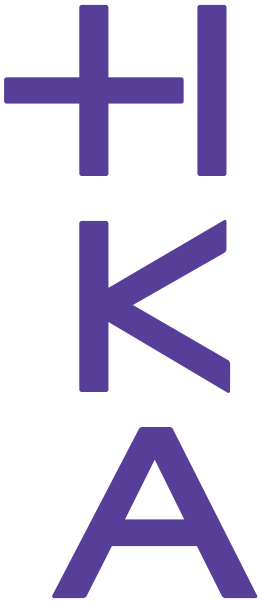
Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Informatik und
Wirtschaftsinformatik



Reservation process for charging infrastructure management in the context of e–mobility

Master Thesis of

Julian Bücher

at the Faculty of Computer Science and Business Information Systems

Reviewer:	Prof. Dr.–Ing. Zoltán Nochta
Second reviewer:	Prof. Dr. rer. nat. Heiko Körner

17. March 2023 – 16. September 2023

Hochschule Karlsruhe – University of Applied Sciences
Fakultät für Informatik und Wirtschaftsinformatik
Moltkestr. 30, 76133 Karlsruhe, Germany

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 16.09.2023

. . . . . .

(Julian Bücher)

Abstract

The continuous development of technologies related to electrified–mobility (e–mobility) is driving increased demand for the charging and transportation infrastructure. This inevitably leads to bottlenecks in the use of the corresponding charging stations (CSs). Despite the application of intelligent mechanisms for load distribution and the use of computer–aided processes for charging electric vehicles, also known as smart charging, scenarios such as the appropriate management of this type of infrastructure that facilitates reservations, receive little attention. Therefore, a system–controlled approach for the administration and distribution of charging facilities emerged as a key aspect for the further establishment of electric vehicle (EV) in society and its acceptance as a safe and reliable way of transport.
What are the functional and non–functional requirements that such a system should satisfy? This question is examined from various perspectives based on a real–world scenario and the corresponding use cases as part of this master's thesis in cooperation with SAP SE. The first part of this work begins with a theoretical examination of existing functionality covered by open industry standards, such as the Open Charge Point Protocol (OCPP) or Open Charge Point Interface (OCPI). As an open standard, many manufacturers of electric charging equipment implement these features in their products, allowing potential service providers to cover a wide range of stations, by supporting one or more of these de facto standards. Hereinafter, the thesis presents the collection and analysis of requirements, that are not part of any of the already listed standards at the time of writing. Resulting in the selection of a subset of functional modules, that is used to implement the design proposal as a proof–of–concept (PoC), demonstrating the feasibility of the proposed design.
After the theoretical part, which isolates the necessary use cases for building a comprehensive reservation system, a basic set of operations and processes covering this functionality is implemented as part of the e–mobility open source software from SAP SE, also known as *Open e–Mobility*. Besides the implementation, the reservation module's capabilities were successfully validated, by using a simulated test environment, verifying compliance in terms of the aforementioned standards. Regarding current reservation systems, the suggested solution, encompassing the web–based user interface, the mobile application and the backend service, showcases the desired functionality of allowing users to oversee both allocated charging infrastructure as well as the corresponding reservations.
Based on the underlying implementation, reservations can also be used in other scenarios in the described problem space. The use of reservations in combination with the time periods of the individual entries could be used for bi–directional charging, better known as Vehicle to Grid (V2G), in order to avoid load peaks in the power grid.

Zusammenfassung

Die sich stetig weiterentwickelnden Technologien im Kontext von Elektromobilität führen unweigerlich zu einer erhöhten Auslastung von Lade–, wie auch Verkehrsinfrastruktur. Dabei sind Engpässe in der Verwendung von Ladestationen unvermeidbar. Trotz der Nutzung intelligenter Mechanismen zur Lastverteilung und computer–gestützter Verfahren für das Laden elektrisch–betriebener Fahrzeuge, wie dem Smart Charging, finden Ansätze wie das gezielte Verwalten von Ladeinfrastruktur durch Reservierungen nur wenig Verwendung. Die daraus resultierende Notwendigkeit eines systemgestützten Ansatzes zur Verwaltung von Lademöglichkeiten ist somit ein zentraler Aspekt für die Etablierung von elektrischen Fahrzeugen innerhalb der Gesellschaft und deren Akzeptanz als verlässliches Fortbewegungsmittel.
Welche funktionalen– und nicht–funktionalen Anforderungen sollte solch ein System erfüllen? Diese Frage wird anhand konkreter Use Cases, eingebettet in ein realitätsnahes Szenario, in der hier vorliegenden Master–Thesis in Kooperation mit der SAP SE aus unterschiedlichen Perspektiven beleuchtet.
Zu Beginn stand eine theoretische Betrachtung der bereits vorhandenen Funktionalitäten, die basierend auf offenen Industriestandards, wie dem Open Charge Point Protocol (OCPP) oder dem Open Charge Point Interface (OCPI), durch die Hersteller elektrischer Ladesäulen bereits angeboten werden und durch Berücksichtigung die Interoperabilität der Lösung erlauben. Anhand dieser standardisierten Operationen, erfolgte anschließend eine Sammlung von Anforderungen, die während der Bearbeitungszeit der hier vorliegenden Arbeit noch nicht als Teil einer der oben genannten Standards berücksichtigt wurden. Jedoch im Kontext dieser Arbeit als essenziell für das zu erreichende Ziel identifiziert werden konnten. Daraus resultierte wiederum durch Selektion einer Teilmenge notwendiger Basisfunktionalität eine Umsetzung im Rahmen eines proof–of–concept (PoC), der die Umsetzbarkeit des vorgeschlagenen Ansatzes demonstriert.
Nach Durchlaufen der theoretischen und praktischen Phase konnte eine Grundmenge an Operationen und Vorgängen synthetisiert und als Teil der SAP SE Open Source Software *Open e–Mobility* beispielhaft unter Verwendung des konstruierten Szenarios umgesetzt und anhand einer simulierten Testumgebung validiert werden. In Anlehnung an bestehende Reservierungssysteme demonstriert dieser Ansatz, integriert in der webbasierten Oberfläche, der mobilen Applikaktion, sowie des Backend–Servers, eine Art und Weise, wie zukünftige Nutzer über das System zugewiesene Ladepunkte wie auch deren Reservierungen verwalten können.
Entsprechend der zugrundeliegenden Implementierung erlaubt das Konzept der Reservierung den Einsatz innerhalb weiterer Szenarien, wie beispielsweise im Kontext von Smart Charging. Hier sind in Kombination mit den Zeiträumen für die einzelnen Reservierungen Ansätze bezüglich bidirektionaler Ladevorgänge, dem Vehicle to Grid (V2G), zur Reduktion der Auslastung des Stromnetzes durchaus denkbar.

Contents

List of Figures

List of Tables

Listing

Acronyms

API Application Programming Interface

BEV battery electric vehicle

CMS Central Management System

CO_2 Carbon Dioxide

CPO Charge Point Operator

CRS Computer Reservation System

CRUD Create, Read, Update & Delete

CS charging station

CSDL Common Schema Definition Language

CSMS Charging Station Management System

CSO Charging Station Operator

CSV comma-separated values

DER Distributed Energy Resource

DN domain name

e–mobility electrified–mobility

eMIP eMobility Inter-Operation Protocol

EMSA E–Mobility System Architecture

EMSP Electric Mobility Service Provider

EV electric vehicle

EVSE electric vehicle supply equipment

EVU Electric Vehicle User

FCFS first–come–first–served

FEV full electric vehicle

GDPR General Data Protection Regulation

GUI graphical user interface

HEV hybrid electric vehicle

HKA Karlsruhe University of Applied Sciences

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IaaS Infrastructure–as–a–Service

IBE Internet Booking Engine

ICE internal combustion engine

ID identification

IP Internet Protocol

ISO International Standard Organisation

JSON JavaScript Object Notation

K8s Kubernetes

LTC Linder Technologie Campus

OCA Open Charge Alliance

OCHP Open Clearing House Protocol

OCPI Open Charge Point Interface

OCPP Open Charge Point Protocol

OData Open Data Protocol

OICP Open Intercharge Protocol

OSI Open Systems Interconnection

PaaS Platform–as–a–Service

PHEV plug-in hybrid electric vehicle

PMS Power Management System

PoC proof–of–concept

POI Point of Interest

RES Renewable Energy Source

REST Representational State Transfer

RFID Radio-Frequency Identification

RUP Rational Unified Process

SG Smart Grid

SGAM Smart Grid Architecture Model

SMTP Simple Mail Transfer Protocol

SOAP Simple Object Access Protocol

SoC state of charge

TCP Transport Control Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

URN Uniform Resource Name

V1G Smart Charging

V2B Vehicle to Building

V2G Vehicle to Grid

V2H Vehicle to Home

V2X Vehicle to Everything

WSDL Web Service Description Language

XFC extreme fast charging

1 Introduction

The declining availability of fossil fuels for the manufacturing of products such as petrol and the resulting consequences like global warming by burning these energy carriers are only two of the arguments driving the debate to promote the development of more sustainable behavior in different areas of daily life [38]. Besides switching from coal or oil as a viable source for energy generation to renewables, or the transition from cars powered by internal combustion engine (ICE) to a more environmentally friendly alternative, is an essential step in reducing the amount of Carbon Dioxide (CO_2) in the atmosphere and slow down the accelerating speed of the climate change. As part of the process of 'electrification of energy conversion systems in mobile platforms' [2, p. 165226] in the context of electrified–mobility (e–mobility), the industry and users still have some challenges to deal with. The most significant issues are the inadequacies in key technologies such as batteries or power trains. Compared to their fossil fuel alternatives, they are less efficient and durable, resulting in limited operating ranges due to their lower capacity. In combination with the current shortcomings, the lack of charging facilities in sparsely populated areas or cities is another unaddressed problem that adversely affects the acceptance of EVs. Other negative consequences, such as the phenomenon of 'range anxiety' [64], experienced by many Electric Vehicle Users (EVUs) in rural areas lacking a widespread charging infrastructure, additionally affect the acceptance and conversion of cars with ICE to hybrid electric vehicle (HEV) or full electric vehicle (FEV). Furthermore, the waste generated by the production or disposal of electric vehicles is a crucial factor. In particular, the disposal of lithium–ion batteries has a harmful impact on the surrounding environment [70], which must be taken into account when further developing this technology. Nevertheless, the long–term benefits of EVs and their impact on reducing CO_2 and the use of fossil fuels are undeniable. Because of this, numerous institutions and governments are eager to endorse the growth of electric mobility and try to counteract the negative impacts mentioned above. For example, government subsidies, as well as other incentives offered by manufacturers, are measures intended to promote EVs as a viable means of transport in the future.

1.1 Conceptual Formulation

Due to the requirement for a system to administer and monitor the associated infrastructure, different approaches for managing CSs and the corresponding connectors have emerged. As a result, multiple software companies partnered with charging station manufacturers or standardization bodies to establish a common denominator for information exchange. Their work led to the creation of standards such as the Open Charge Point Protocol (OCPP), Open Charge Point Interface (OCPI) and ISO 15118, which offer a comprehensive range of features for communication between the EVUs, EVs, and their respective CSs. Despite the extensive range of processes covered, the mentioned standards are not considered complete and require further refinement and enhancements for special situations. Therefore, almost every year, new major versions with minor fixes and improvements in the form of new features are released. Considering certain feature gaps, particularly management or administrative processes to ensure fair and regulated use of charging options, functionalities such as the aforementioned are often missing in current implementations. Addressing this need, this document describes a process for reserving charging points or connectors in advance for a specified period of time. Besides providing the EVUs with a guarantee for charging, this feature should facilitate a detailed administration of CSs for owners of semi–public or private parking spaces and offer an alternative to the current first–come–first–served (FCFS) principle in a more regulated way.

1.2 SAP SE

This work is part of a master's thesis at SAP SE, a multinational software company based in Walldorf, Germany. It is one of the world's leading enterprise software companies, renowned for its innovative solutions that help businesses manage their operations effectively. Founded in 1972 by Dietmar Hopp, Hasso Plattner, Claus Wellenreuther, Hans–Werner Hector, and Klaus Tschira [21], the company aims to develop standard application software for real–time business data processing and to improve the way businesses manage their operations. Apart from *SAP S/4HANA* and *SAP ERP* [66], SAP offers a wide range of enterprise software products and services to various business needs and industries, such as manufacturing, finance, healthcare, retail or public sector organizations. Furthermore, SAP is engaged in developing emerging technologies such as artificial intelligence, machine learning, the Internet of Things (IoT), or solutions in the e–mobility sector. This includes software for managing existing CSs within one company or applications to support the EVUs in finding free charging opportunities more easily. As part of this approach to drive the expansion of e–mobility in the enterprise world, this thesis provides additional concepts and example workflows for the advanced management of charging infrastructure.

1.3 Document Structure

As the main focus of this work, the following chapters describe a potential solution for conceptualization and implementation of a reservation system based on the existing SAP open source solution, *Open e–Mobility*. Starting with the gathering of essential contextual information, the subsequent conceptual section identifies the necessary use cases for the design of the corresponding processes. Followed by a structured timeline describing the approaches for both the theoretical part and the implementation part, broken down into steps. Next, an assessment of previous research investigating comparable methods for addressing the primary objective mentioned in Section 1.1 is performed. This leads to the design part, which takes the use cases collected in Chapter 3 and illustrates them in the form of flowcharts that form the basis of the implementation. On the basis of the outcomes of the previous analysis and design phase, a description of the translation of the selected processes for implementation in software components within the application follows. This document provides a comprehensive description of the entities, actors, and related systems involved. Finally, the results are evaluated and validated according to the scenario created, concluding this paper with a summary in combination with impulses for further development and integration of other aspects mentioned in this work.

2 Fundamentals

The subsequent chapter provides a fundamental understanding of the concepts and general conditions, that apply in the sector of e–mobility, and therefore are omnipresent in most of the described scenarios later on. First of all, Section 2.1 gives a general review of e–mobility, like the involved actors and their systems. Furthermore, existing standards for depicting the communication between these actors and systems and a brief introduction to smart charging and its variations follow. Additionally, the subsequent Section 2.2 illustrates the basic functionality of a reservation system itself, its use cases, and a differentiation between common types of reservation systems. Lastly, the Section 2.3 lists important communication protocols and data exchange formats used in the respective standards described in 2.1.4.

2.1 Electric Mobility

The term e–mobility generally refers to the use of electric vehicle (EV) and other electric–powered transportation options as an alternative to conventional vehicles that are powered by an internal combustion engine (ICE). As a crucial element of the worldwide initiative to counter climate change and achieve sustainable transport solutions, it aims to reduce greenhouse gas emissions, lower dependence on fossil fuels and mitigate the environmental impact of transportation [38]. To achieve this objective, e–mobility depends on the expansion of EVs as a common means of transport and a denser cluster of CSs. Apart from the focus on vehicles primarily addressing personal transportation, this term could also be applied in the context of freight transport, which demands much higher capacities in terms of the power required by such vehicles. Within this research and the following chapters, only BEVs used for personal transportation in the form of cars or smaller vans are considered as the respective audience.

2.1.1 Electric Vehicles

As already mentioned in Section 2.1, EVs are a fundamental cornerstone of e–mobility. Primarily, an EV refers to a car that is powered by only one or more electric engines drawing energy from onboard batteries or similar energy sources. Besides battery electric vehicles (BEVs), which are purely powered by electricity, further invariants like plug-in hybrid electric vehicles (PHEVs) or hybrid electric vehicles (HEVs) exist. In comparison to the aforementioned BEVs, PHEVs combine the power of an electric motor with the reliance offered by an internal combustion engine. This allows the driver to cover shorter distances solely relying on the internal battery, and use the petrol engine during longer trips or when charging is unavailable. Moreover, PHEVs have the capability of recharging

the battery while on the move. Therefore, brake resistance is employed to convert the power generated into electricity, feeding it directly back into the battery. HEVs closely resemble PHEVs as they have access to both power sources. However, unlike a PHEV, HEVs do not have the possibility to be plugged in for charging [38].

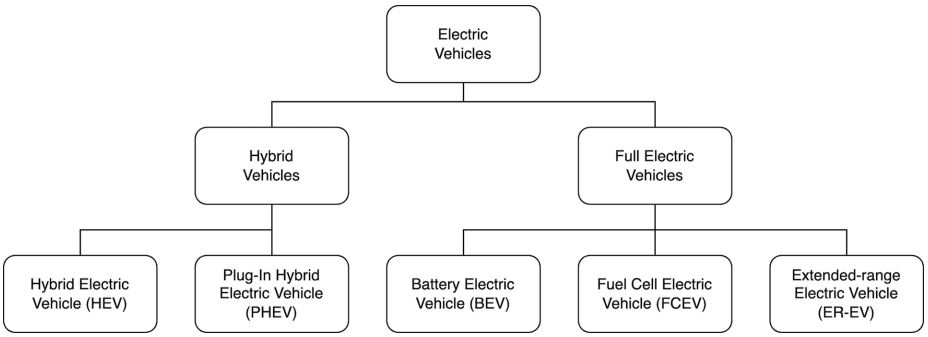


Figure 2.1: Classification approach for different EV types based on [1].

2.1.2 Charging Infrastructure

To facilitate the growing number of EVs, a crucial aspect is the establishment of a broad charging infrastructure comprising multiple CSs. As highlighted in [30], the requirements for charging infrastructure are highly variable, depending on the battery capacity and charging rate, which is predicted to increase in the future. In order to satisfy the various power requirements and to cover the wide range of vehicles and used batteries, a heterogeneous range of CSs exists. From stations providing so–called slow charging with a power supply up to 22 kW, more advanced technologies allow fast charging CSs for power requirements above this certain range.
For a better understanding of the different scenarios in which a CS could be used for charging, this work divides the infrastructure and the corresponding stations into three different groups. These groups, classified according to accessibility level for the EVUs, are presented in Table 2.1 alongside their user restrictions, the associated locations, and most commonly used charger levels:

Table 2.1: Differentiation of accessibility in case of charging opportunities based on [38],[44, pp. 18–19].

Accessibility	Restrictions	Location	Charging Level
Public	No	fleets, highway, distribution centers	3
Semi–Public	Yes	workplace, hotels	2
Private	Yes	private households	1

Alongside the separation according to the accessibility level, a differentiation based on classes of charging stations (CSs) is possible. For a detailed illustration of the dependencies and relationships between the different classes, see Figure 2.2 below.

