Politecnico di Milano
School of Industrial and Information Engineering
Master of Science in Computer Science and Engineering

Software Engineering 2 Project

eMall - CPMS system

Design Document

Authors:
Federico Bono
Daniele Cipollone

Academic Year 2022/2023

Milano, 08/01/2023
Version 1.1

Contents

1 Introduction **3**
1.1 Purpose . . . 3
1.2 Scope . . . 3
1.3 Revision History . . . 4
1.4 Definitions, Acronyms, Abbreviations . . . 4
1.4.1 Definitions . . . 4
1.4.2 Acronyms . . . 5
1.5 Related Documents . . . 5
1.6 Document structure . . . 5

2 Architectural Design **6**
2.1 Overview: high-level components and interactions . . . 6
2.2 Component view . . . 8
2.2.1 Class diagram . . . 11
2.3 Deployment view . . . 12
2.4 Runtime view . . . 13
2.4.1 Generic Routing and Authorization . . . 13
2.4.2 Start charging session . . . 13
2.4.3 Book a charging session . . . 14
2.5 Selected architectural style and patterns . . . 14

3 User Interface **15**
3.0.1 Log in to the web portal . . . 15
3.0.2 View energy sources report . . . 16
3.0.3 View external status informations . . . 17

4 Requirements Traceability **18**
4.1 Functional requirements . . . 18
4.2 Components mapping . . . 19
4.3 Components mapping on Requirements . . . 20

5 Implementation, Integration and test Plan **21**
5.1 Plan details . . . 21
5.2 Additional testing . . . 23

6 Effort spent **24**

7 References **25**

List of Figures

2.1 Overview of the chosen three-tier architecture with actors 7
2.2 Component diagram of the system 9
2.3 Class diagram for the main models of the System 11
2.4 Deployment diagram for the System 12
2.5 Generic Routing and Authorization flow for a generic request 13
2.6 Start charging session 13
2.7 Book a charging session 14

3.1 Log in to the web portal 15
3.2 View energy sources report 16
3.3 View external status informations 17

1 Introduction

1.1 Purpose

The purpose of this document is to describe and explain in details the design choices for the development and deployment of the CPMS system of eMall. The description is structured on multiple levels to give an overview of the system from multiple viewpoints. In the following pages you will find:

1. High level overview of the architecture
2. Overview of the components of the system
3. Deployment overview
4. Overview of the interactions between components
5. Overview of the interfaces offered by the various components
6. The UI of the mobile application used by the CPO
7. The patterns and technologies used in the system

1.2 Scope

eMall App is a platform that helps the end users to plan the charging process, by getting information about Charging Points nearby, their costs and any special offer; book a charge in a specific point, control the charging process and get notified when the charge is completed. It also handles payments for the service.

In the e-Charging ecosystem, there are many different actors involved that we need to keep into consideration while collecting requirements and designing the system. The first information to consider is that Charging Points are owned and managed by Charging Point Operators (CPOs) and each CPO has its own IT infrastructure, managed via a Charge Point Management System (CPMS).

Another actor is the eMSP, which manages the end user's requests, forwarding the booking and top-up management requests to the CPMS. In order to comunicate with the various eMSP, the OCPI (Open Charge Point Interface) protocol is used.

While to communicate with the various charging slots, the OCPP (Open Charge Point Protocol) is used.

The CPMS has the task of managing the energy used at the charging points owned by the CPO. To do this, it must communicate with the DSOs, which supply the energy to the various charging stations.

1.3 Revision History

- v1.0 - 03 January 2023
- v1.1 - 06 January 2023

1.4 Definitions, Acronyms, Abbreviations

1.4.1 Definitions

Connector Charging Socket	Physical connector that allow to transfer energy to the connected vehicle
Charging slot	Physical device with multiple Connectors that can charge electric vehicles. NOTE: from OCPI definitions a Charging Slot can have up to one connector active at any time (i.e. can charge only one Vehicle at any time)
Charging Point	Physical structure composed by multiple Charging Slots
Maintenance of a charging slot	Activity/activities that results in a momentary unavailability of the charging slot
Charging session	period of time when the vehicle is connected to a charging plug for charging
Booking period	period of time between the booking of a charging session and the beginning of the charging session
Internal status	The amount of energy stored in the batteries, if any, the quantity of vehicles currently being charged, and the current energy management settings.
External status	Number of charging sockets available, their type such as slow/-fast/rapid, their cost, and, if all sockets of a certain type are occupied, the estimated amount of time until the first socket of that type is freed.

