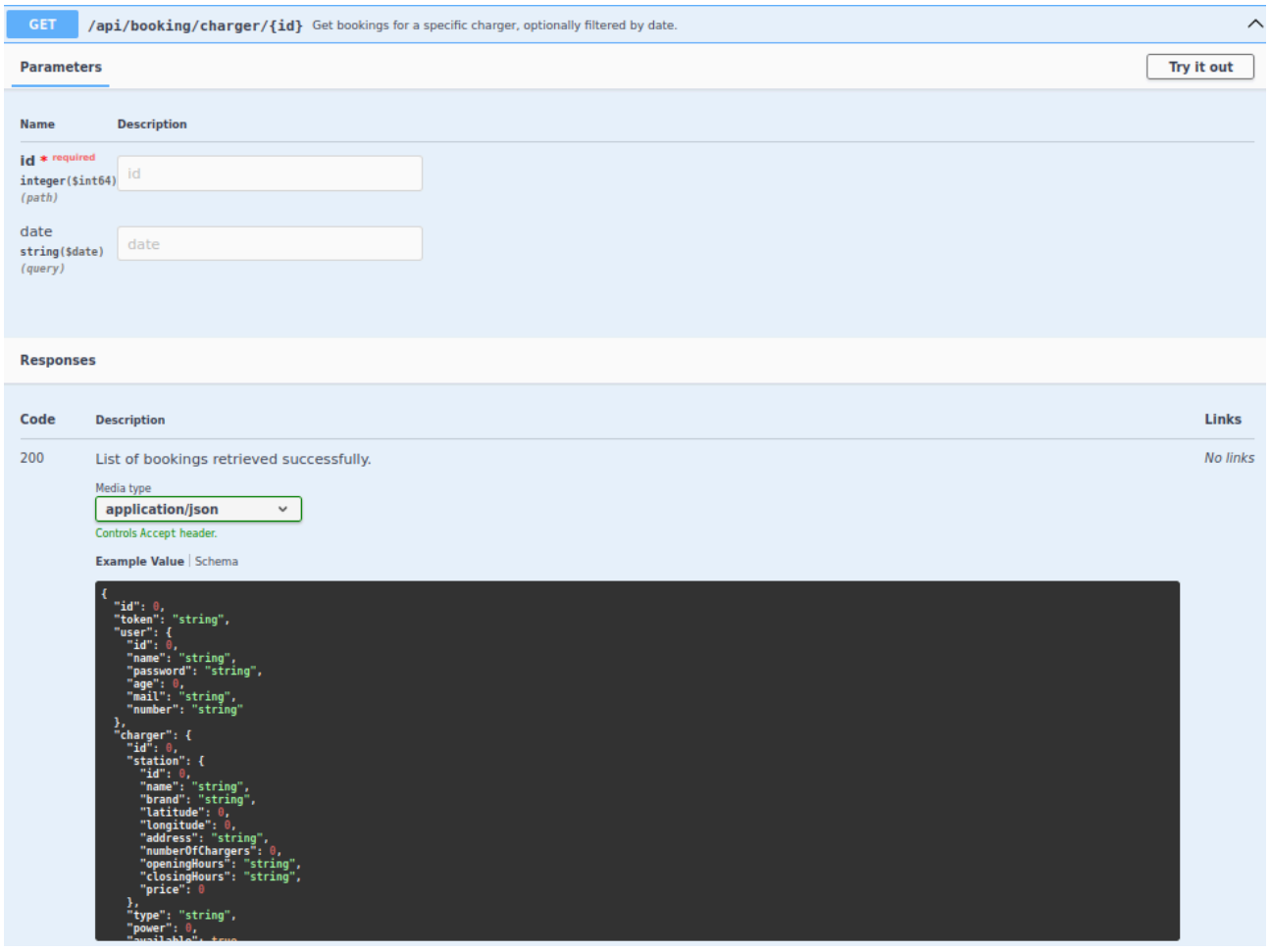
Parameters Try it out

Name	Description
id * required integer($int64) (path)	id

Responses

Code	Description	Links
200	List of client bookings retrieved successfully. Media type: application/json Controls Accept header. Example Value | Schema	No links


```
{
  "id": 0,
  "token": "string",
  "user": {
    "id": 0,
    "name": "string",
    "password": "string",
    "age": 0,
    "mail": "string",
    "number": "string"
  },
  "charger": {
    "id": 0,
    "station": {
      "id": 0,
      "name": "string",
      "brand": "string",
      "latitude": 0,
      "longitude": 0,
      "address": "string",
      "numberOfChargers": 0,
      "openingHours": "string",
      "closingHours": "string",
      "price": 0
    },
    "type": "string",
    "power": 0,
```