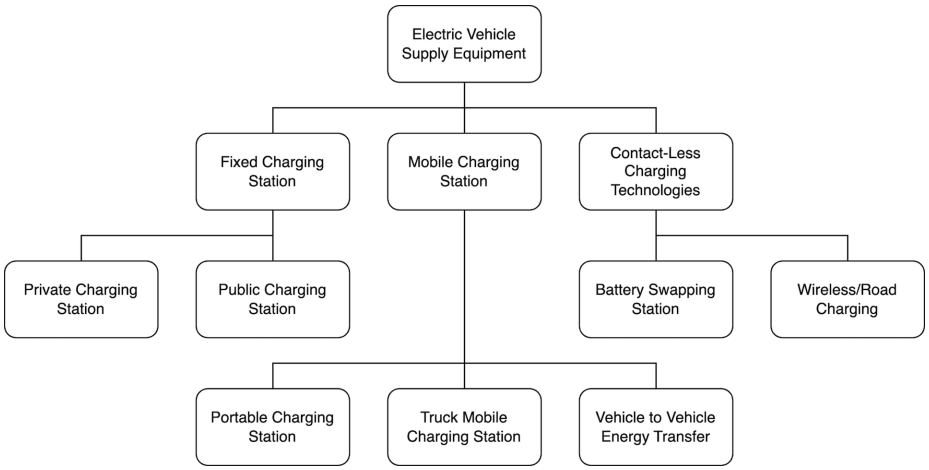


Figure 2.2: Classification for different types of CSs based on [3].

To provide a more detailed view on the infrastructure and the hardware used for charging, also referred to as electric vehicle supply equipment (EVSE), all components and systems necessary to supply electric power from the grid to the battery of an electric vehicle (EV) are explained afterwards. Beginning with the **charging station (CS)** itself, as physical power unit the EV is connected to. As mentioned above, CSs can vary in size and complexity, ranging from small wall–mounted units designed for home use to more extensive public CSs found in parking lots, shopping centers or along highways. To establish a connection between grid, CS and car, a **Connector or Charging Cable** is needed. It contains plugs on both ends, which are connecting to the vehicle's charging port and to the CSs outlet. As an integral part of the EVSE, the **PMS** regulates the flow of electricity from the grid to the vehicle's battery. It ensures safe and efficient charging, managing the power level based on the vehicle's battery capacity and the grid's capacity. Allowing the communication between the EV, the charging station (CS), and the grid, a **CMS** is used as a backend. It allows data exchange regarding charging status, electricity prices, user authentication, and other relevant information. In the case of public CSs, a payment and authentication system is usually integrated into the EVSE. This system verifies the user's identity, authorizes the charging session, and processes payment for the electricity consumed. On top of the described components, various safety features are integrated to protect users, the vehicle, and the surrounding environment. Including features such as ground fault protection, over–current protection, temperature monitoring, and emergency shut–off capabilities [45]. To meet the wide range of existing EVs and their individual power requirements, different categories of chargers have been established based on the power level provided. This work lists the most notable categories, along with their specifications and typical usage, in Table 2.2.

Table 2.2: Listing of different EV charging levels based on [1].

Charging Level	Charging Power	Use Case
1	1.44 kW – 1.9 kW	Typically the slowest charging option and suitable for overnight charging at home
2	3.1 kW – 19.2 kW	Provides faster charging compared to Level 1 and is commonly used for home charging setups and public charging stations
3	20 kW – 350 kW	Fast charging operates at a higher voltage, directly charging the vehicle's battery with direct current power. Commonly used in public fast–charging stations.
XFC	>350 kW	Ultra–fast charging provides fastest charging option. Commonly used in commercial settings.

2.1.3 Battery Technology

For storing the required energy, EVs rely on their built–in batteries. As key components inside an EV, they have to handle high energy capacity and high power, within limited weight and space to affordable prices. Therefore, most manufacturers use Lithium–ion batteries in their cars nowadays. Alternatives such as *Lead Acid* and *Nickel–based* batteries usually have lower durability compared to lithium–based batteries due to their shorter lifespan and subsequent inadequate performance in extreme weather or higher discharge rates [1]. Generally, the advances in this technology leading to an increase in the range of the EVs and a reduction in the charging time, combined with cost effectiveness, are an essential part of the transition from cars using ICE to purely EV.

2.1.4 Relevant Standards

To establish a comprehensive interface for information exchange between CSs, EVs, and their users, various organizations, initiatives and the industry have established several standards for communication between the single actors in this particular environment. Aside from the Open Charge Point Protocol (OCPP) [54], a communication protocol, developed by the Open Charge Alliance (OCA) [55], other protocols and specifications exist, covering different aspects and scenarios relevant during or before the charging process. In addition to the previously mentioned OCPP, further standards relevant to this work are listed and explained below.

2.1.4.1 Open Charge Point Protocol

The Open Charge Point Protocol (OCPP) outlines an open industry–standard communication protocol, used especially in charging infrastructure for EVs. Proposed by the ElaadNL foundation as an open protocol to support the communication between CS and the related backend services, the ownership was transferred to OCA in 2014 [25]. In general, the OCPP is designed to provide interoperability and seamless integration between different CS vendors and network operators, ensuring that EV drivers can charge their vehicles at any compatible CS. Therefore, it is maintained by the Open Charge Alliance (OCA), a consortium of EV charging infrastructure stakeholders, as an open protocol and not proprietary to any specific manufacturer or organization. In this way, it is ensured that the OCPP remains a collaborative and evolving standard [54].
From an architectural point of view, the protocol could be described in the form of a client/server architecture. The EVSE in this model acts as the client, while the CSMS serves as the server. The server responds to the client's requests and manages the charging processes accordingly. This request–response model, where the client sends requests to the server, and the server responds with the appropriate information or action enables real–time communication between the charging station (CS) and the connected Central Management System (CMS). As an interface for communication, two different types of protocols are used. On the one side, the WebSocket protocol, described in 2.3.4, for bidirectional communication, providing a persistent connection between the EVSE and the CMS allowing a faster and more efficient data exchange. On the other side, Simple Object Access Protocol (SOAP), as described in 2.3.2, is suitable for the implementation of one–way communication between the respective entities. To differentiate between particular feature sets, the OCPP is versioned. The versions are presented as floating point numbers, which increase with major or minor releases within the nomenclature, such as OCPP 1.6 and OCPP 2.0. Representing the latest versions of the protocol, which are widely used in current implementations.

For the communication between CS, CSMS and the EVU, the OCPP protocol relies on so called operations. Each of these operations describes a set of instructions, which are necessary to fulfill and successfully complete the underlying process. According to the required operations used in the context of this work, a subset of operations are selected, which are elucidated in the following list. The wording and the explanations are based on the standard documentation for the OCPP version 1.6 [54]:

Authorize Before starting a charging session, the EVU Radio-Frequency Identification (RFID) card or other identification is sent to the central system for authorization. The CMS checks the driver's credentials and responds with an authorization status.

Start Transaction After successful authorization, the CMS sends a start transaction request to initiate the charging process. The CS acknowledges the request, and the charging session begins.

Status Notification The CS may send status notifications to the CMS to update the current state of the CS, such as `Available`, `Charging`, `Reserved`, or `Faulted`.

ReserveNow In case a user needs an available connector on a CS, he could send a *ReserveNow* request via the CMS to the CS, which reserves one specific or at least one connector on the station for a specified duration. The according connector in case of successful reservation changes from status `Available` to `Reserved`. As a result, only the user assigned to the deposited RFID card is able to start charging at the specified connector or the superior CS.

Cancel Reservation For canceling the created reservation the user has the ability to manually send a request with the assigned reservation ID to the CMS, to free the connector again. Otherwise, the CS will notify the CMS, if the reservation reaches its expiry limit.

2.1.4.2 Open Charge Point Interface

Open Charge Point Interface (OCPI) is a standardized, open protocol facilitating communication and supporting interoperability between different charging network operators, which provides EV drivers with access to charging infrastructure from multiple providers. This is achieved by utilizing a harmonized approach. Alongside OCPI, other proposals regarding the development of cross–border EV travel, also known as *e–roaming*, exist. The most prominent ones are Open Intercharge Protocol (OICP), Open Clearing House Protocol (OCHP) and eMobility Inter-Operation Protocol (eMIP) [18]. In contrast to the OCPI, they are all developed by proprietary institutions and integrated inside their offerings providing dedicated roaming platforms. Thus, no way is provided to allow compatibility, making the OCPI standard more suitable in the case of openness and interoperability.
In case of available features, OCPI provides defined endpoints, used for communication between CS, CMSs, EMSPs and CPOs. These endpoints include functionalities such as location discovery, charge point data, authorization, charging sessions, and error handling. For integrating OCPI into existing software systems it is heavily based on the paradigm of Representational State Transfer (REST). This allows communication via standard HTTP methods utilizing JavaScript Object Notation (JSON) as a data format for transmitting information.
The listing below provides a short overview of available features and functionalities provided by OCPI [56]:

Location Discovery Charging networks can exchange information about available charging locations, providing details such as location coordinates, CS types, and status.

Charge Point Data Access to data on CS, including details on availability, status, and pricing, is possible through OCPI.

Charging Sessions OCPI supports real–time information about ongoing charging sessions, including start time, energy consumed, and current charging status.

Authorization and Authentication In combination with their respective CPO accounts or other authentication methods EVUs are able to authenticate themselves to access charging services.

Remote Start/Stop Charging Sessions To initiate charging sessions, EVUs are capable of remotely starting and stopping their current charging sessions. This allows the driver to initiate charging sessions via their mobile apps or other remote means.

Tariff Information Charging networks can share pricing and tariff information, providing transparency to EVUs about the cost of charging at different locations.

2.1.4.3 ISO 15118

To meet the demands of EVUs and address the problems of lack of standardization in charging infrastructure interoperability and information exchange between grid EVSE, EV and EVU, Daimler and RWE started a collaboration in September 2009 to enable Smart Charge Communication. Until its initial release as the standard for Plug–n–Charge connections in June 2014, further progress towards the standardization of charging infrastructure communication was carried out [32]. Nowadays, this standard consists of multiple parts. From the analysis of the underlying use cases to a description of requirements for the communication over the different OSI layers, it describes POI–data, the CS availability, payment, communication standards as well as the *Plug & Charge* or *e–Roaming* approach [10]. However, the required level of data granularity or quality is not specified [44]. Considering these gaps that still exist in the ISO 15118, it is far from being complete and requires further development and adaptation.

2.1.5 Smart Charging

The concept of smart charging or intelligent charging is a systematic approach that optimizes the charging process of EVs to be more efficient, cost–effective, and environmentally friendly using information and communication technologies. Therefore, the subsequent calculations consider factors such as electricity demand, grid capacity, availability of renewable energy, and individual user preferences to allow for optimizations in the user's decision–making [14]. One of the primary goals of this methodology is to balance the demand on the underlying grid, affected by charging large numbers of EVs simultaneously [13]. Especially during peak hours increased electricity costs or potential blackouts could occur. In order to mitigate these situations, smart charging takes the current load of the grid into account and adjusts the possible charging rates based on the given constraints to prevent overloading the grid and optimizing the use of available energy [24]. Compared to the unidirectional charging method originally known as Smart Charging (V1G), the approach of Vehicle to Grid (V2G) enables smart charging systems to charge bidirectionally between the EVs battery and the grid [38, p. 199]. Allowing EVs with sufficient battery capacity to return energy to the grid during times of high demand and turning them into mobile energy storage units to enhance grid stability and resiliency. To offer all these functions, smart charging relies on data connectivity and communication between the single CSs, the EVs, and the underlying grid, which makes it vulnerable to malign third–party entities as well. On the other side, these factors can contribute to better grid management overall, giving grid operators insight into electricity demand patterns and enabling them to plan grid upgrades accordingly.

2.1.6 Smart Grid

More generally, the terminology of a Smart Grid (SG) describes the characteristics of an intelligent system that deals with high energy consumption in order to increase energy reliability and the corresponding costs [49, 68]. It consists of multiple parts, like a sufficient infrastructure, applications, and several technologies to manage the energy flow inside the grid. This includes, for example, monitoring and measuring solutions as important cornerstones, allowing the grid to support an efficient generation and distribution of power. Supplemented by the behavior of the users connected to the SG, which is a component of the continuously operating optimization procedures. Based on [49], researchers have identified 94 potential optimization algorithms thus far, primarily implemented in central orchestration systems and observing the SG to manage these kinds of distributed networks. As a result, the smart grid is capable of dynamically and intelligently controlling the flow of power based on predetermined objectives to meet the needs of all nodes in the network. Considering the delivery of electricity, compared to existing grids that deliver power in one direction only, smart grids enable bi–directional power delivery. Therefore, the smart grid utilizes DER, such as EVs or renewable energy sources, located at consumer homes, to absorb surplus power when not in use. This approach is very similar to the V2G functionality of smart charging described in 2.1.5 and could be combined with it.

2.2 Reservation Systems

Originally, current reservation systems were developed as Computer Reservation System (CRS) to facilitate collaboration between airlines and technology companies [31]. The original objective of the system to be a tool to book resources without the need for paper or other management systems has obviously evolved over time. This also applies to the features and abilities of these systems. Started as a centrally managed system with restricted access to call center operators, which administered the available resources for their corresponding airlines or hotel chains, the further development of the world wide web in 1989 led to a transformation away from the CRS to online booking systems, also known as web–based Internet Booking Engines (IBEs), directly available to the customer via their personal computers at home. In general, a reservation system could be defined as a software application or platform that facilitates the process of booking and securing limited services, resources, or accommodations in advance, based on the definition of Xiang et al. [31, p. 2]. As mentioned previously, besides its original application in the travel or hospitality industry, it is also a common tool for the administration of the transportation or entertainment industry. With the goal of simplifying reservation procedures for customers and minimizing administrative tasks, while increasing user satisfaction. Based on the system architecture and the technologies employed, reservation systems could vary depending on the functionalities required. When utilized in a web–based application, such as an online booking system, it is essential to consider different constraints and use cases, compared to a local system within one institution. To abstract from the requirements necessary for particular deployment scenarios, the following section outlines a set of fundamental characteristics and functionalities that a reservation system

should typically encompass. While the chosen list of features may not include all the use cases required for a specific system, it should furnish a brief overview for general purposes. For reference and guidance in the case of the functionality of the reservation system, the existing research and literature on hotel reservation systems such as [9, 15] as well as the common functionalities found in current booking systems are used. **Booking and Scheduling** could be described as the main purpose of a reservation system. By using this feature users can view available dates, times, or slots and make reservations for specific services, activities, or resources. The system ensures that conflicting bookings do not occur. To provide an overview, which represents the availability of the aforementioned resources, the system requires **Real–Time Availability**. Therefore, users can see immediately if the desired service or resource is available for the chosen date and time. For the purpose of personalizing the user experience, retaining reservation history, and enabling loyalty rewards, the common reservation application provides a **User Registration and Authentication** feature to allow the user to create an account or log into restricted areas of the system to book a reservation. Additionally, managing and monitoring bookings, the option to securely pay for allocated properties using a credit card or digital wallet via **Online Payments** often is part of the service offering. To acknowledge successful reservations or notify users of any changes to their bookings, a **Confirmation and Notifications** should be embedded in the application. This feature not only acts as a confirmation of the booking, but also evidence of the completed transaction and may contain information such as booking ID, date, time, and location. Regarding the need for modification or cancellation of the reservation, the system should provide **Cancellation and Modification** features, enabling the user to cancel or modify their reservations within a specified time frame. Additionally, cancellation policies eventually exist for penalties punishing late cancellations. Moreover, apart from the management of the reservations made through the system, it should provide a **Inventory Management** for the owners of the resources and help them manage their possessions. Other features such as **Reporting and Analytics** or the **Integration with Other Systems**, which allow the generation of reports on booking trends, revenue, and occupancy rates, are advanced functionalities typically not visible to the end–user. Mostly offered as an administrative service extending the basic use of the system, reporting and third–party integration of other systems, such as customer relationship management (CRM), is not part of every software suite. This also applies to the integration of payment gateways.

2.3 Data Exchange

After the presentation of the various services, their corresponding communication channels, and the information provided within Section 2.1.4, the subsequent section deals in more detail with the technical concepts that underlie both the protocols and the application. In addition to a brief overview of Representational State Transfer (REST) in 2.3.1, a quick summary of the technologies Simple Object Access Protocol (SOAP) in 2.3.2, Open Data Protocol (OData) in 2.3.3 and WebSockets in 2.3.4 follows.

2.3.1 REST

Introduced by Roy Fielding as an architectural style [60] to design distributed systems, Representational State Transfer (REST) provides concepts and a set of constraints that define resources accessed via a common interface by HTTP methods specified in the standard protocol description. Stated as the general set of HTTP methods used to query a REST Application Programming Interfaces (APIs), the methods `GET`, `POST`, `PUT` and `DELETE` are the most conventional examples in the context of web development. Representing a set of standardized operations, also known as CRUD operations, similar to the functionalities utilized in the context of relational and non–relational databases, these methods assume the activation of certain functionalities within the respective targets. The verbs used to name the operations carry significant meaning, indicating the associated functionality and the expected outcome of the call. Aside from the mere retrieval of data, which is usually performed by the use of `GET`, the creation and update of an entity involves either `POST` or `PUT`. Intentionally, while awaiting a specific result from performing a read operation, creation methods may or may not return the resulting object. Both possibilities align with the standard and result in subtly different behaviors due to varying design decisions, such as prioritizing reducing network bandwidth versus usability. To determine the appropriate targets for the client, REST defines the concept resources, which allows a resource identifier to be mapped in the form of a URI or a URN. Following this pattern, a legitimate resource representing a car is presumably indicated by the `http://rest.service.com/cars`. Sending a HTTP request with the `GET` method to this resource should result in a list of cars provided by the service. As a valid format for representing data, both JSON and HTML documents are permissible. Even images could be utilized. Further data elements defined by REST are resource metadata, representation metadata, and control data, which are only mentioned here for completeness. In addition to these core components, REST provides numerous options for application and system design. However, they are not relevant to this research and are therefore not described in detail here.

2.3.2 SOAP

As a counterpart to the REST protocol described in the previous Section 2.3.1, SOAP was created by Dave Winer and others in collaboration with Microsoft [60] to addresses the demands of enterprise software. The fundamental idea of SOAP is to provide data as services, tagged with names consisting of both verbs and nouns. For instance, the service named *getCar* is exclusively for providing information about cars and is not intended for any other objectives. In contrast to REST, this ensures better communication of the purpose for which the method is intended, as well as the necessary data that this particular function relies on to fulfill its purpose. All of this information about the handled objects and parameters is communicated to the client prior to the actual interaction and described as part of a Web Service Description Language (WSDL) document, which is publicly available from the URL of the particular services implementing this protocol. This creates a so–called contract between a server and a client, providing security and authorization as well as more direct access to objects available on the server. As a result, these components are much more tightly coupled. Therefore, SOAP, as outlined in [60, p. 4], is typically

recommended for scenarios involving use cases considering transactional operations or in case the environment is dependent on this protocol.

