TQS: Product specification report

Francisco Albergaria [114646], Regina Tavares []
v2025-05-22

[This report is the main source of technical documentation on the project, clarifying the functional scope and architectural choices. Provide concise, but informative content, **allowing other software engineers to understand the product**.
Tips on the expected content placed along the document are meant to be removed!
You may use English or Portuguese; do not mix.]

1 Introduction

1.1 Overview of the project

<contextualize the objectives of this project assignment in the scope of the TQS course>
<introduce your application/product: brief overview of the solution concept. What is it good for? Introduce the name of the product if it has one>

1.2 Known limitations

<explain the known limitations, especially the features that were planned/expected but not implemented (and why...)

To be reviewed and completed by the end of the project >

1.3 References and resources

<document the key components (e.g.: libraries, web services) or key references (e.g.: blog post) used that were really helpful and certainly would help other students pursuing a similar work>

2 Product concept and requirements

2.1 Vision statement

<functional (black-box) description of the application: Which is the high-level/business problem being solved by your system? Which are the key features you promise to address it?>
<if needed, clarify what was planned/expected to be included but was changed to a different approach/concept >
<optional: how is your system different or similar to other well-known products?>
<optional: additional details on the process for the requirements gathering and selection (how did we developed the concept? Who helped us with the requirements? etc)>

Our application addresses a critical challenge in the growing electric vehicle (EV) ecosystem: the fragmentation of EV charging services across multiple providers. As EV adoption accelerates, drivers often face difficulty locating available chargers, managing bookings, and completing payments due to the lack of interoperability between different networks. In parallel, station operator managers lack unified tools to monitor, manage, and optimize their infrastructure effectively.

Our system aims to solve both challenges by offering a centralized digital platform that enables seamless access for EV drivers to multiple charging stations and services in a unified interface, while also providing station operator managers with a dedicated management dashboard for operational control, pricing configuration, and usage analytics.

Key features of our solution include:

a) Station Discovery: Drivers can locate charging stations in real time, filtering by type and availability.
b) Slot Booking & Scheduling: Drivers can book specific time slots at chosen stations, preventing overbooking and ensuring availability.
c) Payment Integration: Secure pay-per-use payment processing through integration with external payment gateways.
d) User and Operator Profiles: EV drivers can manage their profiles and view past charging sessions, while station operator managers can manage their infrastructure, monitor usage, and apply pricing rules.
e) Analytics: Both user types have access to relevant analytics—drivers can view their charging history, and operators can analyze station performance and usage metrics.
f) Favorites: Drivers can mark stations as favorites for quick access.

Initially, we considered a minimal concept limited to booking slots at selected stations. However, we evolved the concept to provide a scalable and dual-purpose platform, addressing both the end-

user experience for drivers and the operational needs of station operator managers with a broader range of features.

Compared to existing solutions, our system stands out through its focus on interoperability, modular architecture, and dual-user support, enabling the integration of third-party payment methods.

The requirements and concept were developed by our team based on the problem scenario proposed in the course assignment. Internal team discussions, iterative brainstorming, and review of real-world pain points helped define the main functionalities. The final set of features was selected based on user value, technical feasibility, and alignment with the project's scope and of course, timeline.

Functional Requirements (FR):

User Management

g) FR1.1: The system shall allow EV driver registration.
h) FR1.2: The system shall allow station operator registration.
i) FR1.3: The system shall allow user authentication.
j) FR1.4: The system shall allow user profile management.

Station Management

k) FR2.1: The system shall allow charging station registration.
l) FR2.2: The system shall allow real-time station status visualization (available, occupied, maintenance).
m) FR2.3: The system shall allow station maintenance management.
n) FR2.4: The system shall allow price configuration per station.

Reservations

o) FR3.1: The system shall allow charging slot reservation.
p) FR3.2: The system shall allow viewing available slots.
q) FR3.3: The system shall allow reservation cancellation.
r) FR3.4: The system shall allow viewing charging slot reservations.

Payments

s) FR4.1: The system shall allow payment processing via external payment gateways.
t) FR4.2: The system shall allow payment history visualization.

Analytics

u) FR5.1: The system shall allow EV drivers to view their charging history.
v) FR5.2: The system shall allow station operators to view usage metrics and KPIs.

Favorites

w) FR6.1: The system shall allow EV drivers to mark stations as favorites.

x) FR6.2: The system shall allow EV drivers to view and manage their list of favorite stations.

Non-Functional Requirements (NFR):

Performance

y) NFR1.1: The system shall respond to requests in less than 2 seconds.
z) NFR1.2: The system shall support at least 500 concurrent users.

Security

aa) NFR2.1: The system shall implement secure authentication.
bb) NFR2.2: The system shall protect user sensitive data.
cc) NFR2.3: The system shall implement role-based access control.

Availability

dd) NFR3.1: The system shall be available 99% of the time.
ee) NFR3.2: The system shall implement automatic data backup.

Scalability

ff) NFR4.1: The system shall be capable of horizontal scaling.
gg) NFR4.2: The system shall support an increasing number of stations and users.