1.4.2 Acronyms

eMSP	e-Mobility Service Provider
CP	Charge Point / Charging Point
CS	Charge Slot / EVSE
CPO	Charging Point Operator
CPMS	Charging Point Management System
OCPI	Open Charge Point Interface
EV	Electric Vehicle
OCPP	Open Charge Point Protocol
DSO	Distribution Management System

1.5 Related Documents

- CPMS RASD (RASD_CPMS.pdf)
- OCPI specifications document (OCPI-2.2.1.pdf)
- OCPP document (ocpp-1.6.pdf)

1.6 Document structure

The document is structured in six sections:

1. Description and introduction of the various design choices made during the design of the system. Descriptions are written at different levels of abstractions: from the general point of view to the detailed view of the single component.
2. User interfaces and design mockups.
3. The requirement traceability matrix is used to map each component to the requirement(s) that fulfils.
4. Implementation and test plans for the entire system
5. Total effort
6. References used

2 Architectural Design

2.1 Overview: high-level components and interactions

As anticipated in the previous chapter, the architecture selected for the design and development of the system is the three-tier architecture.

This architecture allow us to split the implementation into three layers:

1. Presentation: it is the online web page that will be used by the CPOs to access all the information and settings of their charging stations
2. Application: is the backend of the system, dove sono implementati all the logics related to the energy management by the stations, the management of reservations, the interactions with the various eMPS and the DSOs.
3. Data: is the layer responsible to expose connectivity interfaces from the database. It will be a DBMS.

All the layers communicate in a linear way: the Presentation one interacts only with the Application layer, the same as the Data layer. With this architecture the presentation and the Data layers have no direct communication path. Other advantages of using this architecture are that the various layers can be developed with different technologies and that they can be duplicated and differentiated.

The main reasons behind the choice are:

- We are not the producer of the data
- We need to integrate different external services

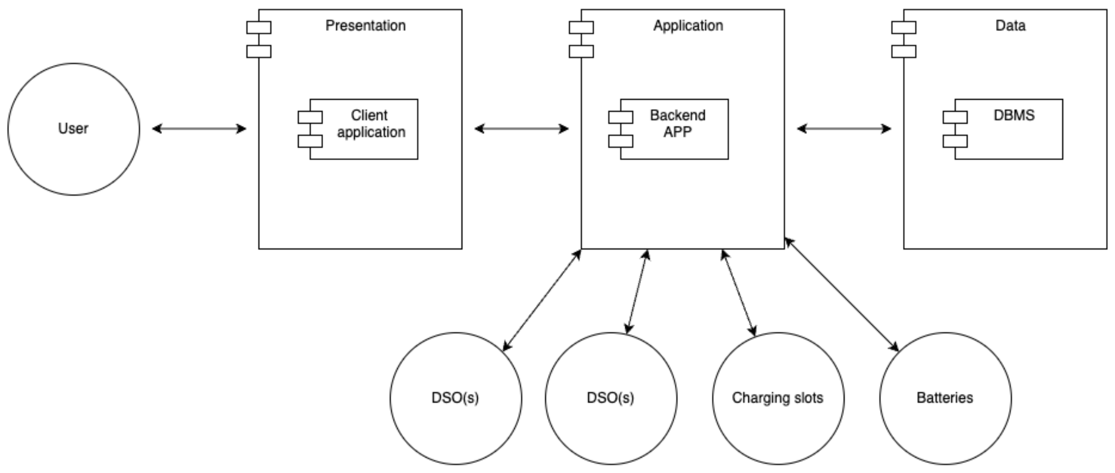


Figure 2.1: Overview of the chosen three-tier architecture with actors

2.2 Component view

The following schema shows all the main components and interfaces of the system. Later on you will find the details of each component.