2.3.3 OData

Another option for implementing web–based services besides utilizing only the REST architectural style is the OData protocol, which aims to build on top of the latter to provide both a higher level of consistency in terms of breaking changes and to increase interoperability between the communication partners [17]. To accomplish these goals, OData services, similar to services implementing SOAP, supply a metadata definition using Common Schema Definition Language (CSDL) as a metadata description language ensuring a machine–readable form of the service functionalities it provides. In the case of generic clients with no knowledge of the implemented logic of the services, these documents exchanged as part of a preflight request between server and requestor, give the clients the ability to interact in a formalized way. Alongside these outstanding features, the OData standard offers additional concepts such as *namespaces* or specific URL conventions to cover much more advanced scenarios. Nevertheless, these aspects are beyond the scope of this informative section and are not covered in this brief introduction.

2.3.4 WebSocket

Based on the RFC 6455 standard [47], the WebSocket protocol facilitates two–way communication between a client and a remote host. Formerly, clients on the web who needed to establish bidirectional communication with their server had to utilize HTTP calls to request updates from their correspondent and send notifications in response. Resulting in numerous obstacles for the server and its client, including the quantity of open Transport Control Protocol (TCP) connections and the overhead generated by the bloating headers appended to each request. Trying to solve these problems, the WebSocket protocol suggests using only one TCP connection for traffic in both directions. In order to create this information channel, Fette and Melnikov's proposal allows taking advantage of existing infrastructure, which lets the users replace existing technologies using HTTP as the transport layer within their applications. To initiate the data exchange process, this communication approach relies on establishing trust through exchanging preliminary handshakes and utilizing the origin–based security model, which is commonly implemented in current browser applications. Once the trust between server and client is established, the protocol allows the applications to communicate utilizing basic message framing layered over TCP stack. This grants web–based services the ability to construct efficient information exchange in two directions, without the need to open multiple HTTP connections. Apart from the possibility of using the WebSocket protocol over unencrypted HTTP channels, it can be utilized in encrypted traffic via Hypertext Transfer Protocol Secure (HTTPS) on port 443 as well.

3 Requirements Engineering

The main focus of this chapter is to describe the requirements that a reservation system for charging infrastructure needs to fulfill. For that purpose, this work uses a fictional scenario to paraphrase and describe the underlying problem. Furthermore, a comprehensive representation of significant edge cases the system design has to consider is depicted as well as an accentuation of highlight potential problems is included. Adapted from the requirements, use cases and goals are defined, giving an overview of the central tasks of this thesis.

3.1 Scenario

Beginning with the capture of fundamental points necessary for outlining the scenario, which represents the problem space the system is intended to operate. The scenario includes, besides the location where the subsequent setting takes place, a group of personas describing fictional people representing actors, who interact with the system itself. Each of these actors has different demands due to their current life situations regarding the usage of the system, which results in different expectations with respect to the fulfillment of their goals, based on the interaction with the system. Formalizing these expectations and transforming them in a more general way, they could be derived as use cases separated by system actors for the implementation afterward.
To cover a broad range of possible users according to age, social backgrounds, and the ease of handling obstacles generated by new technologies, the scenario takes place in the context of the institution Karlsruhe University of Applied Sciences (HKA) located in the city of Karlsruhe. As a partner of the *Green4Ever* project [16], the HKA covers several aspects, which could be used for designing a sophisticated scenario. Aside from a broad range of users, ranging from students to the regular staff, it consists of multiple campus sites distributed all over the city. This allows members of the institution to travel between the single sites and generate persistent demand for charging possibilities during the whole day. Relevant for the description of the single actors, this scenario includes the sites **main campus**, **campus Amalienstraße**, also known as *field office Amalienstraße*, and the **Linder Technologie Campus (LTC)** with the amount of applicable CSs, which are listed in Table 3.1 below.

Table 3.1: Campus sites and parking lots (site areas) in the HKA organization with the according number of available CSs.

Site	Site Area	Number of CSs
Linder Technologie Campus	Private Parking Lot LTC	1
Linder Technologie Campus	Parking Lot LTC	3
Field Office Amalienstraße	Parking Lot Amalienstraße	3
Main Campus	Parking Lot Steinbeishaus	3

Furthermore, this work assumes the preferred way of movement to the sites is by public transportation or with their own cars. As mentioned before, each of these campuses has its own parking lots equipped with a limited amount of CSs for students and the staff working at this site. Corresponding to the accessibility level declared in 2.1, the parking possibilities include public and semi–public charging opportunities. Public parking lots and the corresponding CSs are available for students and staff members as well. In the case of semi–public parking lots, the access is only limited to visitors or staff exclusively. Concerning the possibilities to configure CSs, all public stations are enabled for reservations. To prevent unnecessary complexity, this scenario does not include CSs without support for the reservation feature. For administration and management of these facilities and the restricted charging resources, a software solution called *Open e–Mobility* [26, 27, 29] is used, which includes a mobile application for the EVUs and a web–based application for browser access, mostly used by administrative departments of the HKA. On behalf of these applications, the users have the possibility, according to their roles in the system listed in Table 3.2, to start or stop current charging sessions remotely or check for occupied CSs on the available sites and site areas. Using privileged access via an administrative account, the user is able to deactivate or block connectors of specific CSs, to make them inaccessible until reactivation or a rebooting process. By now, the administrator or the common user does not have the opportunity to reserve an available CS connector in advance. Rather for themselves or on behalf of a specific user. This includes reservations for visitors arriving later or during times with high loads as well. As an implication of this missing feature, a first–come–first–served (FCFS) mentality among the EVUs was established, which led to decreased usage of EVs as a transportation means and an increasingly competitive behavior among the drivers.

3.2 Stakeholders

According to the scenario elaborated in Section 3.1 above, the following groups of stakeholders could be identified. Considering the subsequently defined staff, it is possible to further differentiate into two subclusters for additional clarification. The description below provides an overview of the daily challenges members of these classes of people have to face in terms of the established implications defined by the scenario and their main objectives.

Students Representing the largest category of actors within the scenario and describing people studying at the HKA. Primarily, students spent most of their time on the **main campus** site, because the main part of lectures offered by the different faculties and the attendant exercises are located there. In certain cases, some lecturers offer subjects at other campus positions, where doctoral research projects or research in general is outsourced. This involves several students each semester, commuting between the sites and generating additional demand on the existing charging infrastructure. Due to the fact, that the number of students with EVs outnumbers the capacity of CSs at the different sites of the HKA, a student is typically concerned about the occupation rate of the public available CSs. In consideration of planning their charging sessions, the students want an overview of available CSs at their campus site and want to reserve a public CS, if they are available for reservation. Therefore, they mostly rely on the mobile application offered by the institution, which serves as the main entry point for them to interact with the CSMS.

Staff As the primary workforce of the HKA, the staff describes a group of people working at the institution, including roles like lecturers, cleaners, or librarians for example. Based on their office location, they are typically assigned to one specific or several campus sites. Implicating the dedicated usage of the CSs for charging their EV, available at their current working place. In contrast to students, they have permission to access the CSs on the semi–public parking lots as well as the ones located in the public parking spaces. Due to the new home office regulations, staff members have the choice to work from home several days a week, which may relax the situation in regard to the high demand for charging infrastructure. However, on office days staff members prefer to know the occupation rate of the CSs in the public or semi–public spaces and reserve a spot for a guaranteed charging possibility for their EV. Referring to the fact, that they spent most part of their daily working hours on the computer rather than their mobile, staff members utilize the web application as a primary way of managing their charging sessions. The mobile application, the majority of the users rely on, is only used in certain cases.

Maintenance Personal As a dedicated subset of the regular staff working at the HKA, the maintenance personnel consists of caretakers in the first place. Their regular occupation consists of responsibility for the maintenance of the physical infrastructure as well as service offerings respecting the charging stations located on the different campuses. For work–related trips between the single sites, the maintenance team makes use of a FEV with a dedicated CS exclusively accessible by this vehicle and the respective staff. To monitor the connected charging infrastructure and identify outtakes due to technical errors, the caretakers interact with the system using the administration dashboard, via a personal workstation in their office. In cases where a computer is not accessible, they utilize the mobile application to check the status of the charge points. Besides the functionality as a supervisory system for the infrastructure, the caretakers need access to the reservations made on the CSs to resolve conflicts related to system outtakes or to manage the corresponding users.

Administration Representing the administrative counterpart to the maintenance personnel, the group of secretaries working at the administration office manages internal data relating to staff and students belonging to the HKA, internal workflows, or communication with other organizations. This includes paperwork in relation to certain approvals for new students, maintaining the stored records, and processing incoming requests. Among other things, they are responsible for organizing events on and around the campus sites, planning upcoming visitations from external facilities, and coordinating the utilization and maintenance tasks for the charging infrastructure. As a supervisory unit for the **Maintenance Personal**, they are scheduling maintenance and delegating tasks to the caretakers in case of extraordinary servicing. Therefore, staff members attending these positions need access to the CSMS similar to the scope of the maintenance staff. In contrast to the caretakers, they are not leaving their office for interaction with the CSs, therefore they do not rely on the functionalities the mobile application has to offer.

3.3 Personas

Appropriate to the defined stakeholders in the previous Section 3.2, each stakeholder requires a representation as a persona in regards to a dedicated user account in front of the system. This allows a mapping between the accounts and the existing system roles, which enables easier separation between the use cases extracted later on and the assignment to the different stakeholders. Hereinafter, personas prescribing individuals inside the scenario including their dedicated demands in respect of the challenges their particular stakeholder group has to face.

Lisa Knaus is a **Student** at the HKA and is studying computer science in the third semester. In respect of her current living situation, located in a village with limited access to public transportation, she primarily relies on driving by car as her main transportation means. Fostered through financial benefits promoted by the different car manufacturers for young drivers with interest in acquiring an EV, she sold her old car with ICE and switched to a full electric vehicle (FEV). Under normal circumstances the health of the internal battery allows her to drive with a full charge from her hometown to the main campus site and back without a need to recharge. Occasionally she forgot to recharge at home and needed a charging possibility for her vehicle during the lecture time. Depending on the day and the utilization of the charging infrastructure, this is certainly a challenging task and she needs to arrive very early to get a free charging spot.

Holger Starke is an employee working as a member of the **Staff** at HKA. Within the in–house server administration team, his daily tasks cover the maintenance of the server landscape, applying new versions of internal applications, and the creation of backups, securing the records of the internal user databases. Thanks to the subsidies for electric vehicles from the state and his employer, he has bought a FEV and is using it as his main means of transport. As compensation for the still missing charging possibilities at his apartment near the institution, he drives a short distance with his car every day, to use the existing charging infrastructure at the HKA, to recharge the batteries during working hours. Therefore, he has to use the available CSs at the different campus locations without any guarantee to recharge. With the increasing amount of students and coworkers arriving with EVs over time, the search for a free CS becomes more and more demanding. Especially scheduled service work later in the day, located at other campuses, emerge to a running the gauntlet for parking lots with CSs equipped.

Dieter Krause is engaged in the role of a caretaker as part of the **Maintenance Personal** at HKA. During his previous position, he acted as a carrier for the daily book orders for the students at the university library on the main campus site. For the reduction of CO_2 the HKA bought a FEV for bridging the short distances between the various libraries within the city. This confronted him with the according up- and downside of driving an EV in a very early stage. Missing charging opportunities and insufficient battery life were only two obstacles he had to bother with, every day. This led him to the conclusion to switch positions and he took over the responsibility for the charging infrastructure maintenance and management on the campus sites to improve the situation for all involved parties. Alongside the continuously increasing number of CSs, he has fostered the establishment of dedicated charging possibilities for the members of the **Staff** and introduced an exclusive CS for the EVs of the **Maintenance Personal**. Even on days with a high emergence of student vehicles on the public parking lots, he and his colleagues could recharge the batteries of their FEVs.

Nadine Funke acts as a secretary in the **Administration** office at HKA. Formerly she worked at the public administration office in Karlsruhe, in the area of public parking spaces, where she had her first experience with the problems concerning the management of the charging infrastructure on public parking lots. For this reason, she does not drive an EV herself and uses public transportation instead. In contrast to her previous position at the public administration office, the tasks regarding the management of the CSs are, thanks to the introduced system, much more convenient and do not require a lot of manual work. By accessing the CSMS dashboard, she could filter for the wanted CS and have all the information in one place. However, blocking CSs for upcoming visitors is, despite the new system, still a huge overhead for the secretaries at the HKA. Up to now, she has had to select the specific CS and the corresponding connector to block it, and she has to unblock it again when the guest has arrived.

3.4 Role Mapping

To distinguish between the single users and their privileges in compliance with the range of associated functionalities, the system introduces a set of roles. Besides the role of the basic user, two variations of administrators and a user for demo purposes are provided. For a better understanding of these privileges and their purpose inside the system, the following Table 3.2 lists the relevant roles, their system shortages, and a brief explanation.

Table 3.2: Role collection provided by the system.

Role	System	Shortage	Description
Basic	`BASIC`	`B`	Standard user without administrative privileges
Admin	`ADMIN`	`A`	User with administrative privileges inside one organization
Site Admin	`BASIC`	`B`	User with administrative privileges for assigned sites
Site Owner	`BASIC`	`B`	User with administrative privileges for owned sites
Super Admin	`SUPER_ADMIN`	`S`	User with extended administrative privileges for the management of organizations and tenants
Demo	`DEMO`	`D`	Demo user with predefined login data, primarily used for demonstration purposes

Based on the available user privileges introduced above, which are provided by the system per default, the created personas, established in Section 3.3, are assigned to their corresponding roles afterward. The criteria for the assignment are primarily based on the tasks the users have to fulfill and with respect to their position inside the scenario and is presented below:

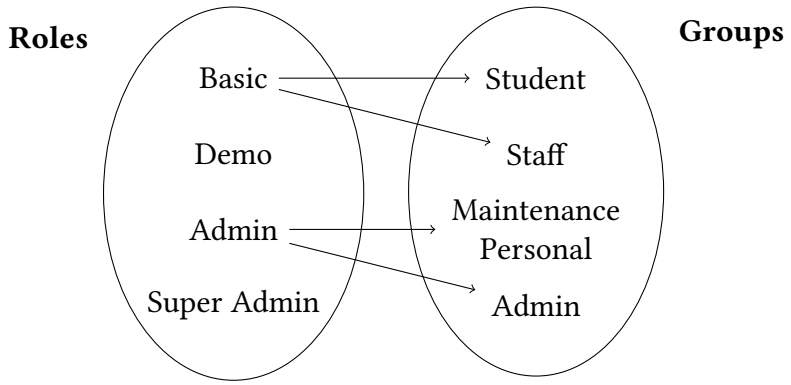


Figure 3.1: Mapping of the roles to the group identified in the scenario described in 3.1.

3.5 Goals

Based on the scenario and the corresponding stakeholders, the relevant goals for the resulting implementation are prescribed. As part of this work, a goal is handled as a requirement, the final product has to fulfill. Taking into account the different wishes in terms of the situations in which the respective stakeholders interact with the system, the following objectives could be defined.

Goal 1 - Management Capabilities As a major concern of this work the designed system, should serve as a central point for management purposes of the registered CSs. This should provide a basic set of functionalities covering specific administration tasks tailored for a certain group of users. Besides role mapping according to a subset of functions, the system should detect unintended usage and prohibit breach of privileges. Furthermore, the involved users will be provided with all relevant information by the system and know their current position within the different phases of the processes at each time.

Goal 2 - Self–Healing and Autonomous Processes In case of invalid user input or violation of predefined constraints inside the processes, the system should provide mechanisms for self–healing and autonomous error handling. In addition to increased consistency of data within the underlying database, the ease of use of the system should be a positive side effect.

Goal 3 - Support of relevant standards Considering the implementation of relevant communication protocols and the interoperability between different systems of CSs, EVs and CSMSs, the system should support well–established standards for communication and data exchange in the context of e–mobility. Alongside the standards like OCPP or OCPI, described in the Subsection 2.1.4, other alternatives exist in the further evolving industry with different targets in mind. Considering the already implemented standard inside the underlying product, the Open Charge Point Protocol should be the protocol for the communication between CS and CSMS.

Goal 4 - Modular Design In addition to the previously mentioned goals that focus on the implemented functionality, the architectural design of the software itself should also be considered. As part of a system that follows both a monolithic and a microservice–based approach, multiple services interact simultaneously alongside each other. The implementation and software architecture of this work aim to avoid compromising existing functionality through poor design choices or neglected separation of concerns. Preferably, the new functionality should be encapsulated as part of a module, which separates the system based on its responsibilities according to the definition in [11]. This should ensure easier extensibility of existing features and allow convenient maintainability.

3.6 Use Cases

The use cases illustrated in the use case diagram in Figure 3.2 were elaborated based on the goal definitions in the previous Section 3.5 and the stakeholder requirements from Section 3.2. Typically, software development processes such as RUP [42] differ between the groups of stakeholders who have an interest in the resulting product and the actors who are actually interacting with the real system. As part of this scenario, the stakeholders actually represent the actors used for defining the use case in regard to interacting with the system. The main use case, as highlighted by **goal 1**, involves managing and administering reservations within the reservation system. This encompasses creating, modifying, viewing, and deleting existing reservations made either by the users themselves or on their behalf by the administrator. Corresponding to **goal 2**, the *Scheduler* entity facilitates the required self–healing procedures. This necessitates the system to communicate with itself and employ background processes to manage forthcoming, current, and expiring reservations. Apart from the primary focus on non–functional requirements related to **goals 3** and **4**, an additional functional requirement with reference to the selective activation and deactivation of the reservation module could be identified. The privileges for maintenance and configuration of the particular organizations, represented as tenants inside the system, are part of the `SUPER_ADMIN` role. Usually, an external service provider outside the customer organizations is responsible for determining the scope of operation, for the service offerings provided to its customers. This role is referred to as `SUPER_ADMIN`. As there is no direct interaction with the system in terms of the reservation process, there is no dedicated actor assigned to this role within the above scenario. For comprehensively addressing all the aspects mentioned in the set goals, it is necessary to include this role with its corresponding functionality when defining the required features.

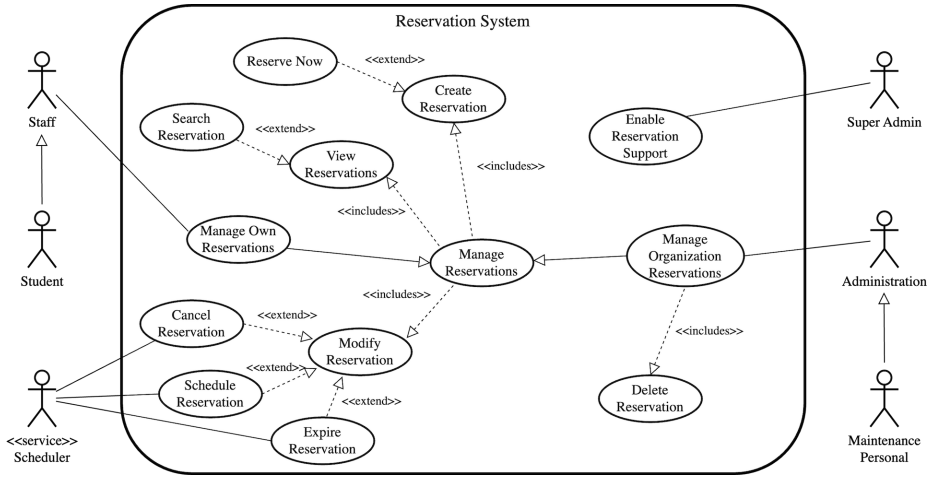


Figure 3.2: Use Cases considering the stakeholders of the reservation system.