Usability

hh) NFR5.1: The interface shall be intuitive and responsive.
ii) NFR5.2: The system shall be accessible via web browser (desktop and mobile).

Integration

jj) NFR6.1: The system shall integrate with external payment gateways.

2.2 Personas and scenarios

<"Personas are fictional people. They have names, likenesses, clothes, occupations, families, friends, pets, possessions, and so forth. They have age, gender, ethnicity, educational achievement, and socioeconomic status. They have life stories, goals and tasks. Scenarios can be constructed around personas, but the personas come first. They are not 'agents' or 'actors' in a script, they are people. Photographs of the personas and their workplaces are created and displayed. [...] It is to obtain a more powerful level of identification and engagement that enable design, development, and testing to move forward more effectively". Adapted from Grudin, J. and Pruitt, J., 2002, June. Personas, participatory design and product development: An infrastructure for engagement. In Proc. PDC (Vol. 2).

Sample personas: secção 4.1, neste artigo (open access)] >

Persona 1: Sofia Antunes

Name: Sofia Antunes
Age: 34
Occupation: Marketing Manager at a tech startup
Location: Lisbon, Portugal
Education: Master's degree in Communication
Family: Married, 1 child
Lifestyle: Environmentally conscious, tech-savvy, often works remotely
Vehicle: Hyundai Ioniq 5

Background:
Sofia is a dynamic professional who juggles a busy work schedule with her family life. Working in a fast-paced tech environment, she often travels for client meetings across urban and suburban areas. As an early adopter of electric mobility, she values sustainability and efficiency in her daily routines.

Frustrations:
Sofia frequently travels across cities for client meetings and often finds it difficult to locate a nearby charging station that is available when she needs it. She has faced situations where stations were already occupied despite being shown as free on the app. She also dislikes having to use multiple apps from different providers, each with its own interface and payment method.

Motivations and Goals:
Sofia wants a reliable and unified platform that allows her to quickly find and reserve available charging stations, especially during her tight schedules. She expects a smooth charging experience with transparent pricing and a history of her past usage. A single app that combines map navigation, booking, and payment would simplify her EV experience considerably.

Scenario – Business trip

Sofia Antunes is preparing for a client meeting in Porto and needs to ensure her Hyundai Ioniq 5 is fully charged before departure. She opens the EV charging app on her phone during breakfast and uses the real-time station discovery feature to find available charging points near her home in Lisbon. She filters for fast chargers within a 5 km radius and selects a charging station that offers off-peak pricing and a free slot in 30 minutes.

Through the app, Sofia quickly books the time slot and receives confirmation, including the estimated cost and the address. She uses the mobile interface to track charging progress in real time.

Later, while waiting for her client, she accesses her profile and checks her cars autonomy usage history to compare charging costs for the week.

Persona 2: Jorge Silva

- **Age**: 47
- **Location**: Braga, Portugal
- **Occupation**: Station Operator Manager (Administrator Role)

- **Education**: Degree in Electrical Engineering
- **Tech proficiency**: Moderate – proficient with business dashboards and scheduling platforms
- **Tools used**: Excel, scheduling tools, mobile phone

Background:
Jorge has worked in infrastructure and utilities for over two decades. In recent years, he transitioned into the electric mobility sector, managing a regional network of EV charging stations in northern Portugal. His daily tasks include monitoring the operational status of chargers, handling maintenance, ensuring optimal customer service, and reporting energy metrics to the company's central office. He is deeply familiar with the technical side of charging infrastructure but is often slowed down by the limitations of the current fragmented software tools.

Frustrations:
Jorge often struggles with fragmented systems that don't offer real-time status updates or easy configuration of pricing policies (such as off-peak discounts). He wastes valuable time switching between interfaces from different charger providers and lacks a centralized view of operational metrics.

Motivations and Goals:
Jorge wants to minimize charger downtime, respond quickly to incidents, and optimize station usage. He needs a platform that allows him to monitor availability, apply maintenance tags, configure pricing dynamically, and access usage statistics across his stations—all from a single interface.

Scenario: Monitoring and responding to a charger malfunction

Jorge is conducting his daily operational check of the EV charging stations under his management in Braga. Suddenly, he receives a complaint from a driver stating that a charger at a key location in the city center is not functioning properly. In the past, Jorge would have to remotely access each charger's vendor-specific interface, interpret inconsistent error logs, and coordinate maintenance manually, losing precious time in the process.
Instead, Jorge opens the ChargeMate platform and immediately sees a real-time dashboard with all his station statuses. The problematic charger is flagged in red, already tagged by the system as "Needs Maintenance" based on automatic error detection and telemetry data from the charger.

Jorge quickly assigns that charger as "Under Maintenance" to avoid user reservations to that specific charger and reconfigures nearby stations to offer temporary off-peak pricing to redirect traffic and avoid user frustration. Within minutes, the issue is under control, with reduced downtime and improved user satisfaction.