GET /api/booking/charger/{id} Get bookings for a specific charger, optionally filtered by date.

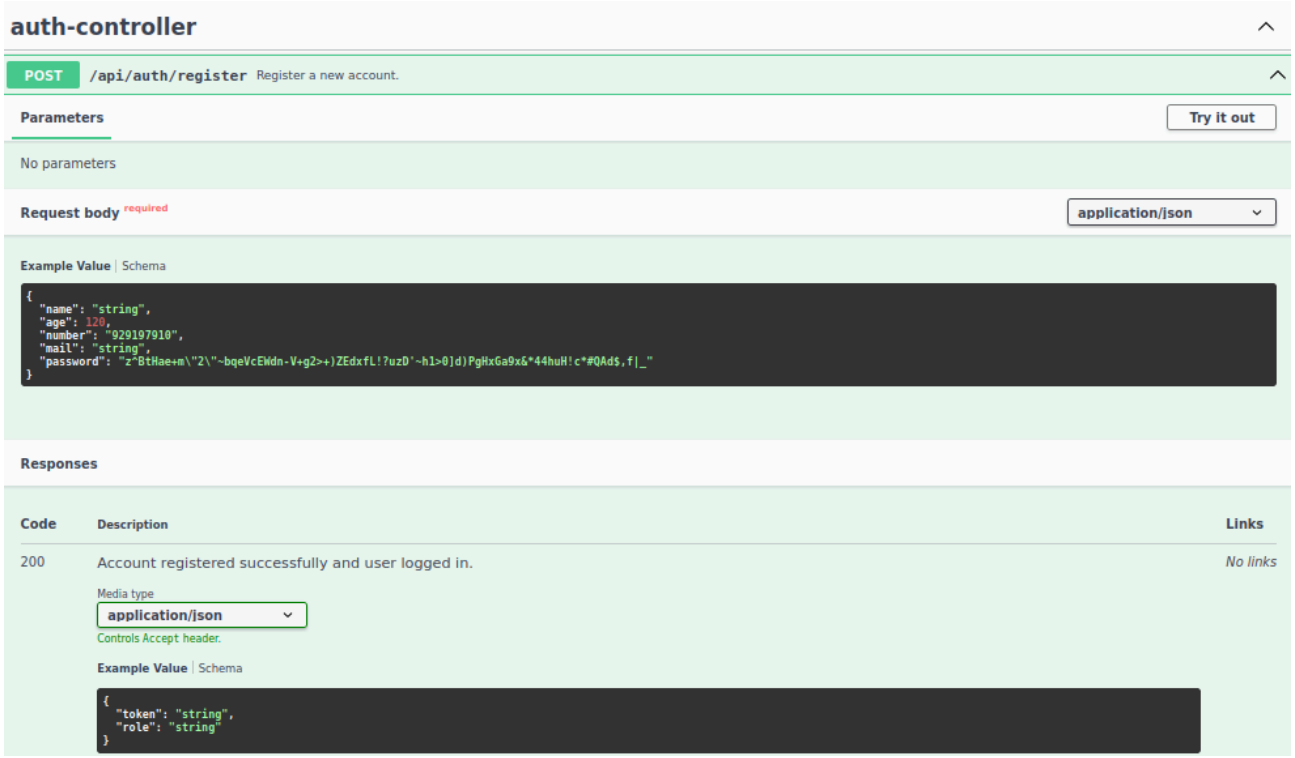