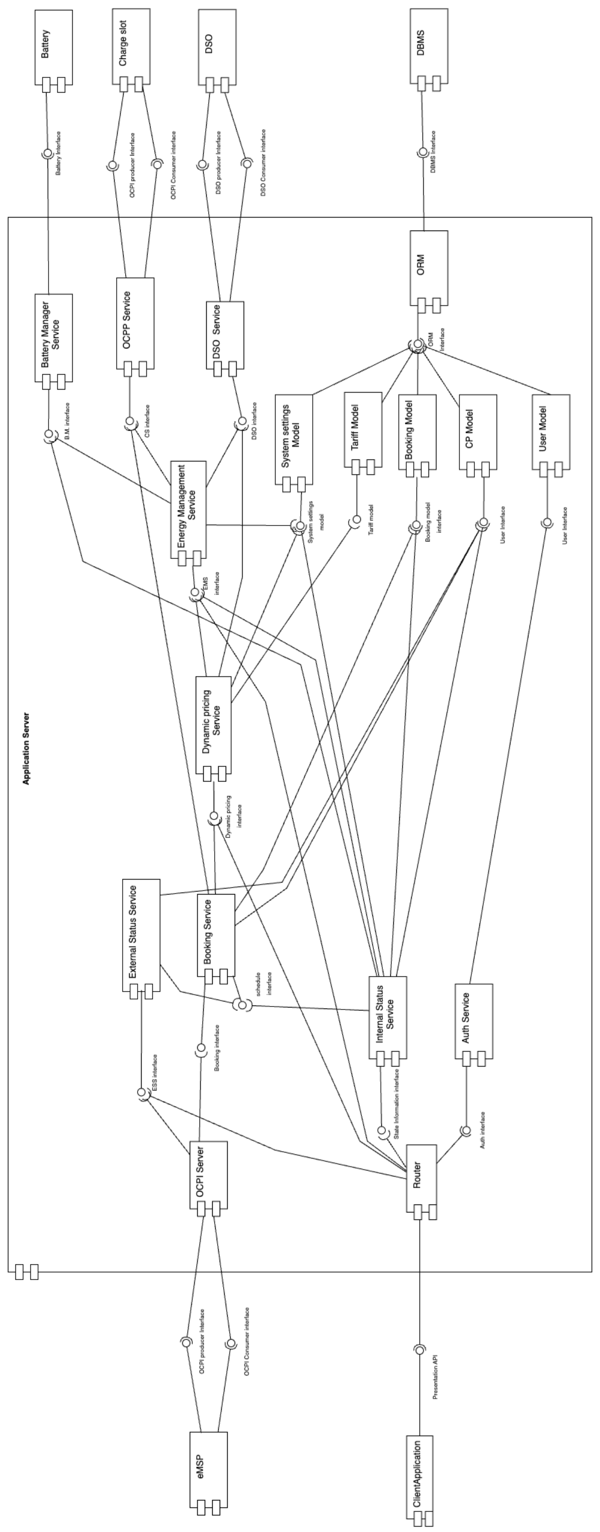


Figure 2.2: Component diagram of the system

- **Application server**: not a real "component" it shows the division between the backend, the frontend, the data layer and the external service. In the three-tier architecture it represents the Application layer.
- **eMSP**: e-Mobility Service Provider that exposes OCPI compliant API
- **ClienApplication**: web application for the system, used by the users.
- **External Status Service**: Service that provides information regarding the external status of the charging station. informations like number of charging sockets available, their type such as slow/fast/rapid, their cost, and, if all sockets of a certain type are occupied, the estimated amount of time until the first socket of that type is freed.
- **Internal Status Service**:Service that provides information regarding the internal status like: the amount of energy stored in the batteries, if any, the quantity of vehicles currently being charged, and the current energymanagement settings.
- **Dynamic pricing Service**:Service that manages and imposes the tariff for each type of charge and period, taking into consideration the current energy price and the current energy management settings. A tariff can also be set using a special function.
- **Energy Management Service**: Component that dynamically decides which DSOs to draw energy from, controlling the use of the batteries present at the charging stations
- **DSO Service**: it manages the communication with the Distribution System Operators DSO
- **OCPP Service**: it manages the communication with the Charge slot (EVSE)
- **Carge slot**: Physical device with multiple Connectors that can charge electric vehicles
- **Battery**: Battery physically present at some charging stations
- **Router**: handles all the requests directed to the Application APIs that comes into the Application Server, uses the Auth Service to Authenticate and Authorize them. After that, if all the checks are passed, it will forward the request to the correct service that exposes the required route. It basically acts as a middleware, its presence is useful as it acts as a single point to develop all the authentication/authorization logic so that if it needs changes we can just update the router (and eventually the auth service) instead of updating all the services.
- **Auth Service**: is an internal service that handles authentication (Signup and login) and authorization (even if at the moment our system does not need that).
- **Booking Service**: it manages the booking for the application, it exposes some endpoints to the router for the booking functionalities of the eMSP it also manages charging sessions.
- **OCPI Service**: it manages the communication with the eMSP, it exposes the endpoints for the PUSH part of the protocol (is needed to send real-time updated to eMSP)
- **ORM**: a library that allows an easy to use mapping between Models and DB table, it handles queries and relationships. It is not worth it to develop an in-house solution, so we will use a library for that.

- **User Model, CP Model, Booking Model, Tariff Model, System settings Model** : Models components that sit between services and ORM, useful to attach event listeners and other effects that need to run when updating the data.
- **DBMS**: Database Management System