<Develop one or more representative scenarios for each persona. You don't need to include all possible details. Pick the main scenarios, related to the core value of the system.>
<The scenarios tell the story of the Personas in their lives, doing their daily/professional activities that are relevant to find the points of contact with the system under specification.
Scenarios are somewhat similar to use cases (they have a goal and tell a story), but, unlike use cases, they capture a larger process, with activities that may not use the software. Scenarios don't required a "template", like the usual use cases description.>
Sample: secção 4.2 neste artigo (open access)] >

2.3 Project epics and priorities

[Apresentar um plano indicativo para a implementação incremental da solução ao longo de várias iterações/releases, explicando as funcionalidades a atingir por *epics*]

To ensure an incremental and manageable development, the project is organized into epics, each corresponding to a major area of functionality in the EV charging ecosystem. These epics will be implemented iteratively across multiple iterations, following Agile principles. The priorities were established based on core user needs (from both EV drivers and station operator managers), technical feasibility, and their role in enabling other functionalities.

The application is designed to support two main types of users:

- **EV Drivers**, who interact with the system to locate, reserve, and pay for charging services, through the EV Driver interface.

kk) **Station Operator Managers (Administrators)**, who use a dedicated administrative dashboard to monitor, manage, and configure the charging infrastructure, including operational oversight, pricing management, and performance analytics.

Epic 1 – User Profile Management (EV Driver & Station Operator)

Enables creation of user accounts for both EV drivers and station operators.

UC1.1 – Register new EV Driver profile

As a new EV driver, I want to create a personal profile so that I can start using the app with access to features for electric vehicle owners.

Acceptance Criteria

ll) Given the registration form is displayed,
mm) When the user submits valid personal data (name, email and password),
nn) Then a new EV driver account must be created,
oo) And the user must be redirected to the EV driver dashboard,
pp) And a success message must be shown,
qq) And the user data must be persisted in the database.

Priority: 5
Difficulty: 2

UC1.2 – Register new Station Operator profile

As a new station operator, I want to register and validate my company and personal credentials so I can access the operator dashboard and manage my infrastructure.

Acceptance Criteria

rr) Given the operator registration form is shown,
ss) When valid company and user credentials are submitted,

tt) Then the system must create a new operator account,
uu) And grant access to the operator dashboard,
vv) And securely store the operator's data.

Priority: 5
Difficulty: 2

UC1.3 – Login as EV Driver

As an EV driver, I want to log in to my account so that I can access my dashboard and manage my charging sessions.

Acceptance Criteria (testable):

- Given the login page is displayed,
- When the user enters valid EV driver credentials,
- Then the system must authenticate the user,
- And redirect them to the EV driver dashboard,

Priority: 5
Difficulty: 2

UC1.4 – Login as Station Operator

As a station operator, I want to log in to my operator account so I can access and manage my station infrastructure.

Acceptance Criteria (testable):

- Given the login page is displayed,
- When the user enters valid station operator credentials,
- Then the system must authenticate the user,
- And redirect them to the operator dashboard,

Priority: 5
Difficulty: 2

Epic 2 – Charging History & Analytics

Allows EV drivers to view past sessions and station operators to analyze performance data.

UC2.1 – View charging history (EV Driver)

As an EV driver, I want to view my charging history to track my behavior and expenses.

Acceptance Criteria

ww) Given the driver has previous sessions,
xx) When they visit the "Charging History" section,

yy) Then they must see sessions listed with:
- Date
- Station
- Duration
- Energy used (kWh)
- Total cost

zz) And sessions must be sorted by most recent,
aaa) And all data must match backend records.

Priority: **4**
Difficulty: **2**

UC2.2 – View usage analytics (Station Operator)

As a station operator, I want to see usage metrics to plan pricing and maintenance.

Acceptance Criteria

bbb) Given the operator is in the analytics dashboard,
ccc) When they select a station,
ddd) Then KPIs must be shown, including:
- Total sessions
- Total energy delivered (kWh)
- Average session duration

Priority: **4**
Difficulty: 3

UC2.3 – Mark favorite stations (EV Driver)

As an EV driver, I want to mark stations as favorites to access them quickly in the future.

Acceptance Criteria

eee) Given the user is viewing station details,
fff) When they click “Add to Favorites”,
ggg) Then the station must be saved to their profile,
hhh) And be visible in a “Favorites” list,

Priority: 3
Difficulty: 1

Epic 3 – Station Discovery (EV Driver)

Allows EV drivers to find and evaluate charging stations based on real-time data.

UC3.1 – Locate nearby charging stations

Acceptance Criteria

iii) Given the user is authenticated and location is enabled,
jjj) When they open the discovery map,
kkk) Then nearby stations must appear,
lll) And show:
- Station name
- Availability
- Charger type

Priority: 5
Difficulty: 3

UC3.2 – Filter stations by preferences

Acceptance Criteria

mmm) Given the filters panel is open,
nnn) When filters are applied (e.g., provider, availability),
ooo) Then only matching stations must be displayed on the map and list.