4 Approach

This chapter describes the order and duration for processing the identified tasks, in accordance with the design specifications outlined in the previous Chapter 3. For this purpose, this work is divided into two parts, which will later serve to classify the aforementioned challenges. The first part is theoretical, including the paper's preparation and research to understand the problem. The second part afterward is practical, encompassing the tasks needed to implement the system.
In order to aid planning, timestamps are estimated for each task or logical unit of tasks to provide reference points during the processing of corresponding dedicated work packages. A logical unit, as used here, refers to a coherent cluster of individual tasks that share content or functionality and can be worked on simultaneously. Working on one task without the others would be less significant. The standard duration for processing a task or unit is set at two weeks (10 days) typically, to ensure sufficient time for completion. The duration may differ in certain cases and is determined based on the relevance and volume of related challenges. After describing the two conceptual parts alongside their respective tasks and processing times, this chapter concludes with a comparison of the planned procedure to the actual one.

4.1 Theoretical Part

With regard to the delimitation of the problem space and the synthesis of the central problem, the main aspect of the theoretical part of this work consists of a comprehensive literature review. This involves analyzing existing work on reservation processes used for managing CSs and allocating charging infrastructure to a specific EVU within a limited time frame, using various methods. With regards to the default processing time mentioned earlier, the duration for compilation as well as analysing articles and field projects related to this area is set to one month. The expected result should be used as the groundwork for the subsequent practical part, which is summarised in Chapter 5. As well as examining the current state of reservation systems, this review process defines a boundary for the necessary fundamentals of contextualization, resulting in Chapter 2. Therefore, systematic requirements engineering, taking into account the facts gathered during the research phase and the collection of baselines, using a customised scenario to select relevant use cases, results in Chapter 3. Chapter 6 presents the design phase, which follows the summarized requirements in Chapter 3 and the results of the literature review in Chapter 5. During this phase, mockups of the relevant interfaces as well as the underlying processes will be created. With the help of these designed concepts, the actual implementation of the reservation module resulting in Chapter 7 takes place. Subsequently, the theoretical

section concludes by validating the achieved functionality and presenting the final results, culminating in 8. Potential avenues for further research are also highlighted.

4.2 Practical Part

The practical part utilizes the results of the theoretical phase, as described in Section 4.1. Starting with the analysis of the current state of the applications used for the development of the prototypical reservation approach of this work, the final step is the implementation of the solution, taking into account the elaborated constraints and requirements. Seven iterations covering the implementation steps for the use cases described in Section 3.6 were conducted, commencing from the end of the theoretical phase that includes four weeks of literature research and information gathering, until the end of the processing stage. This allows for a six–week buffer to address any unforeseen features, correct any issues with the implemented logic, and complete the documentation tasks related to the theoretical part. Following this, a list summarising the scheduled iterations and associated timeframes will be provided.

Iteration 1 18.04.2023 – 01.05.2023

Iteration 2 02.05.2023 – 15.05.2023

Iteration 3 16.05.2023 – 29.05.2023

Iteration 4 30.05.2023 – 12.06.2023

Iteration 5 13.06.2023 – 26.06.2023

Iteration 6 27.06.2023 – 10.07.2023

Iteration 7 11.07.2023 – 24.07.2023

According to this iterative model, a functional increment of the resulting software should be provided after each iteration.

4.3 Actual Course Of Action

As there is a limited amount of available research and existing projects on reservation features for the management of CSs and charging infrastructure, the analysis and setup of the *Open e–Mobility* [26, 27, 28, 29] solution and its particular components commenced one day earlier. Consequently, some adjustments have been made to the above timescales for the general processing of the relevant parts. Beginning with the identification of missing functionality required for the subsequent implementation of the customized processes. The first step was to investigate the *e–mobility charging stations simulator* [29] for local simulation of a customizable charging infrastructure. Because of the identified feature gap regarding missing reservation capabilities conforming with OCPP 1.6 [54], the implementation of this crucial feature for testing and evaluation purposes was included as preliminary work for the *Open e–Mobility* team. After the team members successfully reviewed the implemented logic and minor reworks of the code base, the changes were merged into the standard product. Now, the extended CS simulator was then used as a reservable charging infrastructure for the subsequent iterations. Particularly, to validate the implementation

of reservations compliant with version 1.6 of the OCPP protocol, considering both pre–existing functionalities in the backend application and the adaptations according to this work. This included not only the backend development but also the implementations within the frontend application during iterations three and four. Due to the partially existing features mentioned above, the third iteration takes less time than expected, resulting in an earlier commencement of the relevant functions within the web frontend. In light of the time savings achieved in the preceding iterations, the fifth iteration for implementing the custom reservation function began prematurely as well. To account for the unexpected scope of required operations alongside an end–to–end development approach, the author decided to merge the fifth and sixth iterations, resulting in an extended fifth iteration and a total of six iterations instead of the originally planned seven iterations. Because of the previous schedule adjustments, the final iteration also started earlier and was completed later than intended. Primarily referring to the incidence of bugs and further requirements to manage reservations internally. The resulting schedule, including the iterations, their specific duration, and the corresponding tasks, can be found in Table 4.1 below.

Table 4.1: Schedule for implementing the reservation feature designed in this study.

Iteration	From	To	Objective
1	17.04.2023	28.04.2023	Feature gap detection within the existing application
2	01.05.2023	12.05.2023	Adjustments to the *e–mobility charging stations simulator* [29] to support reservations complying to OCPP version 1.6
3	15.05.2023	19.05.2023	Implementation of OCPP 1.6 reservation functionality in the backend application
4	22.05.2023	02.06.2023	Implementation of OCPP 1.6 reservation functionality in the web frontend application
5	05.06.2023	30.06.2023	Implementation of customized reservation process in the frontend and backend applications
6	03.07.2023	28.07.2023	Implementation of OCPP 1.6 reservation functionality and the customized reservation process inside the mobile application

Besides implementing the reservation system features, this work covered other tasks related to the *Green4EVer* [16] project, although they were only indirectly connected to this work. The decision to include these tasks is primarily based on the simplification of future deployment scenarios, such as demonstration or test cases, which require an automated and reliable setup. Therefore, this section mentions these tasks in addition to a brief explanation of the challenge. A detailed description is omitted from both the

subsequent *Design* chapter 6 and the *Implementation* chapter 7 according to no direct relation to the primary problem that this work covers.

Task 1 – MongoDB Manifests As the public repository offers no other deployment options, the only available one was containerization and operation using the Docker Compose orchestration tool [59]. To offer a more scalable and robust solution, the industry standard for deploying containerized applications, therefore, recommends employing Kubernetes (K8s) [62] as a sophisticated orchestrator. This led to this first task. Nevertheless, the manifest files essential for creating a *StatefulSets* [69] deployment, a standard way for deploying database applications inside K8s, were still not yet included in the current project. Hence, the MongoDB [48] database's custom setup, including test data and users, already existing as part of the local setup, was employed as a blueprint for creating the necessary manifests required for the deployment.

Task 2 – MongoDB Helm Chart To further reduce the effort required to deploy the database application to the orchestration instances, the Helm [33] toolbox and its charts for encapsulating K8s manifests are integrated. From the manifest files created in **task 1**, the corresponding Helm charts for the deployment are created, providing parameterization of a namespace and all required resources, such as *PersistentVolumes* [61], *ConfigMaps* [12] or *Namespaces* [50].

Task 3 – K8s Deployment For the integration of the remaining applications of the solution, additional Helm charts could be organized for the backend and the frontend application. Regarding the limited resources provided by the used orchestration service, the new charts required further modifications to integrate with the MongoDB chart and the deployment restrictions of the project. This involves a shift from the predefined microservice–based deployment strategy to a monolithic approach resulting in reduced resource allocation.

Task 4 - Ingress Deployment & Public Availability In order to access the applications within the cluster, it is necessary to use an *Ingress* [35] acting as a reverse proxy. Therefore, this ingress must be located within the *Namespace* the application lives in. Alongside the charts for the backend and frontend mentioned in **task 3**, the collection includes a configuration for deploying an NGINX ingress controller [51] using the predefined ingress class. In combination with a public DN and the corresponding IP address, the final stage of the deployment consisted of configuring the ingress deployment and registering it at a DN service.

As a result, the K8s setup shown in Figure 4.1, considers a deployment scenario that includes all the required components within a custom namespace, referred to in this project as ev. Alongside the particular applications, the namespace includes an NGINX ingress controller with a publicly accessible DN, and a MongoDB database with built–in reinitialization of custom test data and users. In terms of application distribution, only one node is used with multiple pods within the respective cluster.

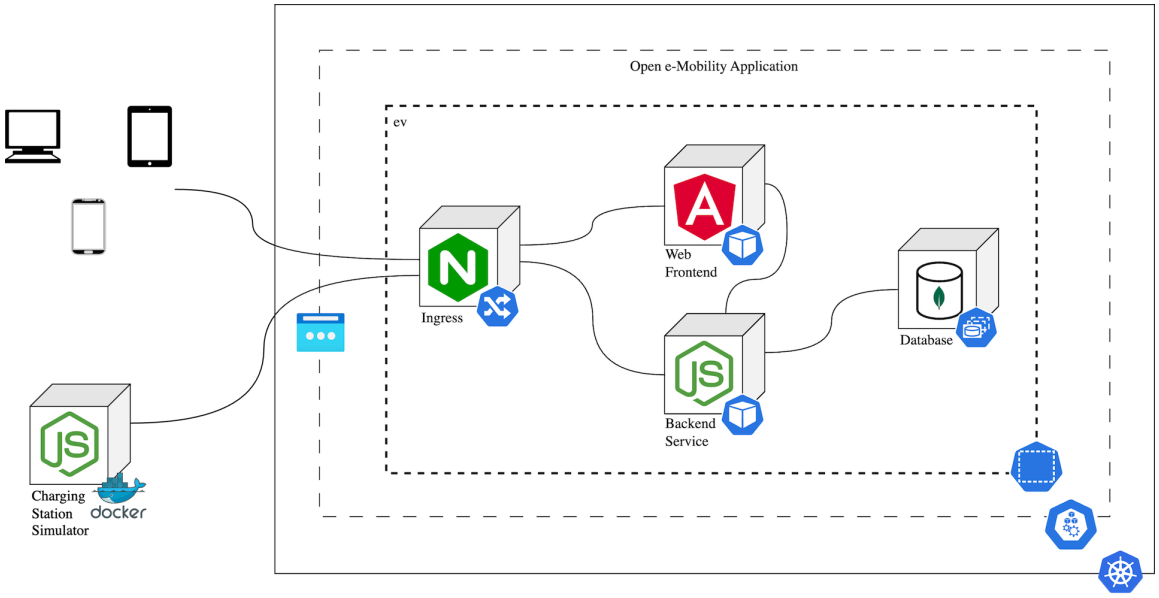


Figure 4.1: Elaborated K8s deployment architecture for the concerned solution.

5 Literature Review

As the primary part of the theoretical phase described in Section 4.1, this chapter provides an overview of research related to the problem statement of this particular work. Based on the existing studies, the current state of reservation systems for charging infrastructure management is identified and outlined in the last Section 5.2 of this chapter. Given the current state of implemented systems of this kind and the results of the available research, the design part in Chapter 6 will use these results for the design of the system and corresponding processes implemented in this thesis.

5.1 Related Work

In the context of supporting the wider acceptance of EVs in society, its decarbonization, and the reduction of air pollution caused by vehicles using ICEs [7], the integration of EVs in the daily lives of citizens is still ongoing and far from being complete. Obstacles like insufficient charging infrastructure, in comparison to the availability of gas stations, long charging times, and high investments for the public and private sectors [7, 57] are only a few examples mitigating the end users interest in switching to FEVs. For this reason, the surrounding e–mobility ecosystem, including the software for the interaction and management of the user, with the charging infrastructure and the convenience of the underlying processes used for the implementation, is an essential part. Given the system landscape and protocols employed in these scenarios, the ability to reserve a CS beforehand is usually lacking and has not yet been integrated into the current implementations of CSMSs. As part of the comprehensive literature review in this thesis, the following selected studies describe methods for applying reservation approaches and their respective processes to current implementations. This is intended to provide an extension to the existing system landscapes and protocols in this field, with the aim of achieving certain objectives.
In [8], the authors argued that the booking of charging infrastructure, especially for CSs, represents a pivotal role in a seamless integration of the CS into the transportation and mobility sector. Apart from the basic requirements for the design and implementation of an interoperable system, they provide a breakdown of the different advance booking approaches into four dedicated types. For designing their system, Basmadjian et al. used the EMSA [40] framework as a blueprint for the system model and its engineering and implemented one of their elaborated types as PoC in a showcase located in Bavaria, Germany. Proclaiming being the first contribution regarding the proposal of an interoperable booking system for EV charging, this study introduced the following four different kinds of reservations. Primarily based on the **start time** and the **reliability** a booking takes place, a differentiation between *Uncertain Ad–Hoc*, *Guaranteed Ad–Hoc*, *Uncertain Planned* and *Guaranteed Planned* is provided. The *Planned* reservation, in contrast to

Ad–Hoc one, allows the EVU to pre–block the respective connector, requiring start and end timestamps for the arrangement, instead of the immediate blocking of a CS during a pre–configured time span. Moreover, when taking into account the reliability of a booking and the corresponding charging stations, *uncertain* reservations are not able to guarantee a free parking space at the specified connector, whereas *guaranteed* ones are able to do so. Using the *planned* booking method makes it possible to perform multiple arrangements for one CS, unlike the *ad–hoc* setup where only one is allowed at the same time. From the requirements that the authors gathered during the design process of their study, such as improving the short–term planning capabilities of EMSPs and CSOs or increasing the fare convenience for the EVUs, their work resulted in the implementation of an uncertain ad–hoc reservation system, including a mobile application for the EVU, and the required backend services for communication and data exchange with the CSs. Concerning data exchange and modeling, they relied on the OCPP protocol and the ISO standard 15118 to facilitate standard communication between the system and the infrastructure. In addition to the proposed implementation, Basmadjian et al. reviewed available research on booking systems for CSs as part of the literature review conducted in this study. In the course of their research, the authors observed that the majority of studies solely focus on improving the satisfaction of EVUs by reducing their waiting time and associated loading costs, while simultaneously maximizing the utilization of CSs. Therefore, these papers applied algorithmic approaches for efficient scheduling of reserved timeslots like in [39, 43, 63] and do not provide sufficient and interoperable solutions for reserving charging infrastructure emphasizing the need for standard communication protocols like OCPP or common data models. Consequently, the authors concluded that, unlike other sectors, the implementation and analysis of arrangements for CSs and EVs is still in an evolving stage.
Apart from systems, which aim to manage the charging infrastructure through the targeted input of an EVU, other approaches, such as in [34], have tackled the problem of CS management in a more intelligent and automated way. Hsaini et al. also recognized the issue that the current versions of OCPP enable bookings only at the time of booking through the *ReserveNow* operation [54]. Therefore, the authors want to improve the aforementioned function, to allow the EVUs to reserve a CS in advance. As a result, they developed a mobile application for the EVUs, an algorithm for optimizing the charging schedule, a web application for monitoring purposes, and a backend, executing the operations based on OCPP. For creating the reservations, the EVU has to specify the arrival and departure time, the desired amount of energy, battery capacity, and optionally the initial state of charge with the desired state of charge by the mobile application. Once the optimization algorithm has been run in the backend application, considering the provided user preferences, a list of available CSs, along with the available energy and corresponding electricity costs, is presented to the user. For a successful reservation, the selection of the CS and the provided time must be confirmed.
A comparable approach is presented in [57]. By utilizing a mobile application, users can request a booking by defining their preferences and specifying their flexibility options in terms of arrival time, battery's SoC, and desired final charge. The system presents a listing of bookable slots, that are generated based on the findings of an optimization algorithm. This enables the user to finalize the booking, by adjusting the provided parameters through negotiation with the system itself.