2.2.1 Class diagram

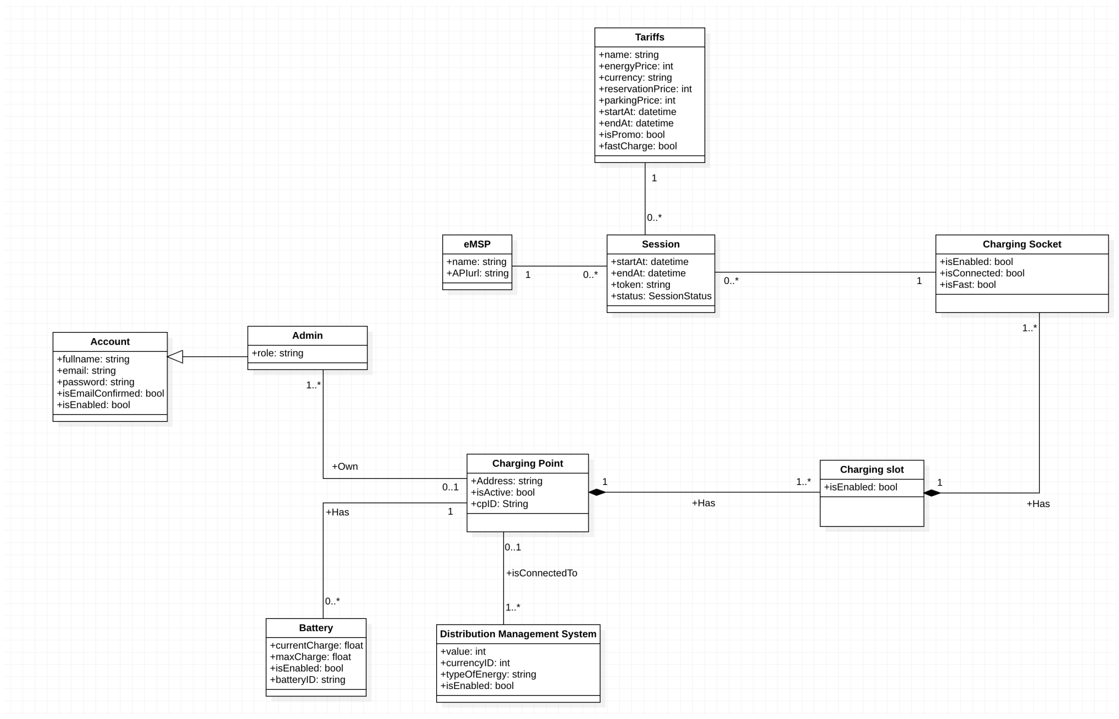


Figure 2.3: Class diagram for the main models of the System

2.3 Deployment view

In this section we introduce a detailed view of the system and various component from a deployment cycle perspective. The following schemes highlight the environments and the tools to be used for the system.

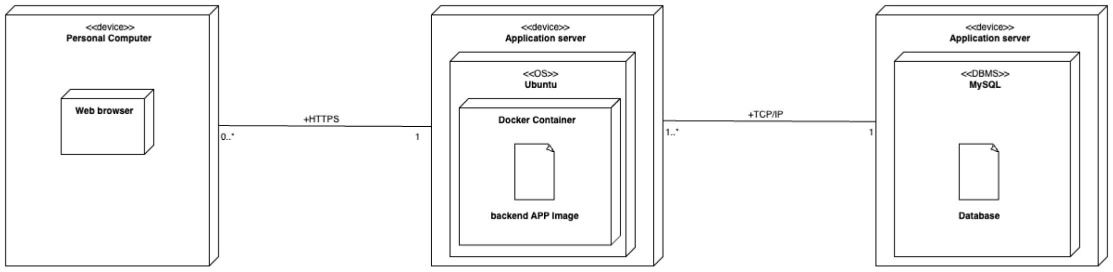


Figure 2.4: Deployment diagram for the System

- **Personal Device**: Personal Computer used by users to run the client web application to communicate with the system
- **Backend Server / Virtual Private Server**: Physical device where is installed the Docker Image of the Backend APP. Multiple replicas are possibile and preferable as they will increase both reliability and availability of the system.
- **Database Server**: Host the database of the system. Multiple local replicas can improve read performances but also increase complexity.

2.4 Runtime view

In this section are shown the sequence diagrams for the various functionalities of the system. To simplify the description there is an initial generic sequence the summarize the routing and authorization part of the request handling. Also, as all the request go through the router, the actors that sends the request to the router are omitted.

2.4.1 Generic Routing and Authorization

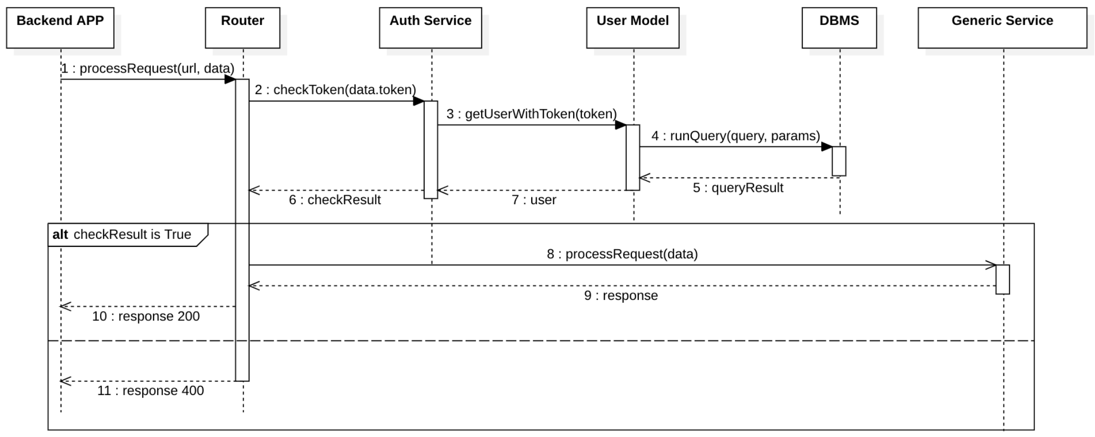


Figure 2.5: Generic Routing and Authorization flow for a generic request

When a request is received by the application web server, it goes through the Router component. It acts as an authorization middleware. As Authorization is needed, the whole Auth Service, Model and DBMS components are envolved. At the end, if authorization has been completed successfully, the request is forward to the dedicated service.