Priority: **4**
Difficulty: 3

UC3.3 – View station details

Acceptance Criteria

ppp) Given a station is selected,
qqq) When its detail screen loads,
rrr) Then it must show:
- Address
- Pricing
- Charger types
- Number of plugs available

Priority: 3
Difficulty: 2

Epic 4 – Slot Booking & Scheduling

Enables EV drivers to reserve and manage time slots at charging stations.

UC4.1 – Book a charging slot

Acceptance Criteria

sss) Given the driver is logged in,
ttt) When an available slot is selected and confirmed,
uuu) Then the booking must be created,
vvv) And the slot must become unavailable for others.

Priority: 5
Difficulty: 3

UC4.2 – Prevent double booking

Acceptance Criteria

www) Given a slot is already reserved,
xxx) When another driver selects it,
yyy) Then the system must block the action,
zzz) And a message that the charger is in use is shown.

Priority: **4**
Difficulty: 3

UC4.3 – Cancel reservation

Acceptance Criteria

aaaa) Given the user has an upcoming booking,
bbbb) When they click “Cancel”,
cccc) Then the slot must be released and shown as available again.

Priority: 3
Difficulty: **2**

Epic 5 – Station Management (Operator Manager)

Tools for operators to manage chargers, pricing, and infrastructure.

UC5.1 – View real-time charger status

Acceptance Criteria

dddd) Given the operator opens the overview,
eeee) Then each charger must display its live status:

- Available
- In Use
- Under Maintenance
- Offline

Priority: **4**
Difficulty: **2**

UC5.2 – Tag charger as under maintenance

Acceptance Criteria

ffff) Given the charger view is open,
gggg) When “Mark as Maintenance” is selected,
hhhh) Then the charger status must become”Under Maintenance”.

Priority: 3
Difficulty: **2**

UC5.3 – Configure off-peak pricing

Acceptance Criteria (testable)

iiii) Given the operator is editing pricing rules,
jjjj) When they set a time range and discount rate,
kkkk) Then the discount must apply during those periods.

Priority: 3
Difficulty: 3

UC5.4 – View performance KPIs

Acceptance Criteria (testable)

llll) Given a station and time range are selected,
mmmm) Then charts must display:

- Total kWh delivered
- Number of bookings
- Average session time

Priority: 3
Difficulty: 3

Epic 6 – Payment & Receipts (EV Driver)

Payment and billing logic for completed sessions.

UC6.1 – Execute payment

Acceptance Criteria

nnnn) Given a session has ended,
oooo) When the driver confirms payment,
pppp) Then the system must:

- Show total cost
- Execute payment via selected method
- Return a confirmation or failure message

Priority: 2
Difficulty: 4

UC6.2 – View payment history

Acceptance Criteria

qqqq) Given the user opens the payment history,
rrrr) Then they must see a list of payments with:

- Date
- Method
- Amount

Priority: 1
Difficulty: 2

3 Domain model

<which information concepts will be managed in this domain? How are they related?>
<use a logical model (UML classes) to explain the concepts of the domain and their attributes, not a entity-relationship relational database model>