Unlike the previous solutions, [71] suggests a solution that eliminates the EVU's interaction until a certain point in the reservation process. Therefore, Zarkeshev and Csiszar developed a concept based on an automated booking process that only requires notification of the user, using real–time information about the vehicle's status and location. Primarily designed for usage in smart cities and scenarios alongside the highway, each CS takes the role of a server with a pre–defined radius of operation. By driving through such a zone, the CS automatically identifies the EV and its battery level, which results in an automatic notification with a proposal to make a recharge, in cases of low charging level. Assuming the driver accepts the notification, the system will automatically reserve a charging slot at the CS. The system will assign a charging time based on the availability of the CS and the duration of the charge. These factors will be considered in relation to the status of the battery. Moreover, the server acts as a monitor, keeping track of all bookings and notifying the relevant drivers in the case of delayed arrivals, which could lead to cancellations or postponements of bookings, resulting in adjustments to the overall schedule.
When considering the communication and information exchange exclusively between the EVs and the CSs, implementing charging management into a smart grid system alongside scenarios like V2G could be the next possible step. By using reservations, the solution presented in [58], proposes opportunities to optimize resources on both the power grid side and the EVU side. In addition to previous solutions, Orcioni et al. developed their approach by extending the OCPP protocol with the prescribed *ReserveNow* functionality. This resulted in a mobile application for end users, in which the related OCPP operations are implemented in the backend service, allowing users to reserve a CS in advance, by negotiating charging parameters such as arrival time, duration, location, price, percentage of final charge and the required power. Moreover, a data model is provided that considers these parameters.
Based on smart grid integration approaches like V2G, other studies like [23] utilize the reservation approach for the design of a flexible user–centric architecture for capacity optimization of the underlying power grid, considering the energy requirement of the grid and its capacity restrictions. Therefore, Garcia et al. applied the SGAM [67] methodologies to provide a potential implementation of e–mobility as a distributed storage asset, also known as DER. Unlike the objectives of the previous studies, the modeling for booking a charge session through a mobile application, targets the balance of RES, along with the discharge of EVs compensating power peaks during times of high demand. As well as the other implementations, this system is also based on the OCPP protocol, and establishes a booking process that locks the CS after selection through an appropriate interface by the backend system. Unlocking takes place upon arrival and successful authorization via a token or in case of non–arrival of the driver.
Other algorithmic approaches using automatized processes to reduce the cost of charging or to increase consumer satisfaction of EVs, for example, also rely on reservation–based scheduling schemes. In this paper [39], for example, the authors propose a method for CS to decide the service order of multiple requests, prohibit the introduction of additional systems altogether, and try to extend the functionality of CS itself. To address this, Kim et al. developed a linear rank function based on the estimated arrival time, waiting time, and amount of power required to calculate the order of charging sessions. As a result, the person requesting the charging session arrangement may choose to take advantage of the

opportunity to recharge or may have to move to another station altogether.
Furthermore, in [20] Flocea et al. addressed the issue of uncertain availability of CS along the route of EVUs. This proposal aims to overcome the limitations of the *ReserveNow* feature defined in the OCPP protocol to enable drivers to plan longer trips by creating charging arrangements for upcoming days. Based on the CSs booking and transaction history, this solution is backed by an algorithm that generates the corresponding reservations in the form of intervals. This guarantees the availability of the reserved connector when it arrives at the station and avoids possible overlaps. To create a reservation, the user needs to choose a valid timeslot, including start and end time, resulting in the blockage of the respective CS and preventing other EVUs from charging their vehicles there. In order to manage the bookings internally, they are assigned to a particular status type, which describes their position within the overall booking life cycle in the system. Apart from *New*, *In progress*, *Completed*, the status *Cancelled* is introduced, which allows the system to treat the reservation according to certain circumstances. Moreover, the developed algorithm could adjust the arrangements according to the previous conditions and assume that the reserved time counts as charging time. When the reserved time comes to an end, charging will stop automatically.
In terms of a more elevated perspective on booking systems and potential architectures considering ways, to select required stakeholders and requirements, [7] presents a reference architecture to conceive interoperable systems proposals. The benefits of such architectures for future booking systems in this context and the key stakeholders needed during the requirements engineering phase are listed by Basmadjian et al. in this paper. Besides the identification of the necessary requirements of the aforementioned stakeholders, the designed reference architecture, and a PoC developed using this design, the authors introduced a set of design parameters for reservation systems. In order to achieve demand–side management and capacity planning of existing charging infrastructure through these types of applications, the following design parameters proposed by the authors must be considered: *Enforceability*, *Planning*, *Fee*, *Data Availability*, *Roaming* and *Scheduling*. When it comes to parameter coverage, each system can only fulfill a certain subset. The selection of these subsets should be based on the objectives the reservation system aims to achieve, during the design stage. As a result, different types of systems may be created, each tailored to a specific purpose or scenario.
In contrast to the majority of research addressing only fixed charging stations as described in Subsection 2.1.2 by the classification of available charging infrastructure. The approach of [72] especially targets reservation processes for the utilization of mobile charging stations within a booking system. The authors offer a design approach for an intelligent mobile charging control mechanism for electric vehicles, where mobile charging is promoted as an alternative recharging solution, using mobile plug–in chargers to facilitate on–site charging service scenarios. Relying on a reservation–based scheduling scheme that approximates optimal solutions for mobile chargers circulating between parked vehicles with charging appointments, the aim is to use mobile vans with plug–in chargers as CSs to provide a versatile charging service. With the introduction of charging session bookings, accurate estimates of future charging demand can be made and the strain on charging infrastructure and lack of CSs in certain areas could be alleviated.

5.2 Current State

Considering the various processes and methods that propose approaches to managing CSs through reservations described in Section 5.1, the current state of this particular type of system for charging infrastructure management is illustrated below.
Firstly, this section describes the similarities in the technologies used and the overlaps identified in process design and implementation. Most of the investigated systems include, in addition to the mobile applications for the EVUs usage, one or more backend services, that handle the information exchange between the mobile frontend and the corresponding charging infrastructure. For reservation creation and communication between the CSMS and the CS, all systems rely on the OCPP protocol, as an open standard protocol for charging infrastructure communication. In order to mitigate unforeseen circumstances affecting the arrangement made, the systems found included background processes, ranging from cancellation of the booking after a certain period of time, to automatic adjustment of the underlying charging timetable to reschedule the upcoming charging session. Moreover, certain methodologies suggest utilizing physical obstacles, such as sensors or system–controlled blockers, to ensure the successful completion of the booking process.
The most significant difference between the proposed solutions is the way an arrangement is modeled as an entity within the system and the degree of interaction with the EVU. From fully automated scheduling algorithms, that create bookings based on calculated intervals using vehicle and location data, requiring only confirmation from the user to minimize the need for user intervention, to methods that require various user inputs such as actual battery status, estimated time of arrival or desired battery charge level at the end of the booked session. Some proposals have combined both approaches and established negotiation processes, that include game theoretical aspects to negotiate the parameters of the arrangement directly with the backend system or the CS. Therefore, various properties are needed to model the booking inside the system. The properties that most reservations naturally support are the corresponding CS and connector, as defined in OCPP, as well as the start and end time, which extends the aforementioned protocol and simplifies appropriate scheduling for the system.
Despite the fact, as mentioned in [8], that such systems lack general, as well as generic design criteria for conceptualization, due to minimal research and applications inside real scenarios. However, this work identified certain common design criteria that all existing proposals share. Primarily, this is relying on the required functionality and the objectives they have to fulfill. Besides incorporating payment options and integrating self–managed rescheduling algorithms to reduce overlapping, additional conventional design choices are present within the examined systems.
Based on the results obtained, such as the reduced travel time, depending on the demand for the charging infrastructure and the type of reservation chosen, the application of such solutions does not directly increase the utilization of the underlying charging infrastructure. Moreover, studies on alternative charging approaches like mobile CSs, that provide more flexible processes, have only just scratched the surface in the existing literature. Nor a focus on the interoperability of the resulting implementations could be identified.
Concerning the architectural design of the solutions, most of the found functionalities extend or integrate with the OCPP standard and place their custom enhancements on

top of it. Only in [20] a modular approach is provided as a separate functionality, co–existing with the OCPP reservation functionality within the CSMS. In terms of features such as the implementation of recurring arrangements, for more than one charging session within a specified date range, no elaboration exists. The same applies to using role concepts for assigning specific functions exclusively to particular roles for enhanced control over infrastructure management. Moreover, there is no control flow available for precise regulations or differentiations of specific phases that a reservation is capable of undergoing.

6 Design

This chapter presents a design that takes into account the processes, cross–checked with existing approaches for booking systems in the context of e–mobility, as discussed in Chapter 5. Building upon the definitions of requirements and use cases outlined in Chapter 3, the following Sections deal with the conceptualization of the required entities, such as the arrangement, as well as the functionality for sufficient management capabilities of the underlying charging infrastructure. Furthermore, the mockups created during the theoretical part of this work are associated with the corresponding capabilities that the system has to satisfy. These interfaces, representing the flow of user interaction, are combined with the related processes implemented in the underlying system.

6.1 Reservation

As a fundamental unit, the reservation entity and its constituents serve as the foundation for the design process. Hence, multiple sources, such as the OCPP standard, which is employed by most CSs, and the associated research in Chapter 5, are given primary consideration. To cover the basic functionality for interacting with the underlying charging infrastructure, the booking must contain specific properties, that enable an unambiguous mapping between the entities provided in the standard solutions and those internal to the system. Considering the absence of features not covered by existing standards, this thesis introduces several properties as a dedicated extension to the existing work, which should provide further possibilities in the management of bookings and related infrastructure. Starting with the requirements for compliance with version 1.6 of the OCPP standard, the selection inside Table 6.1 includes the same essential information as the aforementioned protocol.

Table 6.1: Reservation properties based on *ReserveNow* operation in [54].

Property	Description
ID	Distinct identifier for the entity in the CS and system
Charging Station ID	Association to the corresponding CS
Connector ID	Association with the relevant reserved connector
Tag ID	Association with the RFID tag for the user
Parent Tag ID	Association with the superior RFID tag
Expiry Date	Date when the booking expires

Based on these standard properties required for the OCPP *ReserveNow* operation, the author proposes the extensions listed in Table 6.2.

Table 6.2: Reservation properties extending OCPP protocol in version 1.6.

Property	Description
From Date	Date the reservation begins
To Date	Date the reservation ends
Arrival Time	Dedicated time the reservation starts
Departure Time	Dedicated time the reservation ends
Status	Reservation status according to 6.1.2
Type	Type of reservation according to 6.1.3
Car ID	Association to users car

Especially these enhancements enable the system user to make a reservation in advance, in addition to the already supported immediate reservations for locking connectors directly, by booking a single or a recurring arrangement within a selected date range and time slot. Furthermore, the option to add the car to the booking ensures the compatibility of the system for the integration of SG applications, e.g. providing V2G operations. The introduction of the *status* and *type* properties serves as markers to distinguish between booking types and the current status. The details of this are discussed below. Properties such as prioritizing different categories of users or integrating a pricing scheme, are not covered by this work and could be developed as part of future work, to extend this system approach. To fully understand the new relationships between entities in the system and their use in the proposed framework, the next subsection outlines these new associations.

6.1.1 Entity Relationships

For the purpose of arranging the properties and entities within the reservation module and the rest of the system, it is necessary to examine the relationships that the specific entities, such as the CS or a RFID tag, have with an arrangement and the quantity in which they can be used. For presenting this type of mapping concerning the relationships, they are depicted as an entity relationship diagram to show the respective connections and to illustrate them in Figure 6.1. Stated as a constraint by the OCPP standard, it is intended that an individual can only reserve a particular CS and connector during a given time. Additionally, either an individual user, identified by their associated RFID tag or a group of profiles linked to a parental RFID tag, that represents this subset, may use the reserved charging slot. The chosen vehicle must also adhere to these restrictions.

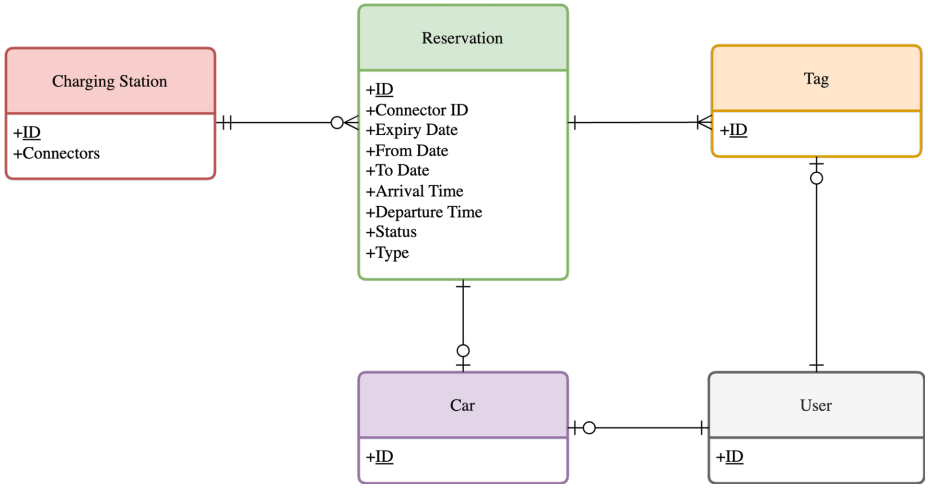


Figure 6.1: Entities and their relationships with the corresponding quantities.

Beyond the relationships established during this design process, the succeeding subsections detail the newly introduced properties in terms of their relevance to later implementations and the functionality they provide.

6.1.2 Reservation Status

According to the findings of Flocea et al. [20], the reservation process progresses through several stages before it is completed. These stages can be depicted as a status, which indicates where the booking is currently located in the process and provides an overview of the options available to the relevant user or the system itself to interact with the arrangement. Thus, the study mentioned above introduced four distinct states, distinguishing between *New*, *Done*, *In progress*, and *Cancelled*. Considering these pre–existing states, this thesis proposes an extension that allows a more granular treatment of this kind of entity during its life cycle. Besides breaking down the particular steps within the life of a booking, these additional states allow further functionality in terms of mitigating undesired behavior and enforcing software–based rules in terms of the design parameter *Enforceability* mentioned later.

Table 6.3: Introduced states to extend the reservation life cycle.

Reservation Status	Equivalent in [20]	Description
Done	-	EVU was present and charged the EV
Scheduled	New	Planned reservations whose start date has not yet been reached
In Progress	In progress	Reservation currently in progress
Cancelled	Cancelled	The reservation has been cancelled
Expired	-	The reservation has reached its expiration date.
Unmet	-	The EVU did not arrive punctually

Taking into account the current states, the proposed extensions additionally cover the scenarios of a user not arriving on time, the expiration, a successful fulfillment, and partially the scheduled state until the start of the reserved time slot. These aspects are not considered by Forcea et al. and should be integrated to address certain situations. To characterize these particular situations, the state diagram below, shown in Figure 6.2, provides an overview of the possible transitions according to the conditions that must be met.

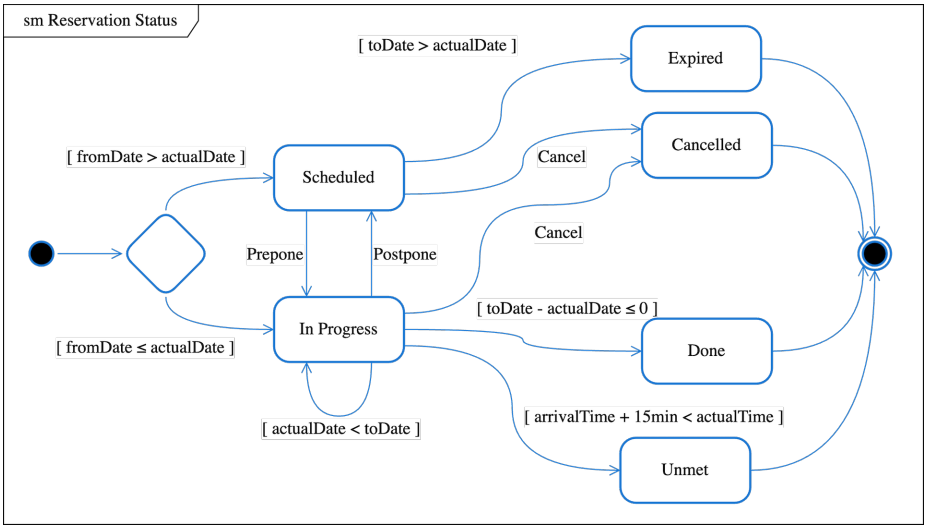


Figure 6.2: State diagram with the corresponding state transitions for the possible reservation states within the implementation.

Not all reservations undergo the same stages as depicted above due to their specific type. To provide an overview of each type included in the system design, the next subsection offers guidance on the potential variations.

6.1.3 Reservation Types

As outlined by Basmadjian and colleagues, a booking system can accommodate various types of bookings. In addition to the immediate blocking of resources through the *Ad–Hoc* reservation mechanism, the *Planned* ones allow for advance booking, opening up a wider range of possibilities for incorporating this functionality into the management of charging infrastructure [7, 8]. For this purpose, the authors distinguish between the established types based on their *enforceability* and propose uncertain and guaranteed reservations, which can be combined to form four possible variations. The *uncertain* reservations cannot guarantee that the reserved CS and its corresponding parking spot will not be occupied by another vehicle upon arrival, whereas the corresponding *guaranteed* type ensures this possibility. Enabling this reliability requires the implementation of additional functionalities, such as the ability to control the physical infrastructure itself. In this design process, considering no direct access to physical infrastructure services as blockers for parking and similar technologies, the resulting system tries to provide a hybrid type combining the best from the world of *uncertain* and *guaranteed* reservations using only software–based solutions to provide the desired level of *enforceability*. Building a connection to the types found in the existing literature, the Table 6.4 then maps the types used in this design and the subsequent implementation to the existing ones.

Table 6.4: Pre–defined reservation types and their mapping to the types used in this work.

Reservation Type	Equivalent in [7, 8]	Description
Reserve Now	Ad–Hoc	Reserving the CS and its corresponding connector immediately until a specified expiration date is reached. During this time, no other user can use the same connector for charging purposes.
Planned Reservation	Planned	Reserves the CS and the corresponding connector in advance, enabling recurring bookings by selecting a date range. The connector and the CS will stay available until the specified timeslot starts.

After introducing the respective entities, the next section discusses the design of the system itself, including the relevant processes and the overall architecture.

6.2 Reservation System

To empower the user to create and manage their arrangements, a reservation system is required. This section presents a design proposal for the development of the system, which is used as groundwork for the adjacent chapter which aims to put such a system into practice. Firstly, the necessary capabilities considered relevant in this work are discussed, as well as the needed design criteria in terms of the system and its architecture.

6.2.1 Design Criteria

Establishing a functional system approach for managing charging infrastructure entails the consideration of several design parameters, as outlined by Basmadjian et al. [7]. Apart from *Enforceability*, *Planning*, *Fees*, aspects such as *Data Availability*, *Roaming* and *Scheduling* are important design parameters a system design has to address. Concerning the given options and possibilities present in this design approach, the author of this work opts for the inclusion of *Enforceability* alongside *Planning*, *Data Availability* and *Scheduling*. Furthermore, the FCFS approach has been selected as the most appropriate method for mapping real–world charging session bookings.
In accordance with the terminology employed by [7], the system can be defined as follows:

$$\left\{ \text{"Limited"}_{Enforceability}, \text{"Yes"}_{Planning}, \text{"No"}_{Fee}, \text{"Yes"}_{Data\ Availability}, \text{"No"}_{Roaming}, \text{"FCFS"}_{Scheduling} \right\}$$

To accomplish these goals, the relevant capabilities that attempt to meet these design criteria are outlined below.

6.2.2 Management Capabilities

When managing various entities, the most commonly utilized functionalities involve creating, updating, or deleting related information. Alongside these processes, the acronym CRUD is often referenced. In reservation systems, the context permits an additional operation known as cancellation, which invalidates the selected booking. As outlined in Chapter 3, the relevant use cases for the previously mentioned functions are already identified. In addition to these general features, the subsection also explains the use case for enabling them on demand. Considering these capabilities, the following management capabilities cover at least parts of the design choices such as *Enforceability*, *Planning*, *Data Availability* and *Scheduling*, alongside the scheduling capabilities described in Subsection 6.2.3.

6.2.2.1 Create Reservation

In order to enable all of the subsequent procedures, reservation creation is a mandatory function, which can be considered the initial process. The user fills in his information through the chosen application, followed by system–side validation that leads to further processing in subsequent steps or immediate termination of the process. Afterward, the system checks for any existing bookings with the same identifier to avoid duplicates and ensure consistency with respect to the underlying database schema. If one with the same identifier exists, the process tests for the same user according to their RFID tag and updates the record according to matching tags or rejects it to prevent privilege escalation. Next, the system checks for other arrangements in the defined time range and ensures that there are no collisions, also known as overlaps, between individual bookings according to their dates and time slots. Any collisions detected, as well as the escalation of privileges during the update process of an existing reservation, result in a predetermined end. If no conflict is detected, the respective type is checked and, in the case of a *ReserveNow* type, the encapsulated *ReserveNow* operation defined in the OCPP standard [54] is triggered. This operation prompts the system to send a request to the relevant CS to immediately lock the connector. Otherwise, when dealing with a *Planned Reservation*, the selected time range will be checked, and if the start time is approaching, similar to the case of a *ReserveNow* methodology, the appropriate CS is contacted. Finally, the entity is saved and a notification, described in more detail in Subsection 6.2.4, is sent to the user to inform him or her about the successful creation.