2.4.2 Start charging session

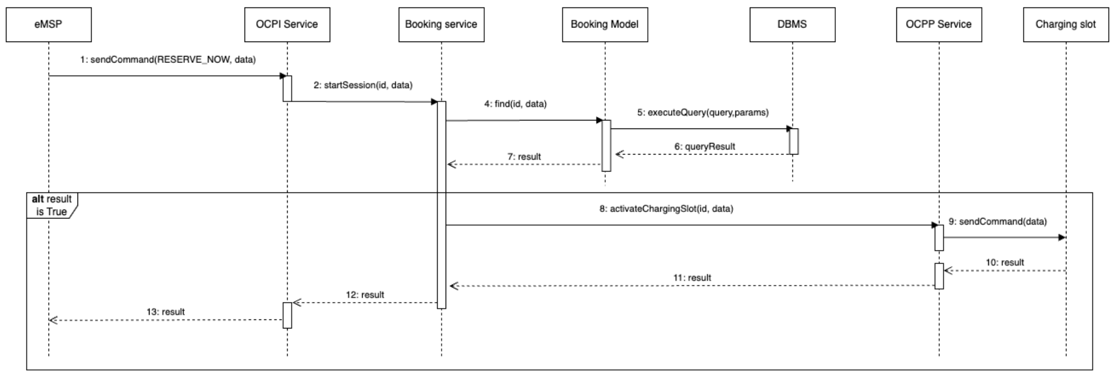


Figure 2.6: Start charging session

2.4.3 Book a charging session

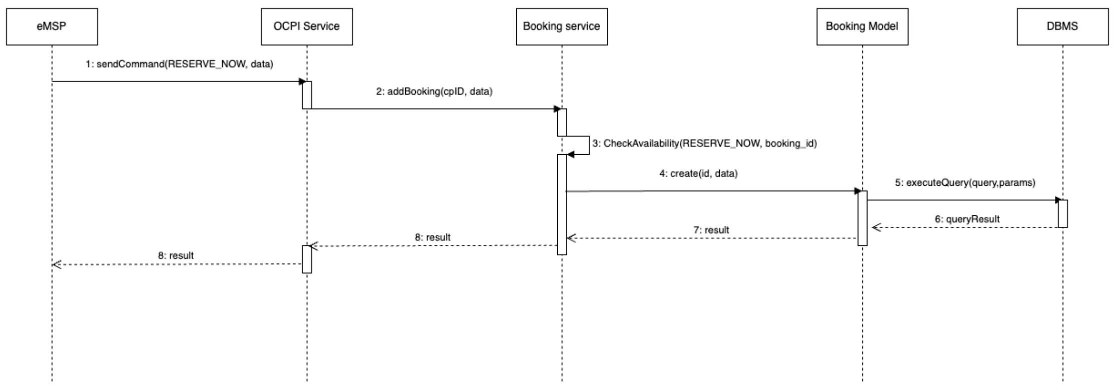

eMSP
OCPI Service
Booking service
Booking Model
DBMS
1: sendCommand(RESERVE_NOW, data)
2: addBooking(cpID, data)
3: CheckAvailability(RESERVE_NOW, booking_id)
4: create(id, data)
5: executeQuery(query,params)
6: queryResult
7: result
8: result
8: result

Figure 2.7: Book a charging session

2.5 Selected architectural style and patterns

- **Three-tier architecture**: As stated before the three-tier architecture was chosen because our system is not the producer of the data, and that there is the need for implementing different functionalities (e.g. payments) and external services.
- **Monolith architecture**: To simplify the development and the deployment of the system, to avoid the so called "vendor lock-in" by choosing a specific service provider for the microservices infrastructure and mainly because no usage spikes are expected, the monolith architecture has been chosen. That means that all the application server replicas contains the whole backend application and not the single services.
- **Authentication with Access Token**: As there are no specific requirements on the authentication functionality, the simply access token pattern can be considered a good suit for the system.