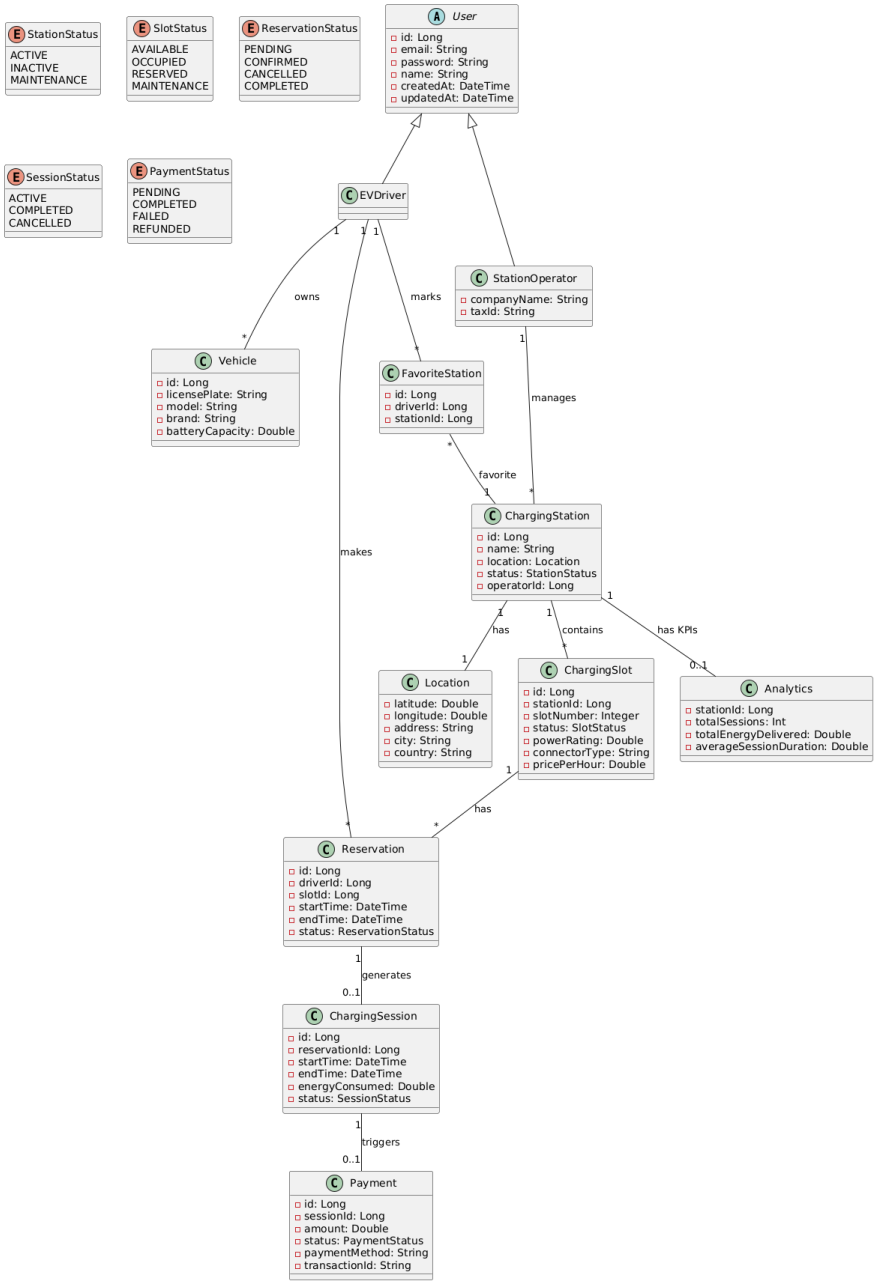

Figure 1. Domain model diagram.

4 Architecture notebook

4.1 Key requirements and constrains

<**Identify issues that will drive the choices for the architecture** such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions? Are there integrations with external systems? Is the system to be offered in different user-interfacing platforms (web, mobile devices, big screens,...)?
For a more systematical approach:

- Note the collection of Architectural Characteristics the software architect should be aware
- Identify architectural characteristics that are relevant for your project (will drive the key design decisions). Note the case study and the explicit characteristics related to users and extensibility. This will support later non-functional tests.

User Interface Requirements

The system is delivered exclusively through a web-based interface.
Two distinct user interfaces are provided:

ssss) EV Driver Portal: For vehicle owners to manage charging sessions, reservations, and view their history.

tttt) Station Operator Dashboard: For station managers to monitor and manage stations, view analytics, and configure pricing.

The interface is responsive and accessible for standard web browsers (desktop and mobile).

Integration Requirements

External System Integrations:

uuuu) Payment Gateways: For processing charging session payments.

Real-time Requirements

The system must handle critical real-time data, including:

vvvv) Station status updates (available, occupied, maintenance)

wwww) Active charging session monitoring

xxxx) Reservation status changes

WebSocket technology is used for efficient real-time communication between the backend and the web client.

Scalability Requirements

The system is designed to support:

yyyy) At least 500 concurrent users at launch

zzzz) A growing number of charging stations and concurrent charging sessions

aaaaa) Horizontal scaling of the backend and database as needed

Security Requirements

bbbbb) Secure user authentication and authorization

ccccc) Secure payment processing via external gateways

ddddd) Role-based access control for different user types (EV driver, station operator)

Performance Requirements

eeeee) Response time: Less than 2 seconds for standard operations

fffff) High availability: 99% uptime

ggggg) Efficient handling of concurrent charging sessions

Data Management Requirements

hhhhh) Reliable storage of user profiles, preferences, station configurations, charging session records, and payment transactions

iiiii) Regular backups and recovery capabilities

Key Architectural Decision Drivers

Modularity and Extensibility

jjjjj) The backend is structured in a layered architecture (Presentation, Application, Domain, Infrastructure).

kkkkk) Service classes in the backend are modular, each responsible for a specific business area (user, station, reservation, analytics, operator management).

lllll) Integration with an external system (payment) is encapsulated in a dedicated module, making it easy to add or replace providers.

Real-time Communication

mmmmm) WebSockets are used for real-time updates to the web client (station status, session monitoring, reservation changes).

nnnnn) The architecture avoids unnecessary complexity by not using a message broker; all real-time needs are handled directly by the backend.

Data Consistency

ooooo) All critical data (reservations, payments, sessions) is stored in a single PostgreSQL relational database.

ppppp) Transaction management ensures data consistency, especially for reservation and payment operations

Security and Compliance

qqqqq) Secure authentication mechanisms (e.g., JWT)

rrrrr) Data encryption in transit (HTTPS via Nginx reverse proxy)

sssss) Secure API endpoints and role-based access control

ttttt) Payment security is ensured by delegating processing to trusted external gateways

Scalability and Performance

uuuuu) The system is containerized with Docker for easy deployment and scaling.

vvvvv) The backend and database can be horizontally scaled as demand grows.

wwwww) Nginx is used as a reverse proxy for load balancing, SSL termination, and static content delivery.

xxxxx) Caching strategies can be added if needed, but are not part of the initial simplified architecture.