Parameters Try it out

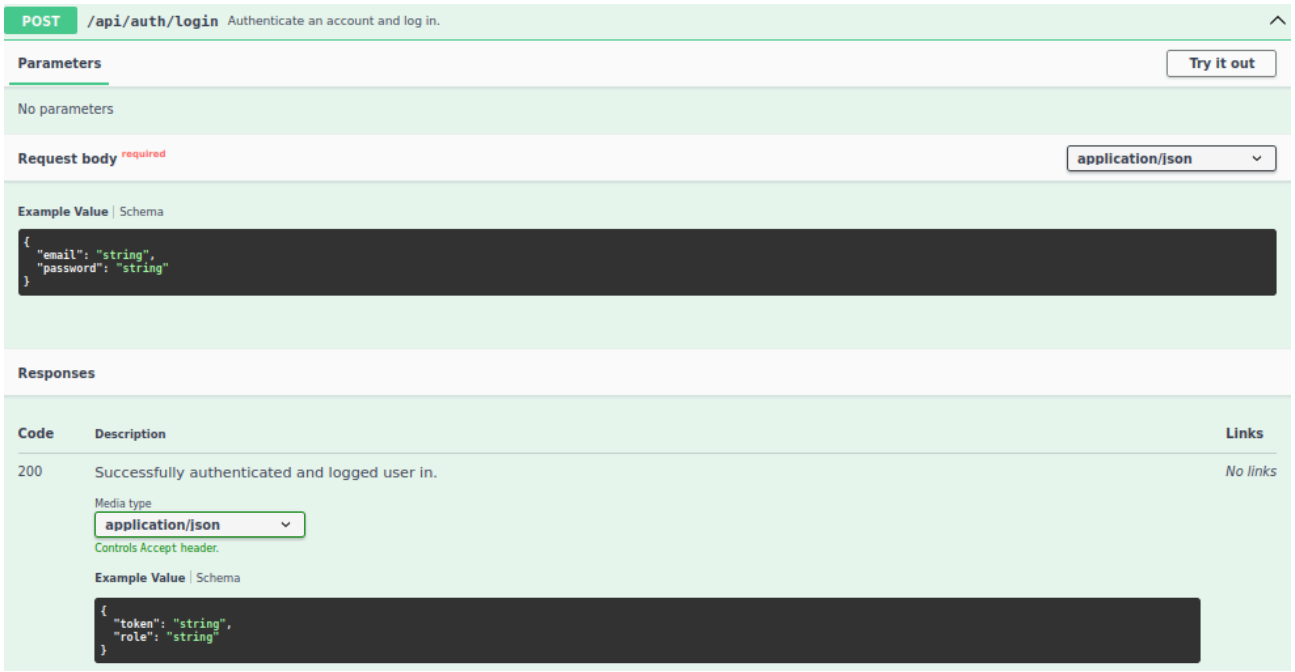
Name	Description
id * required integer($int64) (path)	id
date string($date) (query)	date