3 User Interface

3.0.1 Log in to the web portal

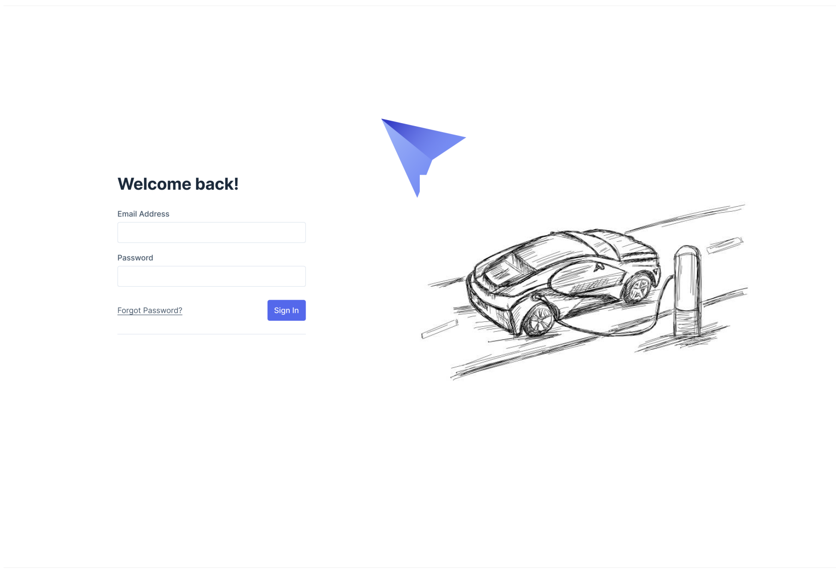
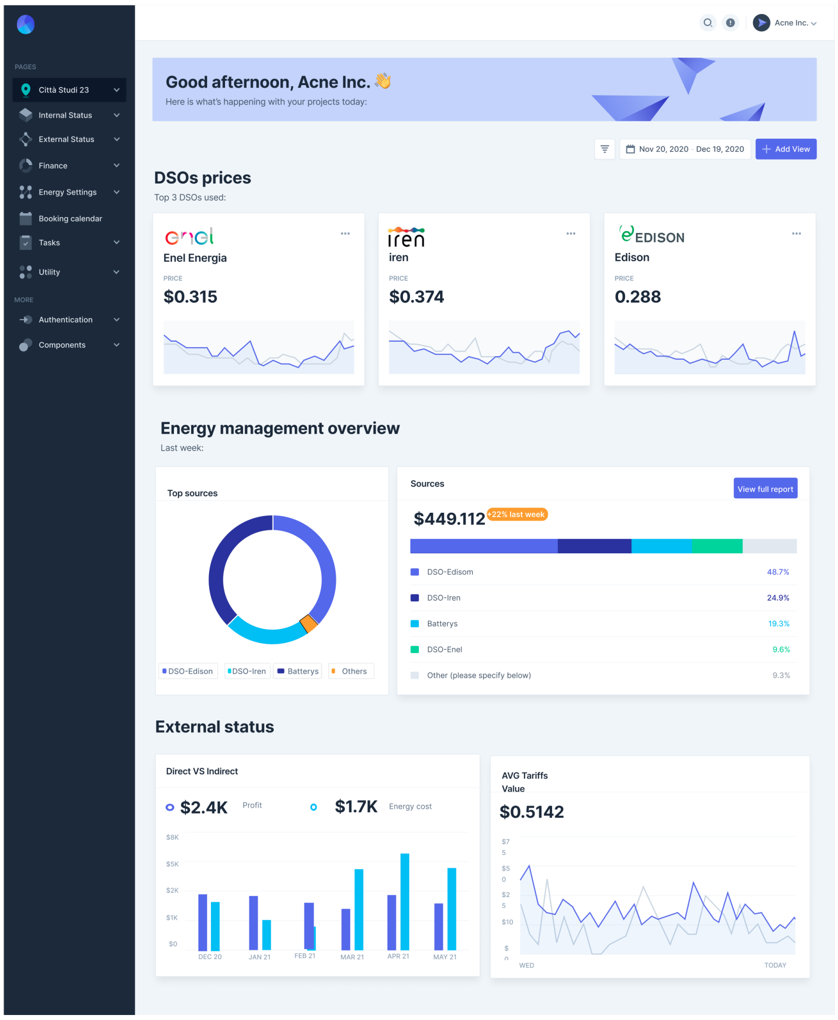


Figure 3.1: Log in to the web portal

In this first view, the user opens the site hosted by his own CPMS, and enters his credentials to access the pages dedicated to him.

3.0.2 View energy sources report



DSO-Edisom	48.7%
DSO-Iren	24.9%
Batterys	19.3%
DSO-Enel	9.6%
Other (please specify below)	9.3%

Figure 3.2: View energy sources report

The CPO administrator wants to view the report regarding the energy sources used during the last month. Then access the portal using your credentials. Then select the relevant charging station, using the drop-down menu on the left. Then press the "View full report" button. The site will display a file containing the requested data.