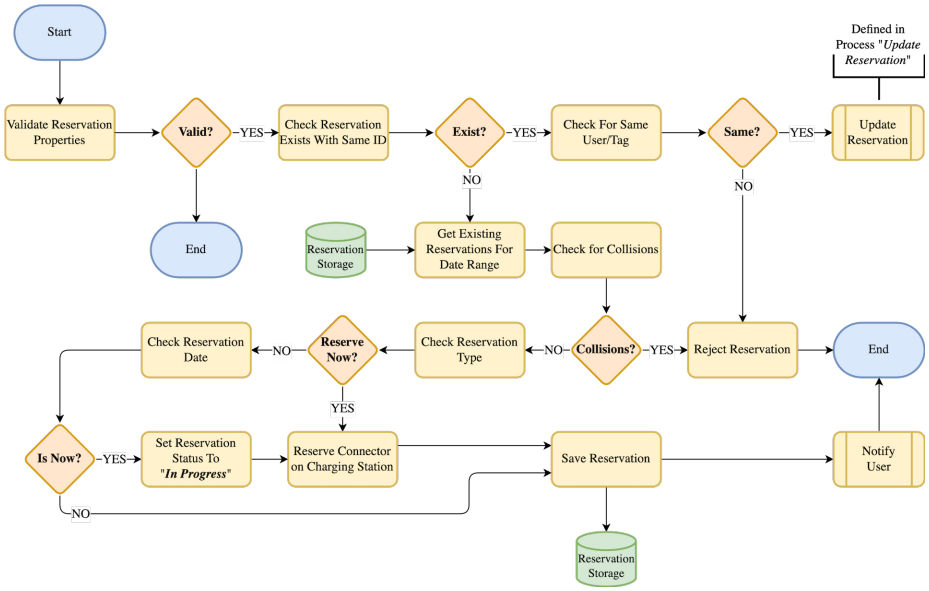
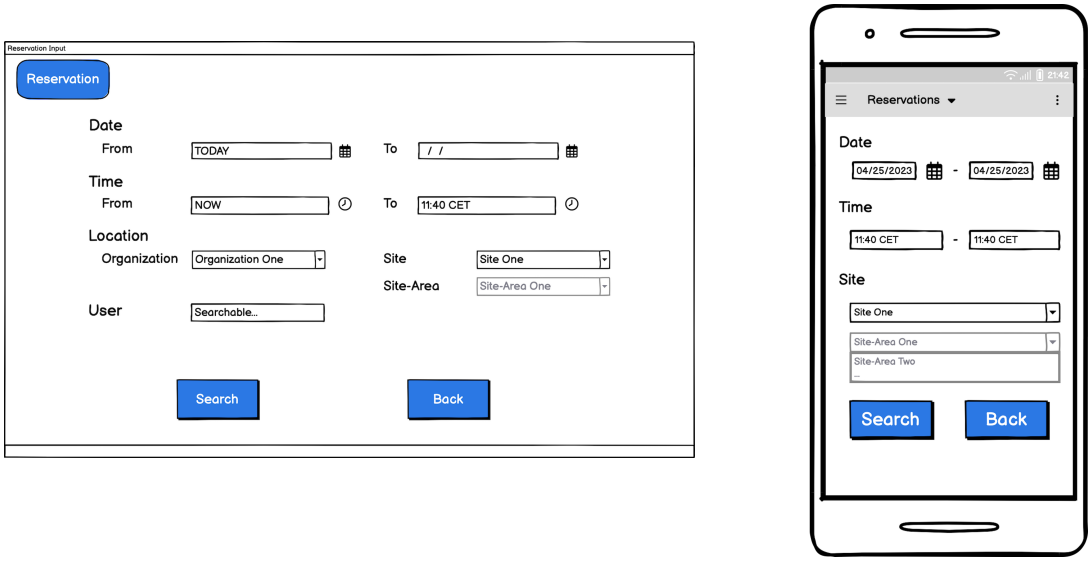


Figure 6.3: Process flow with all necessary steps to create a reservation.

To handle the essential inputs, the subsequent mockups offer a suitable user interface encompassing all the required information provided by the user. To address the collection of this vital data, a proposition for the graphical user interface (GUI) of both the web and mobile app is presented.



(a) Create a reservation in the web application.

(b) Create a reservation in the mobile application.

Figure 6.4: Mockups for the user interface of the mobile and web application relating to the creation of reservations.

6.2.2.2 Update Reservation

To modify and manage existing bookings, the user is able to update a reservation. In addition to changing the date, time and CS, modifications can be made to the connectors as well as the provided vehicle, if available. Similar to the creation process, the corresponding user input and changes are validated, resulting in further processing of the update process or immediate termination. Referring to successful validation, the system searches for a record with the same ID to update it accordingly. To capture potential events, such as unsaved arrangements, the update process turns into the create process as described above to mitigate the erroneous behavior. Otherwise, the properties received are updated. If another time or CS is selected, the reservation on the previous station is cancelled, and the newly selected CS and connector are reserved. Afterward, a series of steps similar to the creation process described above takes place. This includes checking for new collisions created by other users during the update and rejecting them accordingly, preparing the notifications for the selected CS, and finally saving the updated record. This also contains another notification that the user will receive upon successful completion.
Due to a comparable range of input, the recommendations for the respective interfaces of the mobile and web applications are almost identical, as shown in Figure 6.6 below.

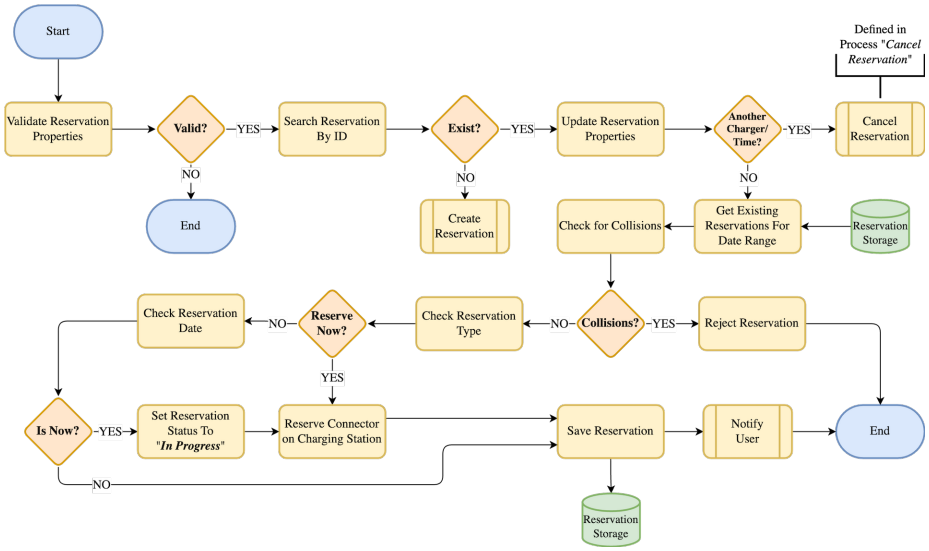
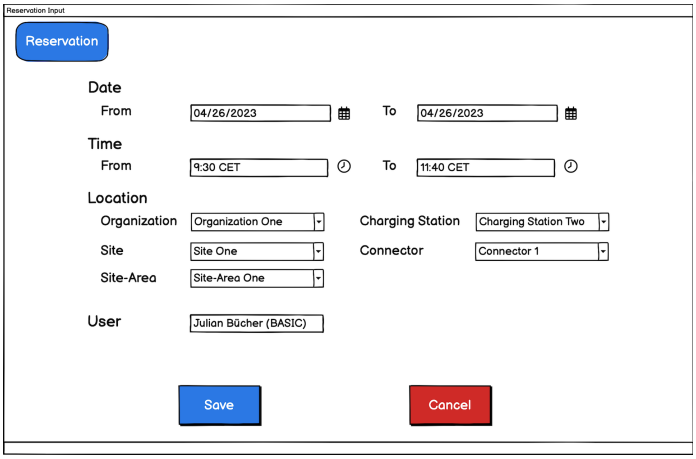


Figure 6.5: Process flow with all relevant steps to update a reservation.



(a) Update a reservation in the web application.

(b) Update a reservation in the mobile application.

Figure 6.6: Mockups for the user interface of the mobile and web application concerning the update of reservations.

6.2.2.3 Cancel Reservation

In addition to making a reservation, the user needs the option to cancel it in the case of changes due to unforeseen or unexpected circumstances. To accomplish this task, the cancellation operation encapsulates the predefined *Cancel Reservation* feature in OCPP version 1.6 [54]. As well as cancelling the active booking on the CS, it also allows cancelling bookings in advance, taking into account the proposed life cycle as described in Subsection 6.1.2. Initially, the designed process checks for an existing record with the same identifier in the database. If such an entity does not exist, the execution ends. This practice avoids the processing of unnecessary information and prevents communication with the corresponding customer service. If the arrangement exists, the status is validated to cancel only bookings according to the predefined state transitions in Figure 6.2. Furthermore, the current status of the entity is determined, which results in a *Cancel Reservation* request contacting the appropriate CS if the booking is in progress. After the status is set to *Cancelled*, the record is saved once more, and a user notification containing the status transition, is sent to the user.

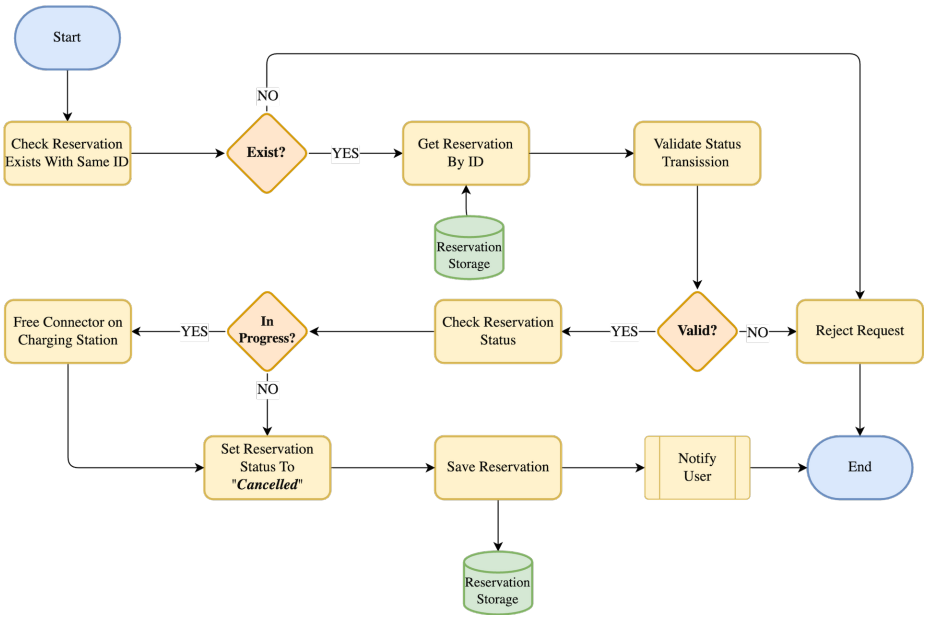
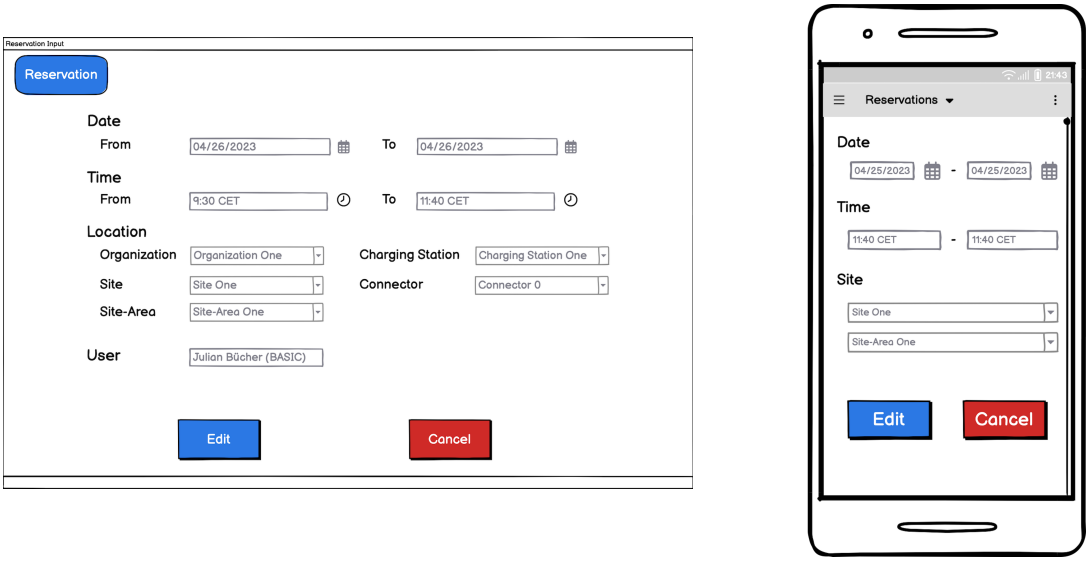


Figure 6.7: Process flow with all according steps to cancel a reservation.

By virtue of the atomic nature of this function, it is designed as a dedicated button within the mockups presented below. To confirm the cancellation of a booking and complete the process, users will be presented with a confirmation dialogue in both applications, which is not displayed in the illustrations below.



(a) Cancel a reservation in the web application.

(b) Cancel a reservation in the mobile application.

Figure 6.8: Mockups for the user interface of the mobile and web application in order to cancel a reservation.

6.2.2.4 Delete Reservation

Keeping in mind the management capabilities and complying with Article 17 of the General Data Protection Regulation (GDPR), which outlines the 'Right to Erasure' [6], the system offers the functionality to delete an arrangement. Aiming for comparable outcomes, this process operates in a similar fashion to the cancellation process explained previously, which also affects the design of the corresponding mockups. Initially, the system storage is checked for the existence of the provided identifier, which results in process termination if the record is not found. Otherwise, the booking is queried and the status is checked. If the status is *In Progress*, it is immediately cancelled. Afterward, the operation permanently deletes the record in the database.

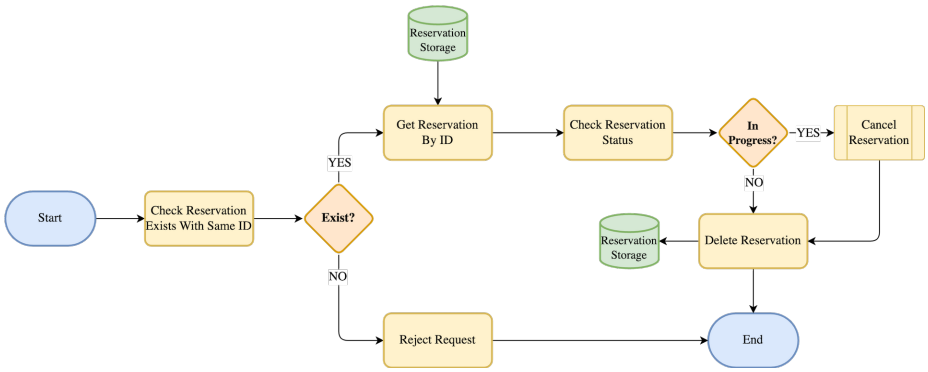


Figure 6.9: Process flow with all according steps to delete a reservation.

As mentioned previously, no proposals have been made for the corresponding user interfaces. The author suggests the implementation of the delete feature in a similar way to the *Cancel Reservation* method discussed earlier. Therefore, no additional design approach for the application interfaces is required.

6.2.2.5 Enable Reservations

Considering the reservation system as an independent part of the whole system, the user should have the possibility to activate or deactivate the respective functionality according to his needs. To incorporate this functionality, it is presented as a toggle to control the operation of the system with all its procedures. Similar to the approach presented in [57], which describes the reservation module as a dedicated service. According to this proposal, it is possible to deactivate it as a module integrated as a subsystem. This allows such functionality to be implemented alongside the processes described in existing standards, such as OCPP or OCPI, and does not interfere with their implementations. Like the *Delete Reservation* function, no user interface mockups are provided due to the simple nature of the function. In the subsequent implementation, this toggle should be considered within a central management interface that configures the functionalities of the different tenants.

6.2.3 Scheduling Capabilities

To address the scheduling capabilities discussed in Subsection 6.2.1, the subsequent procedures have been developed to manage reservations independently. This enables the system to maintain a consistent state and autonomously manage the associated charging infrastructure. For this purpose, the scheduling, expiry, and release procedures are created to handle the reservations according to their current state.

6.2.3.1 Schedule Reservation

By utilizing the scheduling process, the system identifies present and forthcoming reservations from the database according to a pre–defined threshold. If there are no upcoming reservations in the near future, the scheduler process is terminated, otherwise, the CSs and connectors defined in each arrangement are reserved. Similar to the create and update functions described within the management capabilities Subsection 6.2.2,
the OCPP *ReserveNow* operation [54] is implemented to provide standards–compliant communication between the system and the CS. If connectors are in an occupied state, the *Stop Transaction* operation is performed, resulting in an immediate termination of the current charging session and returning the CS to an available state. Allowing the system to carry out the upcoming booking on the released connector, resulting in the CS accepting the *ReserveNow* request. Consequently, the status is updated to *In Progress* and saved once again.

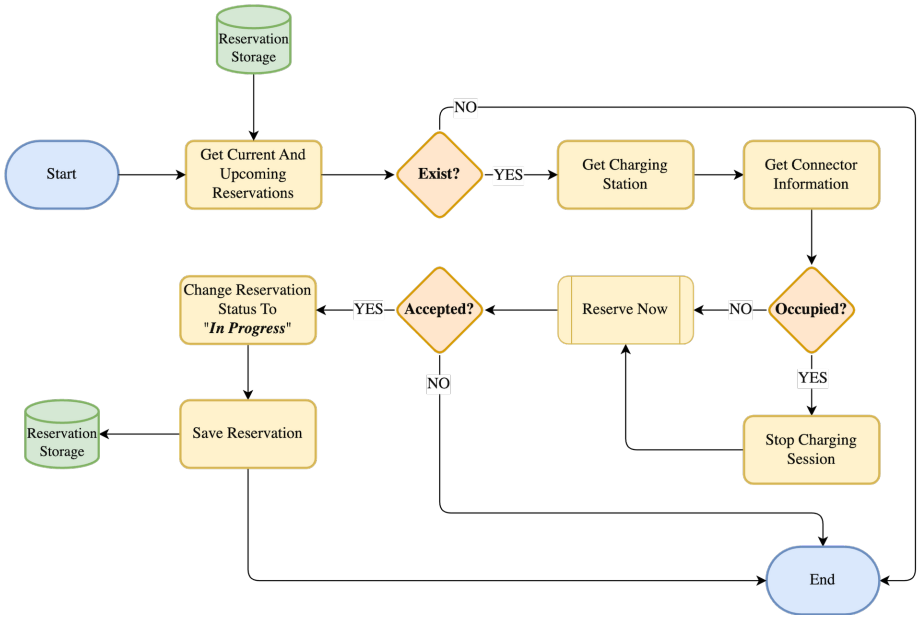


Figure 6.10: Process for scheduling upcoming reservation on a specified CSs.