These requirements and constraints guided the architectural decisions made, ensuring that the system met both functional and non-functional requirements while maintaining flexibility for future extensions and modifications.

4.2 Architecture view

→ Discuss architecture planned for the software solution: what are the main building blocks? include a diagram (a logical view, such as a package or block diagram). Avoid implementation technology or deployment references, but protocols/standards can be included.
→ refer to the architecture style applied, if any
→ explain how the identified modules will interact. Use a sequence diagram to clarify the interactions along time, when needed
→ discuss more advanced app design issues: integration with Internet-based external services, data synchronization strategy, distributed workflows, push notifications mechanism, distribution of updates to distributed devices, etc.>

The ChargeMate system follows a layered monolithic architecture with clear separation of concerns.

Presentation Layer:

The web client (React + Vite + Tailwind) provides a responsive interface for both EV drivers and station operators, communicating securely with the backend via HTTPS.

Reverse Proxy (NGINX):

NGINX is used as a reverse proxy between the web client and the backend. It handles SSL termination (HTTPS), routes client requests to the appropriate backend services, and serves static frontend assets. NGINX also provides load balancing, security headers, and improves overall system performance and scalability. Its use abstracts the backend from direct exposure to the internet, adding an extra layer of security and flexibility.

Application Layer:

The backend (Spring Boot REST API) implements all business logic, organized into modular services (User, Station, Reservation, Analytics, Operator Management). Each service is responsible for a specific business area.

Domain Layer:

Contains the core business entities and logic, separated into domains (User, Station, Reservation, Analytics, Payment, Operator).
Infrastructure Layer:

Provides technical capabilities, including a PostgreSQL database for all persistent data.

External Integrations:

The backend integrates with an external system for payment. This integration is modular and can be replaced or extended as needed.

Communication Patterns:

yyyyy) Synchronous REST APIs are used for client-backend and backend-external system interactions.

zzzzz) WebSockets are used for real-time updates to the client.

aaaaaa) NGINX reverse proxy ensures secure, efficient, and scalable communication between the client and backend.

Advanced mechanisms such as distributed workflows, push notifications, or data synchronization across distributed devices are not required for the current scope, as the system is designed for web-based access and real-time updates are handled via WebSockets.

This architecture ensures the system meets both functional and non-functional requirements, while remaining simple, maintainable, and extensible for future needs.