3.0.3 View external status informations

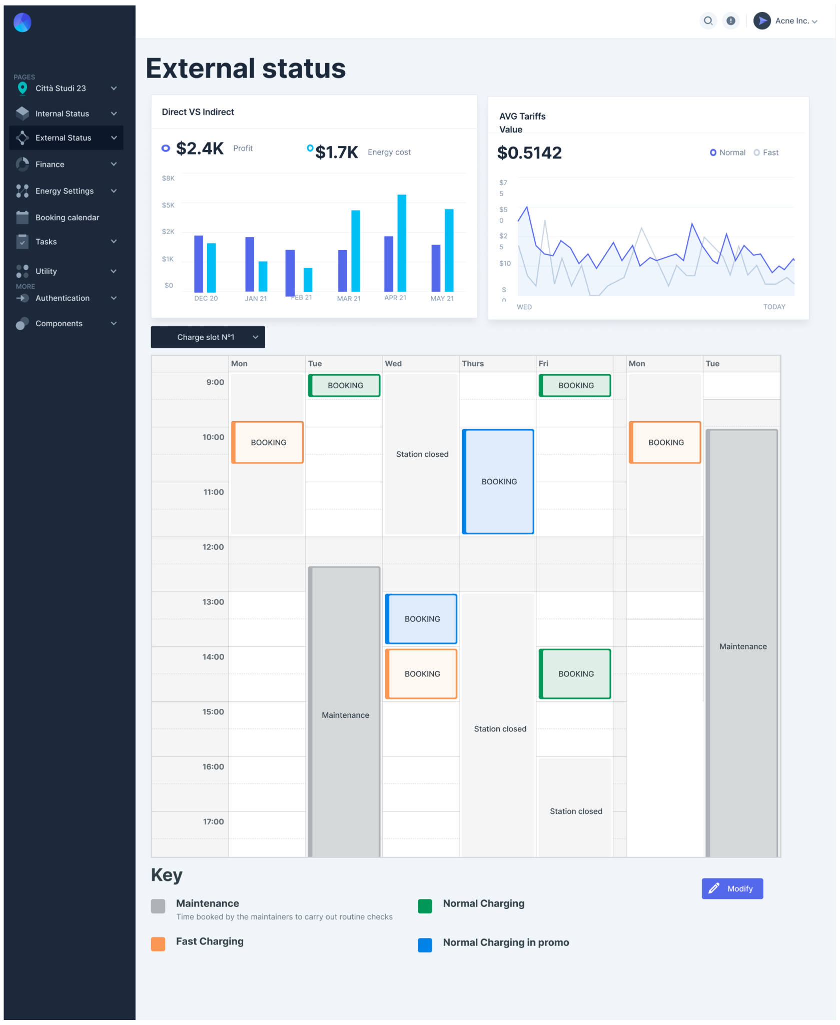


Figure 3.3: View external status informations

The CPO wants to view information regarding the external status of one of its charging stations. He then accesses the portal using his credentials and selects the Charging station using a special drop-down menu located on the left. After that select the "External status" page. The page containing all the information regarding the external status will then be displayed.

4 Requirements Traceability

4.1 Functional requirements

In the following table are summarized the requirements extracted from the RASD.

Requirement	Description
R1	Allow to book a charging session on a specific CS
R2	Allow to start a charging session on a specific CS
R3	Allow to stop a charging session on a specific CS
R4	Forward information about the state of a charging session to the eMSP
R5	Allow Login for CPO
R6	Allow to view information about the internal status of a charging station
R7	Forward of the offered tariffs
R8	Allow CPO to set or change tariffs
R9	Allow CPO to enter static information about charging stations
R10	Forward information about the location of charging stations
R11	Allow CPOs to view information about the external status of a charging station
R12	Allow CPOs to change the settings and management settings of the charging station
R13	Dynamically decide the tariffs offered

4.2 Components mapping

To keep the mapping simple, we define some abbreviations for the components.

	Component
AS	Application Server
ESS	External Status Service
BS	Booking Service
APP	Client Application
RT	Router
AT	Auth Service
ISS	Internal Status Service
DPS	Dynamic Pricing Service
EMS	Energy Management Service
DSOS	DSO service
OCPPS	OCPP Service
BMS	Battery Manager Service
OC	OCPI Service
OR	ORM
MD	Models (User, CP, Booking Model...)

4.3 Components mapping on Requirements

Comp.	Requirements	Reason
AS	R*	Application server is needed for all the functionalities as it is the foundation for the system
ESS	R4, R7,R11	External Status Service is used to send and analyze data regarding the external state of the system
BS	R1-3	Booking Service manages the "state machine" and the related effects for the charging sessions.
APP	R5-6,R8-9,R12	Client Application is the medium for the communication between the CPO and the system
RT	R5-6,R8-9,R12	Router is the middleware component for the "frontend" functionalities.
AT	R5-6,R8-9,R12	Authentication Service allows to check the state of the user
ISS	R6	Internal Status Service is the service that allows to show and analyze data regarding the internal state of the system
DPS	R1,R7-8,R11	Dynamic Pricing Service is the service capable of calculating and indicating active tariffs, which can be dynamic according to electricity prices
EMS	R2-3,R9,R12	Energy Management Service is the component that allows you to manage energy management policies at the various charging stations
DSOS	R8,R13	It allows the bidirectional communication with the various DSOs
OCPPS	R2-4,R6,R9-10	It allows the bidirectional communication with the various CS/EVSE
BMS	R9,R12-13	Battery Manager Service allows communication with the various batteries and their administration
OC	R1-R4,R7,R10	It allows the bidirectional communication with the various eMSP
OR	R*	It allows to easily access the DBMS APIs and to manage the relationships between the models
MD	R*	It allows to access the ORM interfaces and to trigger functions on models' events (e.g. send a notification when charging is completed)