Responses

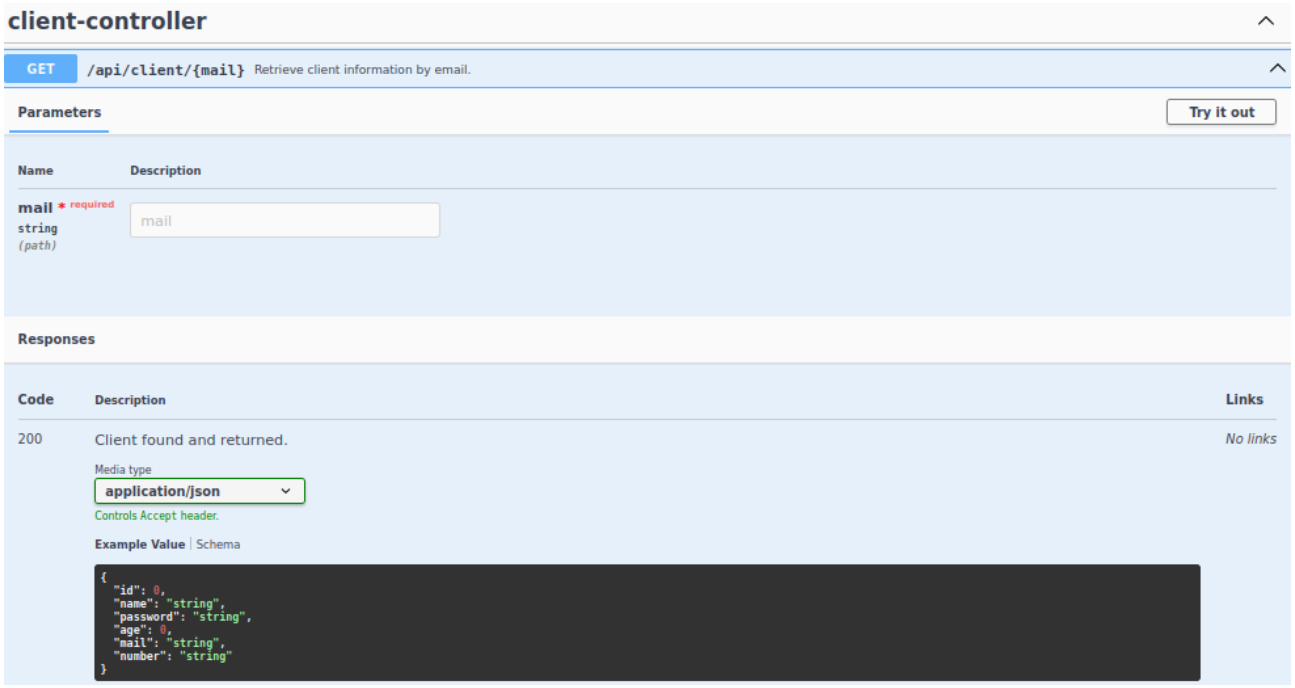
Code	Description	Links
200	List of bookings retrieved successfully. Media type: application/json Controls Accept header. Example Value | Schema	No links


```
{
  "id": 0,
  "token": "string",
  "user": {
    "id": 0,
    "name": "string",
    "password": "string",
    "age": 0,
    "mail": "string",
    "number": "string"
  },
  "charger": {
    "id": 0,
    "station": {
      "id": 0,
      "name": "string",
      "brand": "string",
      "latitude": 0,
      "longitude": 0,
      "address": "string",
      "numberOfChargers": 0,
      "openingHours": "string",
      "closingHours": "string",
      "price": 0
    },
    "type": "string",
    "power": 0,
```


auth-controller
POST /api/auth/register Register a new account.
Parameters
Try it out
No parameters
Request body required
application/json
Example Value | Schema
{
"name": "string",
"age": 120,
"number": "929197910",
"mail": "string",
"password": "z^BtHae+m\"2\"~bqeVcEWdn-V+g2>+)ZEdxfL!?uzD'~h1>0]d)PgHxGa9x&*44huH!c*#QAd$,f|_"
}
Responses
Code
Description
Links
200
Account registered successfully and user logged in.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"token": "string",
"role": "string"
}


POST /api/auth/login Authenticate an account and log in.
Parameters
Try it out
No parameters
Request body required
application/json
Example Value | Schema
{
"email": "string",
"password": "string"
}
Responses
Code
Description
Links
200
Successfully authenticated and logged user in.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"token": "string",
"role": "string"
}


GET /api/auth/validate Validate JWT token and return the user's role.
Parameters
Try it out
No parameters
Responses
Code
Description
Links
200
Token is valid.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"role": "USER"
}


client-controller
GET /api/client/{mail} Retrieve client information by email.
Parameters
Try it out
Name
Description
mail * required
string
(path)
mail
Responses
Code
Description
Links
200
Client found and returned.
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"id": 0,
"name": "string",
"password": "string",
"age": 0,
"mail": "string",
"number": "string"
}

6 References and resources

Project resources

Resource:	**URL/location:**
Git repository	https://github.com/TQS-Org/TQS-Project
QA dashboard	https://sonarcloud.io/project/configuration?id=TQS-Org_TQS-Project
CI/CD pipeline	https://github.com/TQS-Org/TQS-Project/tree/main/.github/workflows
Deployment ready to use	http://deti-tqs-23.ua.pt:3000/