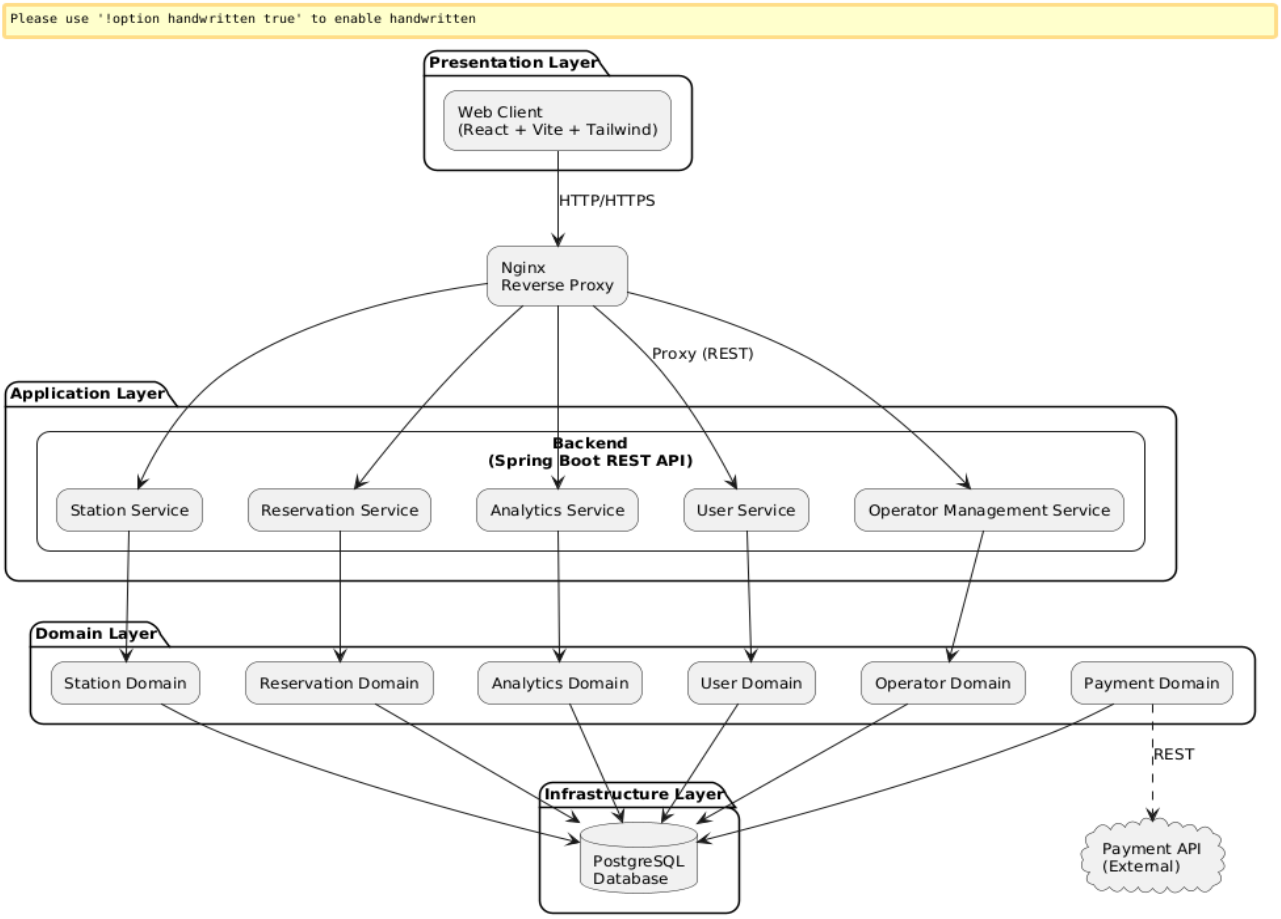

Figure 2. Logical view architecture diagram.

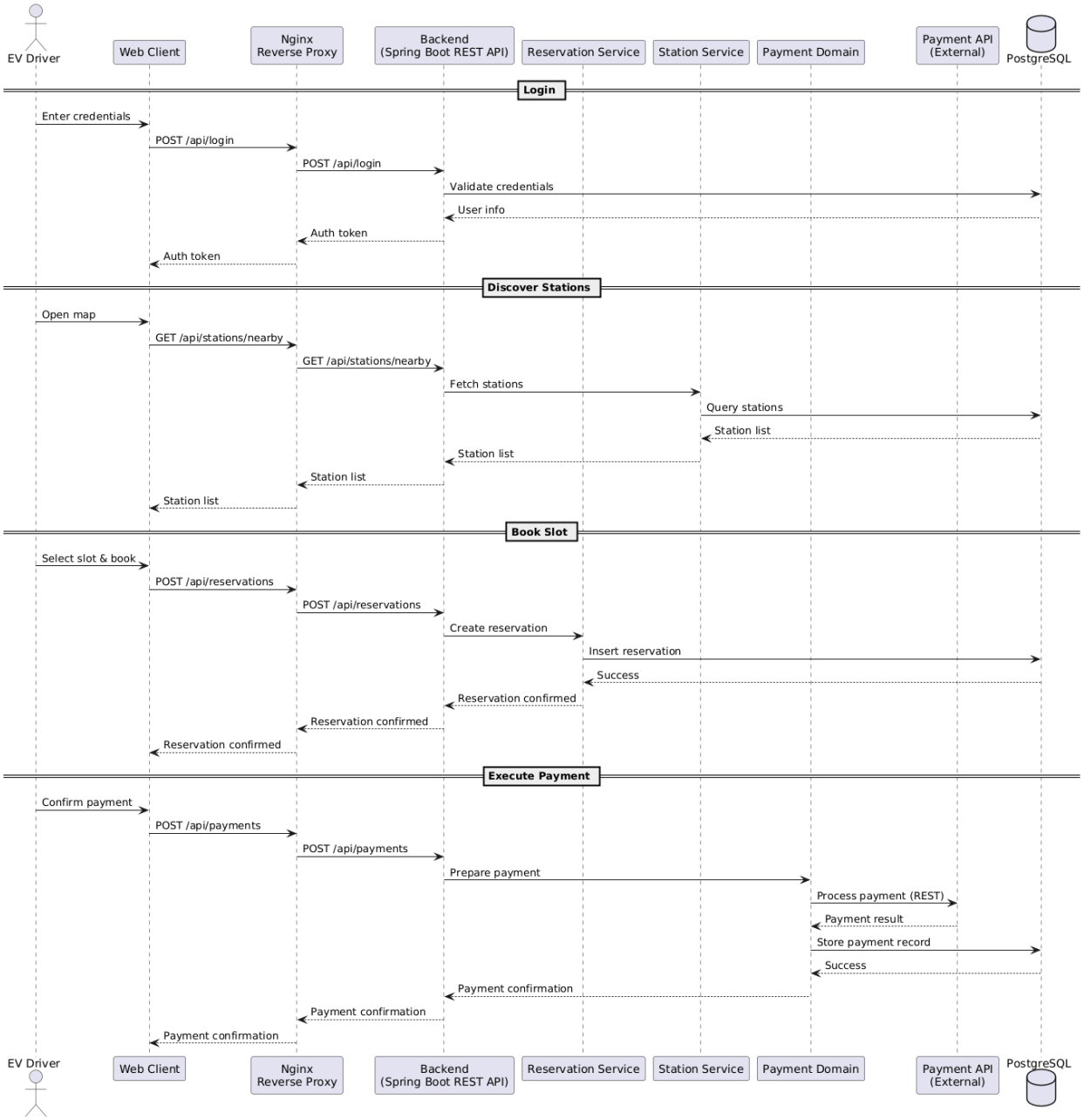

Figure 3. Sequential diagram for EV Driver actor.