5 Implementation, Integration and test Plan

5.1 Plan details

To implement the system components we've decided to follow a planned structure, dividing the components into various groups in order to be able to develop them in parallel, using a more complex workforce. Given the limited number of functionalities, we think that a progressive release strategy (i.e. with multiple smaller releases) is not really worth for the final user, that means that the only way to bring value to the user will be to have the system released all at once.
This assumption allows us to follow different strategies, both bottom-up and top-down.
To keep aligned the tests set and the actual codebase, we decided to follow the famous TDD (Test Driven Development) strategy, where unit/integration tests are written before the code and then the actual codebase is developed to fulfil those tests.
To develop the best user experience possible, we decided to follow the top-down approach, splitting the development of the various components by functionalities viewed from the user's point of view. We've can use the goals defined in the RASD as a starting point.

We've defined the following functionalities:

1. Login
2. View the CP locations, filter them and view the details of a location
3. Book a charge session at a specific location for a specific connector
4. Start and complete a charge session
5. View the list of reservations at the various stations and charging slots
6. Independently manage the energy management policy
7. Create adaptive tariffs with respect to energy costs and the presence of batteries
8. View the internal status of the system
9. View the external status of the system
10. Establish a connection with the various DSOs
11. Manage the connection with all the various charging slots present
12. Provide information such as rates and external status to eMSPs

13. Be charged with the correct amount after the charging process

Based on those functionalities we can define the following development path:

1. **Application scaffolding**
   In this phase we set up the servers (from a logical point of view, through docker), the frameworks and the connections with the DBMS. We use the Router component given by the framework, so it can be considered as being developed/integrated in this phase.
2. **Authentication and notification scaffolding**
   In this phase we develop the Authentication Service, the Notification Service and we connect with the chosen Email Provider to send the email confirmation link to the user. The related UI and Models are also developed in this phase, alongside with the needed database migrations.
3. **OCPI service integration**
   We use some open source libraries that implements the needed interface for the OCPI (both PULL and PUSH methods), and wrap it with a utility service.
4. **OCPP service integration**
   We use some open source libraries that implements the needed interface for the OCPP (both PULL and PUSH methods), and wrap it with a utility service.
5. **DSO Service service integration**
   Build an interface for DSOs using their standard
6. **Battery Management Service**
   Implementiamo il servizzio che permette la comunicazioen e gestione delle eventuali batterie.
7. **Models and database**
   We start to develop the the models and the database for the management of the information to be stored within the system
8. **Energy management system**
   We implement the energy management service at the various stations. By creating an interface to allow settings to be changed and saved
9. **Dynamic pricing system**
   We implement the service for dynamic price management
10. **Booking Service**
    In this phase we start to develop the functionalities to book a charge from a specific CP, we need to expose those functionalities from the OCPI Service.
11. **Sessions functionalities**
    In this phase we add the charging management functionalities to both OCPI Service and Booking Service.
12. **External Status Service**
    Implement the service for monitoring the external status of the stations, and make its interfaces available to the OCPI Service

13. **Internal Status Service**
    Implement the service for monitoring the internal status of the stations, and make its interfaces available to the Router

5.2 Additional testing

After development and integration we will run some system-level tests. First test to run is on non-functional requirements, developers need to test accessibility features and base performance, doing so we can easily find the most important bottlenecks of the application. Another test that is useful to run is the so called "chaos test": various components of the system are disconnected on purpose to try to verify the behaviour of the system, the goal here is to check that the system fails safely, without losing or leaking data. The final test that will be run is the acceptance test: the system will be tested with real users in the production environment to verify that is really useful to them.

6 Effort spent

Task	Time spent
Introduction	2 h
Architectural Design	22 h
User interface	5 h
Requirements Traceability	3 h
Implementation, Integration and Test Plan	4 h
Revision	2 h

7 References

- **TDD (Test Driven Development)** - https://it.wikipedia.org/wiki/Test_driven_development
- **Three-Tier Architecture** - https://www.ibm.com/topics/three-tier-architecture
- **OCPI specs** - https://evroaming.org/app/uploads/2020/06/OCPI-2.2-d2.pdf
- **Chaos Testing** - https://www.ibm.com/garage/method/practices/manage/practice_chaotic_testing/
- **Acceptance Testing** - https://en.wikipedia.org/wiki/Acceptance_testing#User_acceptance_testing