Besides providing autonomous functionality for synchronizing reservations with the corresponding CSs, this procedure allows the execution of recurring reservations during the defined period. Furthermore, if the user disconnects the connector during the reserved

period, the underlying logic ensures that the arragement remains active until a certain threshold is reached.

6.2.3.2 Expire Reservation

Forming the counterpart to the scheduling process, which synchronizes upcoming and ongoing reservations with the CSs. Designed as an expiring process, it handles all reservations that are neither cancelled nor expired and have reached their expiration date. The process ends immediately upon discovering such reservations. Otherwise, the status is changed to *Expired*, the user is notified and the record is saved again.

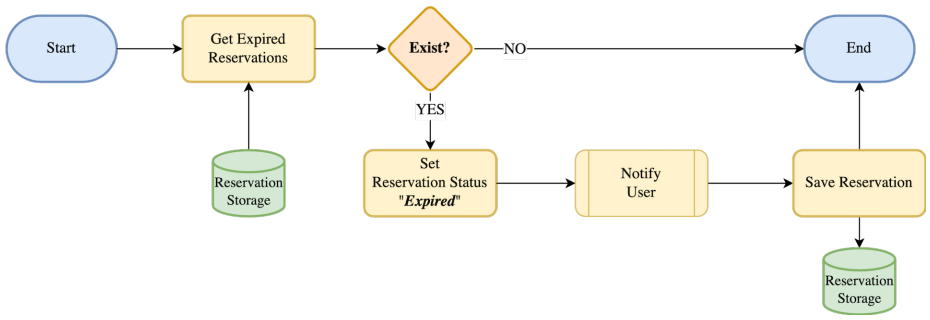


Figure 6.11: Process flow for reservations reaching their expiration date.

In cooperation with the *Free Reserved Connector* procedure described in the Sub–subsection 6.2.3.3, the *Expire Reservation* procedure handles the transition from the active state to a final state that invalidates the reservation. Using the *Expired* status deals with not correctly updated reservations according to the *Done*, *Cancelled* or *Unmet* states. Mostly caused by information or connection loss between the management instance and the CS, such outtakes eventually occur. This approach ensures that concurrent active reservations are minimized and provides a self–healing process similar to self–healing processes provided by operating systems, to clean up the growing data store.

6.2.3.3 Free Reserved Connector

In the case of a reserved connector being blocked, the respective user who made the booking did not show up and failed to cancel the reservation, thus preventing the connector from being available again. To reduce the likelihood of this situation, the system provides a process to automatically remove unmet reservations from the CSs and associated connectors. In order to identify these bookings, the system selects reservations that are already in progress, the connector is not in the *Charging* state and the specified arrival time is overdue by a certain threshold. After identifying reservations that meet these criteria, the system cancels them on the CSs. Once the station successfully acknowledges the cancellation, the status is updated to *Unmet*. Otherwise, the process ends immediately. The same rule applies if the CS does not permit the cancellation of the reservation.

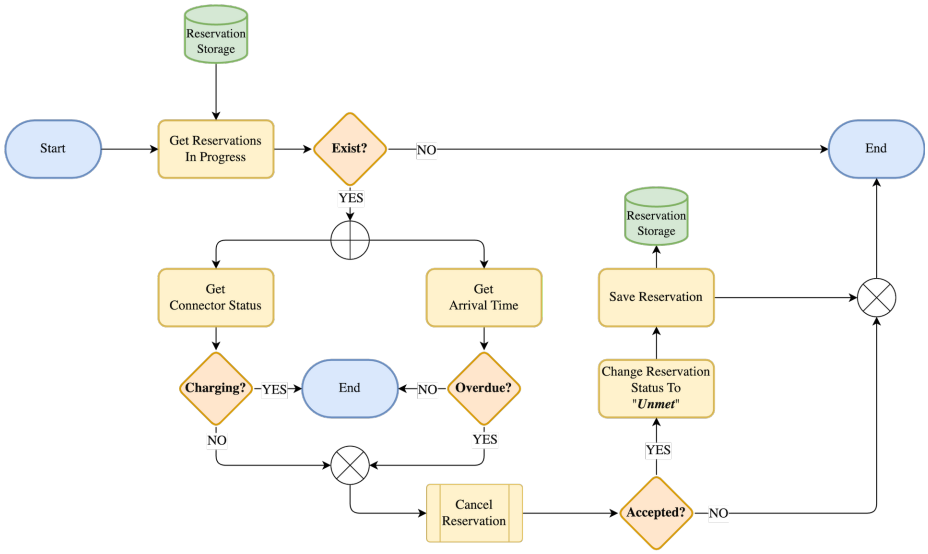


Figure 6.12: Process flow for releasing connectors with unfulfilled reservations.

6.2.4 Notification Capabilities

Besides the management and automation techniques presented above, the exchange of information with the user encompasses one aspect of the predetermined design parameters. Concerning the methods in [71], the system should interact with the user in a certain way to inform him or her about upcoming reservations or schedule changes, which the user should consider for further planning activities. For this reason, this system design outlines several scenarios, that the author considers to be relevant, for the solution to inform the user according to the changes made to the reservations. This also satisfies the aspects of the *Data Availability* design parameter in a more general sense. Within the scope of the aforementioned work, this parameter is limited solely to data regarding the status of charging stations and connectors. However, additional expansions could be made to include data on reservation statuses. For a clearer insight into the constraints associated with each notification type, a brief explanation of the concerning scenarios is succeeding.

6.2.4.1 Reservation Status Changed

Besides the users' ability to manually configure their reservations, the bulk of the work is performed by background tasks that run parallel to the main processes. For standard users, it is neither feasible nor beneficial to regularly monitor the status of their reservations on their own. Thus, it is necessary to send notifications containing information about changes in the status of an arrangement throughout its life cycle. Due to its supportive role within the system context, this type of functionality could be integrated into one of the previously designed processes. Considering an intended integration, the next parts of the design processes could be proposed as suitable extension points, using the notification capabilities within *Schedule Reservation*, *Expire Reservation*, and *Free Reserved Connector*.

6.2.4.2 Reservation Upcoming

To address the emergence of reservations, two distinct types of notifications can be extracted from the more generic one. One is the notification of the user to whom the reservation belongs and the other is the notification of the user who intends to charge at a CS, if a booking comes up in the near future. Both scenarios are represented by the notification type of a *Reservation Upcoming*.

Upcoming Reservation Notification According to the process of scheduling reservations, users must receive a notification, if their booking is due to start in the near future. This type of notification should provide the user with the relevant metadata about the reserved charging session and its commencement, ensuring they are informed about the upcoming reservation.

Upcoming Reservation Warning Message, that the CS the driver is currently using, has an arrangement in the pipeline, which conflicts with the estimated duration of the charging session. This should persuade the driver to consider relocating his car. Otherwise, the charging session is stopped when the particular reserved timeslot starts and the EV is no longer charged, making it pointless to occupy this station.

Locating these specific notification types, using them within the *Schedule Reservation* feature as part of the previously mentioned designs might be the best option.

6.2.4.3 Charging Station Blocked

Considering the *Upcoming Reservation Warning* mentioned previously, only the user occupying the reserved CS is notified that it is potentially blocking a reserved port. To complement this scenario, the user of the upcoming arrangement belongs to requires notification as well. To guarantee that the user is informed of the occurrence of unintended behavior, the *Charging Station Blocked* notification fulfills the objective of the desired information flow. In this way, if the CS is still blocked and the other EVU does not remove the car from the parking lot accordingly, the notified user is able to switch to another available CS in that area. Despite stopping the charging session in accordance with the *Schedule Reservation* process, the vehicle could still end up blocking the parking place for the arriving user, resulting in a negative user experience. Therefore, the system is intended to provide such mitigation techniques to inform the user in advance and allow him or her to adjust the reservation. Considering the integration of this particular notification, the *Schedule Reservation* as a part of the designed processes seems appropriate.

6.2.4.4 Reservation Cancelled

When considering the *Reservation Status Changed* notification introduced earlier, this notification type seems to be a duplicate of a still–existing function. However, there should be a specific notification in the system design to inform the user, when a reserved charging session is being cancelled. The system's implementation and internal administrative structure do not only allow the user, who created the arrangement, to cancel it. If an administrator performs this operation, the user should also be notified, that their reservation

is about to be cancelled. Consequently, this notification introduces an extension to the changing status notification, in order to deliver this information. Based on the intended use, the *Cancel Reservation* function permits an appropriate integration of this notification.

6.2.4.5 Reservation Unmet

Adapted from the same idea as the *Reservation Cancelled* message, this kind of notification enables the system to inform the user about unwanted behavior. When a user fails to arrive within the designated time frame for their reservation, the *Free Reserved Connector* process usually takes care of these reservations. However, given the damaging effects of such a behaviour, like blocking a charging session that could be used by another user, and the associated loss of profit, when combined with the charges for the reservation. To reduce the occurrence of such actions, the system should send a message advising the user that their current booking is no longer viable according to this process. Furthermore, it should be accompanied by a warning, to remind the user to cancel a reserved session, if he or she no longer demands it, to cultivate more altruistic behavior. As previously mentioned, the *Free Reserved Connector* enables the proper use of this notification type.

The integration possibilities and application domains addressed in this chapter are derived from the current state of the standards used in this thesis and are based on the considerations and design principles available at this stage of the design work. As these functionalities and features are evolving, their use could be extended to other areas and related systems.

7 Implementation

To build on the design of the processes in the previous chapter, the actual implementation of the corresponding functionality within the applications of the underlying solution is presented. First, some preliminary work is considered, using the standard functionality for reservations according to the OCPP standard in version 1.6. This foundation allows the extension to encapsulate these functionalities in order to ensure compliance with the standard. After completing the initial development stages, the proposed design for the entities and processes is translated into code. This is succeeded by an analysis to identify the packages and components that the extension must use to ensure its operability. Considering the system roles mentioned in Chapter 3, the assignment of the respective rules, the allowed scopes, and the respective functionality are taken into account. Furthermore, to enable access through client applications, the implemented functionality is mapped to specific endpoints, representing resources according to the underlying REST principles described in Subsection 2.3.1. A more detailed insight into these resources, in combination with their behavior, is offered at the end, highlighting the interaction between the components to satisfy the preconceived logic.
Concerning solely the creation and management process, emphasis is given to the development of the parts, that constitute the logical units of the process. While the frontend logic is vital to illustrate the user interaction with the backend service and guide them through the procedure, the corresponding implementations are only shown as the established interfaces based on the pre–existing designs in the applications. This bridges the gap between the mockups created in terms of their influence on the final results. Hence, they are included as an attachment to each specified capability. The same applies to the frameworks and programming languages used. To provide a technology–agnostic approach, the specific characteristics of the technologies in terms of implementations are not addressed. However, for reasons of completeness, the frameworks and programming languages used are noted below. *Node.js* framework [52] is used for the backend development and the charging station simulator, *Angular* [5] for the web frontend and *React Native* [65] as a cross–platform framework for the mobile application. The preferred programming language is *TypeScript* [37], which serves as a type–safe extension of *JavaScript* [36].

7.1 System Prerequisites

In terms of standards compatibility, there were issues with the backend, web frontend, and mobile application, as they did not support the *ReserveNow* operation of OCPP version 1.6, nor did they support the *Cancel Reservation* operation either. Additionally, the CS simulator used for local simulations of the test environment also lacked this functionality. To establish a common baseline, as mentioned in Chapter 4, the first step of the implementation phase,

was to align the current implementation status with the predefined operations in the relevant standard. As a result, the necessary entities and feature toggles were implemented and verified against the processes, outlined in the OCPP standard documentation. Not included in this study is a comprehensive explanation of the development steps, required to adjust the applications, which is assumed to be a necessary prerequisite for the topic covered in this study. Although other standards, such as the OCPI or OICP mentioned in Chapter 2 and part of the backend implementation, also offer reservation proposals, these are not addressed in this work as well.

7.2 Reservation

Beginning with the implementation of the proposed entity based on the given design. To ensure type safety and to comply with the existing design of the application, using the type–safe system of *TypeScript*, the entities are implemented as interfaces. By employing this technique, this implementation ensures the specific types for their corresponding properties and considers software design principles such as reusability suggested by Gamma et al. in [22, p. 47ff]. In addition to providing extensibility through the potential of deriving from these instances, this interface–driven method enables developers to adapt the entities and combine them into more complex types, covering advanced scenarios.
To implement the dedicated statuses and types, and to associate them with the appropriate stages in their life cycle, the concept of *Enumerations* is used. This ensures secure type transitions, as suggested in the design chapter, and is widely available in most programming languages. An example of such predefined life cycle transitions is provided by the implementation in Listing 7.1, which covers some instances of transitions below.

Listing 7.1: State transitions enforcing the reservation life cycle using *TypeScript*

```
export type ReservationStatusTransition = Readonly<{
  from?: ReservationStatusEnum;
  to: ReservationStatusEnum;
}>;

public static readonly ReservationStatusTransitions:
    Readonly<ReservationStatusTransition[]> =
   Object.freeze([
     { to: ReservationStatus.IN_PROGRESS },
     { from: ReservationStatus.SCHEDULED, to: ReservationStatus.IN_PROGRESS },
     { from: ReservationStatus.IN_PROGRESS, to: ReservationStatus.CANCELLED },
   ]);
```

The resulting class diagram reveals the final implementation of the classes within the codebase, which, in addition to the `Reservation` interface, includes the connections to the other parts of the resulting entity. Alongside the previously mentioned `ReservationStatus` and the `ReservationType`, the diagram in Figure 7.1 shows the other relevant interfaces. For example, the `ReservationAuthorizationActions` as part of the internal privilege control system, ensuring access to specific operations and entities through `AuthorizationDefinitions` and their according `AuthorizationActions`.

Thus, these interfaces, along with the `CreatedUpdatedProps`, could be categorized as administrative properties, to provide uniform handling of system entities internally.

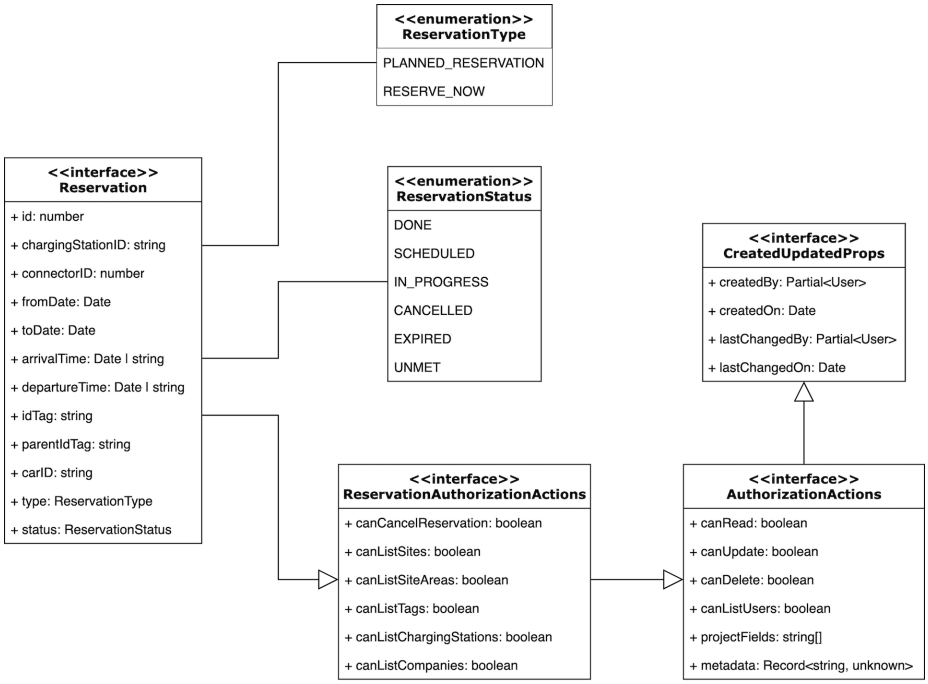


Figure 7.1: Class diagram for the elaborated entity and its associated classes.

Regarding the proposed associations with the included interfaces, the resulting implementation is equivalent to the design entity relationships, shown in Figure 6.1, from the previous chapter. Thus, no further class diagram illustrating the quantity of each entity within the relationship is necessary in this section. According to the `Reservation` interface as the foundation for further development, the application proposal is the next stage of this part of the thesis.

7.3 Reservation System

In order to take up the groundwork provided by the construction of the `Reservation` entity and the process design, the development of the reservation system according to the gathered work is next. For a better understanding of the application serving as a baseline for the elaborated extension, the system itself is analyzed, by considering different views on the system and its modules. As a result, different views out of the context of software architecture are used, to facilitate interested readers of this work to come up with an own implementation, utilizing the provided design approaches.

7.3.1 Architectural Views

Starting with a high–level view of the existing system architecture, to identify the existing components and interfaces incorporated by the backend. For this reason, the module view

of the backend system, shown below in Figure 7.2, provides a detailed insight into the system itself, to discover the standards utilized and their associated functionalities, which are implemented as dedicated servers. To present this highly abstract point of view, these modules are represented as functional units in terms of the decomposition style described by Clements et al. [11, p. 67]. This architectural style is frequently used in order to divide the system into units of implementation that enable better structuring. Regarding the assumptions of this work, the most significant modules and interfaces including the REST server, the **WebSocket** interfaces, and the OData interface, due to its growing importance in the business world and for the purpose of completeness. In respect of the prefix used, each server responds to a different type of communication protocol to provide a specific set of features. Enabling several clients to use the REST protocol for communication purposes, the `REST Server` allows data exchange based on HTTP requests, facilitating the REST principles, described in Subsection 2.3.1. For this purpose, this server encloses the functionality of the subsequent servers, using so–called REST resources to provide homogeneous entry points, that the clients could use to communicate with the servers behind it, without knowledge about each specific resource. In the case of the **WebSocket** interface, the backend system provides a reliable way to exchange data in real–time, using the **WebSocket** protocol mentioned in Subsection 2.3.4. Primarily, this interface is used to communicate with the managed charging infrastructure, including directly accessing the `OCPP Server` for the administration of CSs. The `OData Server`, with the inclusion of the **OData REST Interface**, allows OData clients to access the backend functionalities. By enhancing the capabilities of the standard REST protocol through publishing a public description of the offered services, in the form of a description document similar to the SOAP WSDL, it provides an alternative for utilizing the services in more advanced scenarios.