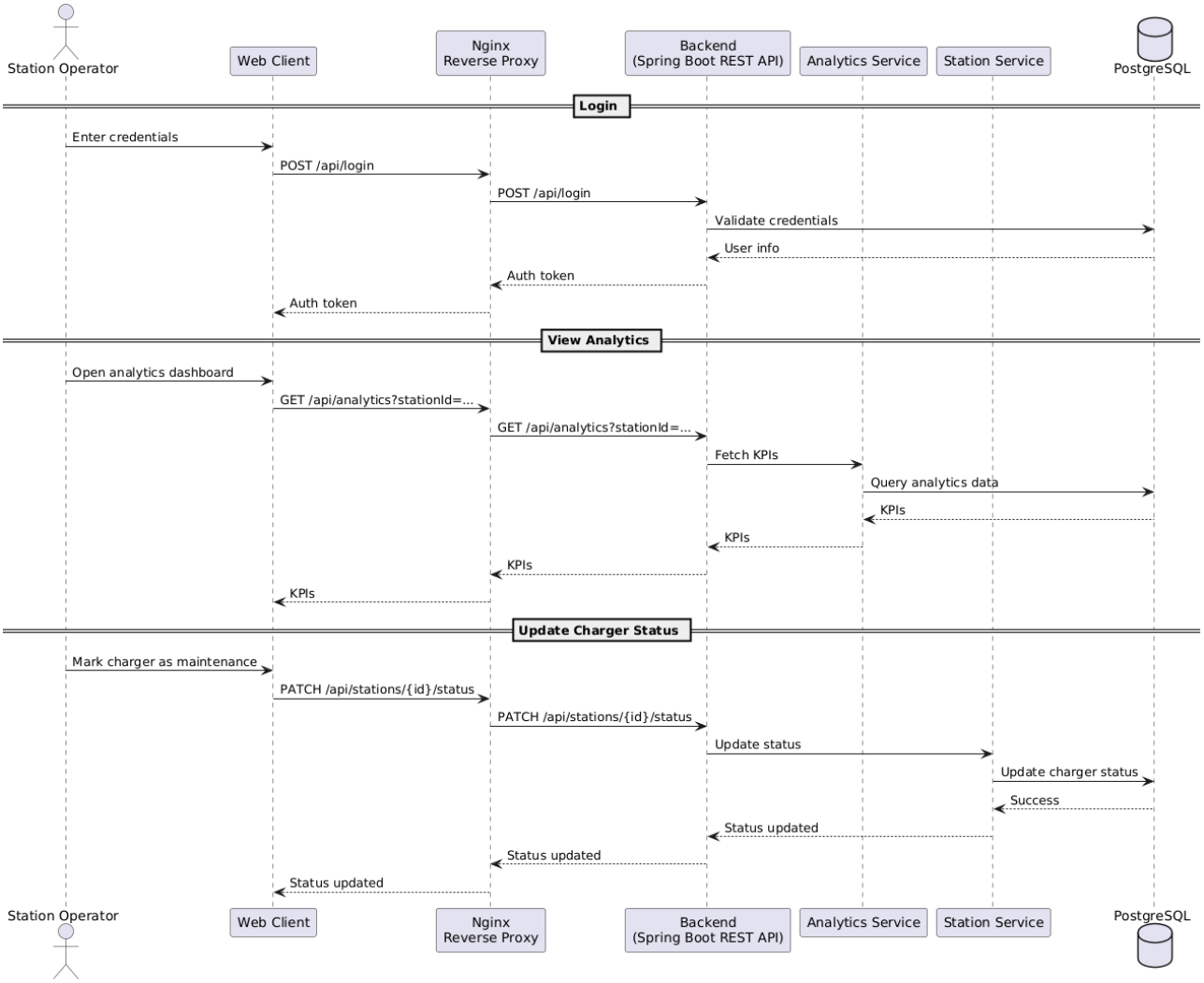

Figure 4. Sequential diagram for Station Operator actor.

4.3 Deployment view

[Explicar a organização prevista da solução em termos configuração de produção (*deployment*). Anotar, no diagrama, as tecnologias de implementação, e.g.: colocar o simbolo do PostgreSQL na Base de dados,...]. Indicar a existência de containers (Docker), endereços IP e portos,...
Esta parte será completada quando houver efetivamente deployments

5 API for developers

[Explicar genericamente a organização da API e coleções principaus. Os detalhes/documentação dos métodos devem ficar numa solução *hosted* de documentação de APIs, como o Swagger, Postman documentation, ou incluída no próprio desenvolvimento (e.g.: maven site)
→ Be sure to use best practices for REST Api design. Keep minda REST API applies a resource-oriented design (APIs should be designed around resources, which are the key entities your application exposes, not actions)