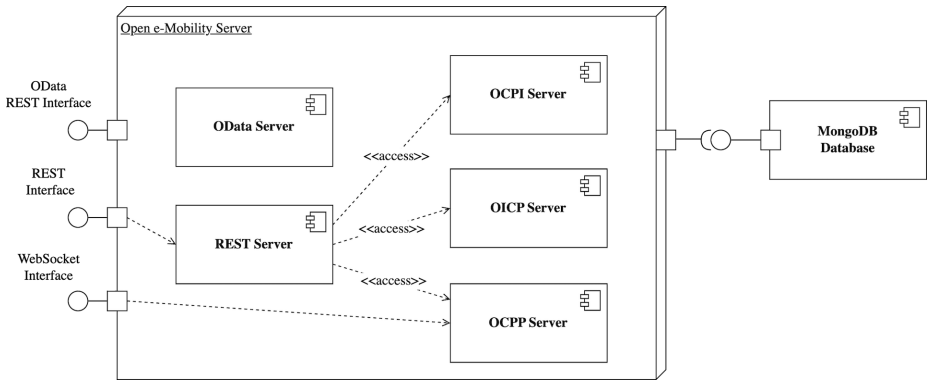


Figure 7.2: Overview of the various functional components within the existing project.

As indicated by Figure 7.2, and due to the reliance on the targeted client applications, which primarily use REST based communication, the main development work is located within the `REST Server`. Besides its role of mapping functionality to the outside world, the corollary of integrating the reservation system inside this server is considered trivial. For a deeper understanding of the internal structure of this server, and to reduce the architectural height to identify the relevant components and packages, for the required

changes to implement the extension accordingly, the following sub–parts drill down to provide a more detailed view.

7.3.1.1 Packages

After the choice of the `REST Server` as the location for the functionalities, the introduced architectural style of decomposition, allows for the arbitrary adjustment of the perspective into which a system and its modules could be further subdivided. As part of Figure 7.3, the utilized extraction process of relevant modules into packages that structure the application is displayed. This view not only shows the logical structure of the project but also allows developers to group together functionality or logical units with a high degree of cohesion in a particular location. As introduced in the previous section with reference to the class diagram in Figure 7.1 for the reservation type, the entity, also known as a type, can be located within the `types` package. The `requests`, which represent pre–defined request bodies for the request `validator` instances, sanitize incoming payloads before processing. Following this naming convention, applicable to all types of implementations in the application structure, the corresponding packages could be easily identified. So the appropriate implementations for `router` and `service`, to provide the REST resources, as well as the `storage` classes for database access, can be readily determined. Also worth mentioning is the mapping of permissions for the existing roles within the `authorisation` package, the addition of `notification` types to notify the user, and the integration of new `tasks` to provide scheduled background jobs for autonomous reservation processing. However, all of these artifacts and their associated changes are addressed in the following sections and are presented solely for awareness of their existence.

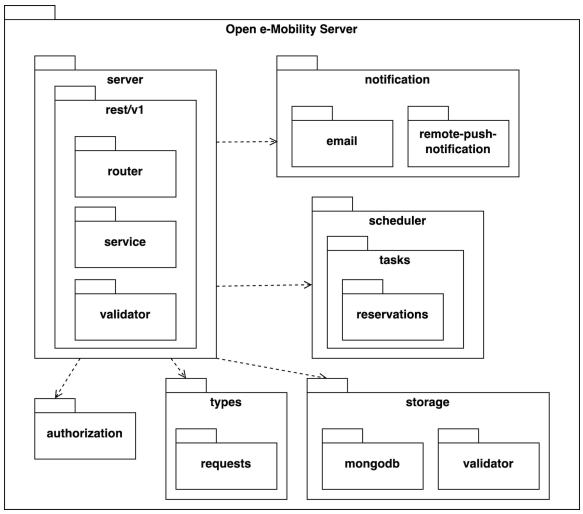


Figure 7.3: Overview of the extended packages within the targeted section of the backend project.

With the structure of the service and the location of the adaptations in mind, the next step is to introduce the components and their associated connectors. Architecturally, the 'component–and–connector views (C&C)' [11, p. 136] are used to show the inner workings

of the system and the interaction between the individual components. Concerning the follow–up implementation, this aspect could be used for orientation and guidance in the project and provide a starting point for further development tasks, to locate the changes introduced by this work.

7.3.1.2 Components

Before elaborating on the behavioral interaction between the components, all the implemented aspects have to be mapped in the context of the designed system. Thus, Figure 7.4 provides an overview of the added components and their integration with the already existing ones. In line with the C&C view, the connectors indicate the available and required functionalities and establish the appropriate connections between the corresponding components, resulting in the reservation system as part of the existing service landscape. For the purpose of abstraction between the existing implementations and the extension, this work treats the `Reservation System` as an independent subsystem that requires only a restricted number of dependencies to other areas of the application. Considering only the information required from other application domains, the portability as a self–contained system to other scenarios should be guaranteed.

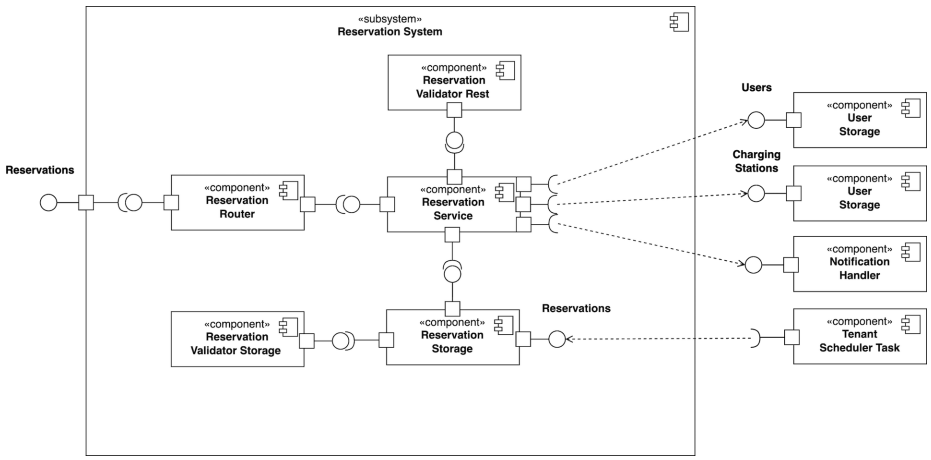


Figure 7.4: Overview of the individual components inside the extended web service.

Taking the design principles of the initial developers and the previously mentioned package structure into account, the `Reservation System` must contain several classes necessary for the RESTful web service implementation. This includes the use of a `Reservation Router` that manages incoming requests of client applications and allocates them to the relevant methods offered by the `Reservation Service`, determined by the requested resource and the parameters in the URL. In the majority of cases, data is requested, which is provided by the `Reservation Storage`, encapsulating the access to the database and maintaining the lifetime of the corresponding connection. To verify the entities arriving with the HTTP request bodies that are to be persisted in the database, the `Reservation Validator Rest` after receiving the requests and the `Reservation Validator Storage` before further processing in the database, form the first layer of validation, to prevent malformed input. Finally, the

`Notification Handler` and the `Tenant Scheduler Task` represent pre–existing components utilized by the `Reservation System` for supportive purposes and are not an explicit part of the system itself. According to its name, the `Notification Handler` is responsible for dispatching notifications to the relevant user in certain cases and the `Tenant Scheduler Task` coordinates background operations. By guaranteeing an entryway for external components and systems that require reservation capabilities, the `Reservation` interface, which points outside the system, allows access to the implemented functionalities inside. On the basis of the elaborated components and their location within the architectural construct of the application, the next subsection continues to illustrate the relevant information flow and interaction to achieve the capabilities in relation to the design work.

7.3.2 Management Capabilities

Combining the designed processes and the aforementioned architectural views, the next parts describe the actual functionality implemented within the system, while keeping the implemented components in mind. Beginning with the management capabilities for the respective entities, which include creation, update, deletion, and cancellation offered as the basic operations offered by this functional module. This also incorporates the administrative capability to enable the reservation extension. In order to gain insight into the implementation of the backend logic in the frontend parts, as well as the comparison with the previously designed mockups, this theoretical perspective is complemented by the final implementation in the frontend applications of the system.

7.3.2.1 Create Reservation

Beginning with the implementation of the **Create Reservation** functionality shown in Figure 7.5 and executed every time receiving a request for creating a reservation from an actual user at the `ReservationRouter`. Delegated to the `ReservationService` by calling the `handleReservationCreate` method, the `ReservationValidatorRest` immediately validates the provided input and stops the execution and the further processing of the entity by detection of malformed or erroneous properties. In case of a successful validation and the non–existence of a record with the same ID, the system checks for collisions with other bookings in the selected time range to prevent overbooking of a CS. After fulfilling all the required conditions for making a reservation, the selected CS is contacted in case of a *ReserveNow* reservation or a scheduled booking with an upcoming timeslot, and the entity is persisted through the `ReservationStorage`, after a final validation by the `ReservationValidatorStorage`. Considering the scenario, if a record with the same ID exists, this design permits the existing reservation to be updated by conformity with the deposited RFID tags as well. A mismatch otherwise results in the execution being aborted to prevent another user without the required privileges from adjusting the reservation. Finally, a confirmation notification of the successfully created arrangement is sent to the concerned user.

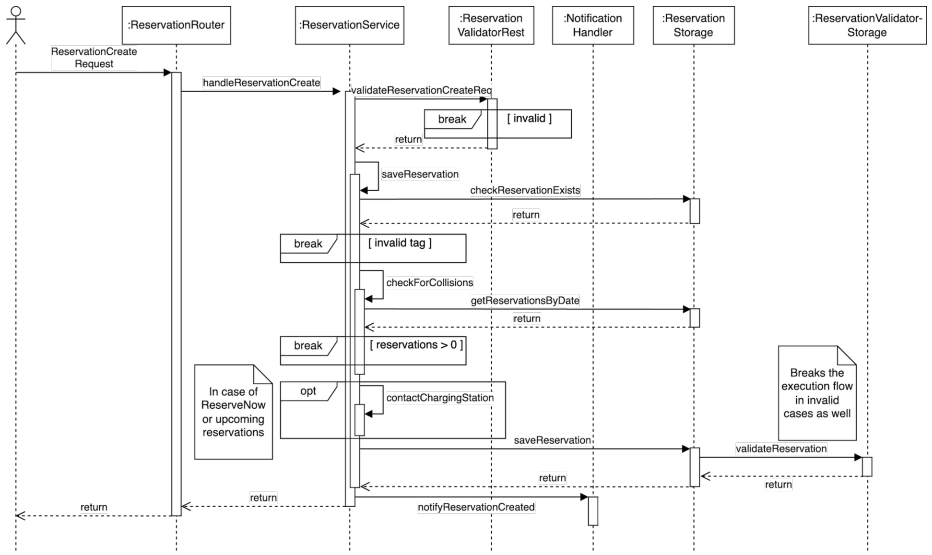
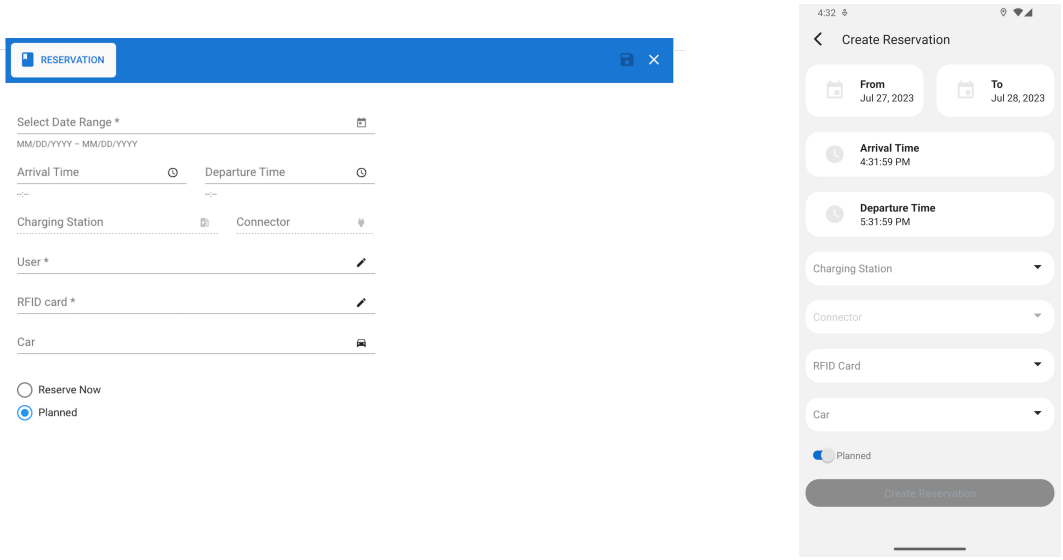


Figure 7.5: Component interaction for fulfilling the process flow of creating a reservation in reference to the proposed design.

With the required booking properties in mind, the corresponding input screens of the mobile and web applications resulted in the subsequent layout shown in Figure 7.6. As well as mapping the relevant properties to the input fields on the user interfaces, the existing layout plays a crucial role in the proposed appearance of the result. For supporting both types of reservations elaborated in this work, a selection toggle is implemented, which appears when the proposed extension is enabled for the respective tenant and automatically adjusts the displayed options on behalf. In combination with an integrated validation of the input fields, the approach to provide one central interface as an entry point for creating the reservation should allow for maximum convenience for the user. Another aspect to mention is, that the selection of a `Parent ID Tag`, representing a superior RFID tag for a group of subordinate tags, is not considered in this particular implementation. This simplifies the handling of the data model and enables a one–to–one relationship between the user and the reservation. Although the input option is absent in the frontend, the associated property indeed exists in the underlying entity.
Delivering this functionality using the REST methodology and the appropriate resource, the **Create Reservation** process could be called on behalf of the following endpoint listed in Table 7.1.

Table 7.1: REST endpoint for reservation creation.

Resource Identifier	HTTP Method
`/reservations/`	`POST`



(a) Create a reservation in the web application.

(b) Create a reservation in the mobile application.

Figure 7.6: Implementation of the user interfaces of the mobile and web application relating to the creation of reservations.

7.3.2.2 Update Reservation

Despite the similarities to the previously mentioned creation process in relation to the details covered, the **Update Reservation**, pictured in Figure 7.7, typically expects that the arrangement the user wants to update already exists. In conformity with its logical twin, the user request passes through `ReservationRouter` and is forwarded by the `handleReservationUpdate` method to the `ReservationService`, which implements the required update logic. Then, `ReservationValidatorRest` performs the initial input validation, before querying the reservation associated with the provided ID through `ReservationStorage`. Because of the aforementioned overlaps between the creation and updating process, no further description of the other steps undertaken is given in order to avoid repetition, which results from the fact that the system updates reservations in a manner, comparable to the CS outlined in the OCPP standard. As indicated there, an existing entity with the same ID is replaced by any incoming *ReserveNow* request with the same ID. Such a flawed design could allow the cancellation of reservations made by other users, who have the same ID as the existing object at the station, which should be prevented. So, the proposed system approach implements a check for this uncommon but potential scenario. Relating to future work, this behavior should be observed, as it is still included in the latest OCPP version 2.0 specification. In order to handle alterations to the CS or connector, the updating procedure guarantees the revocation of the booking at the previous station and gets in contact with the new station accordingly.

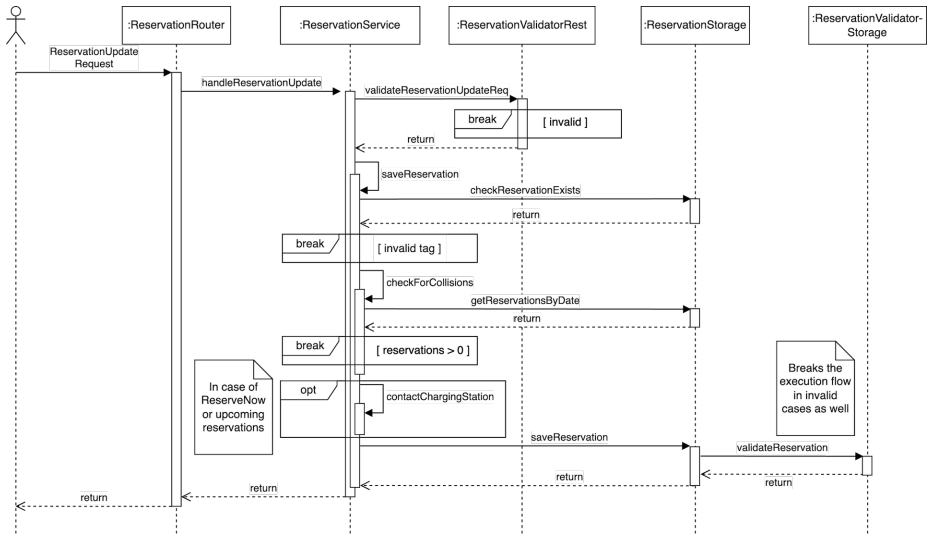
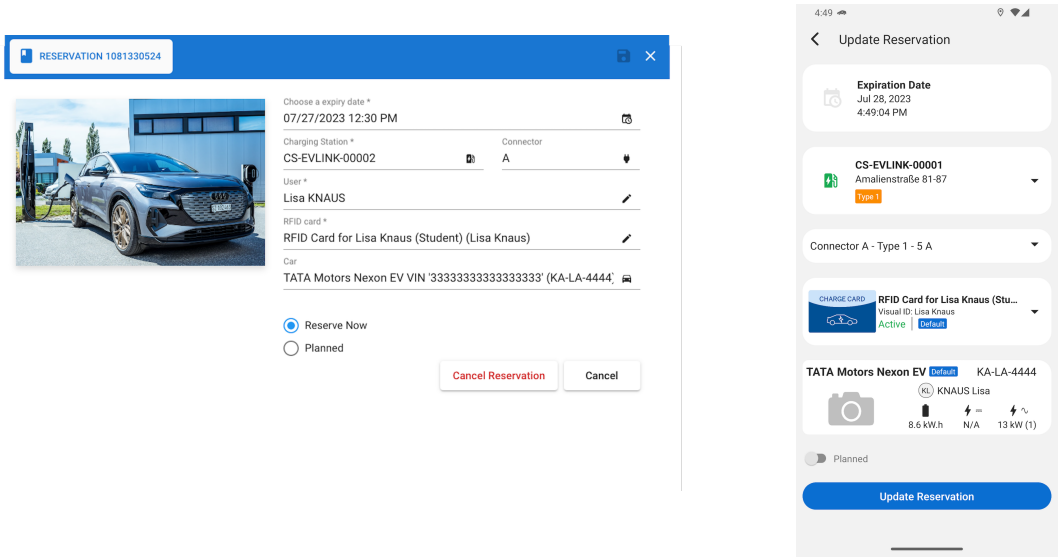


Figure 7.7: Component interaction for updating a reservation considering the proposed design draft.

Because of the parallels in the underlying process, the interfaces for facilitating the update of an entity, include the same input choices as the interface designed for creation. The sole variation within the browser application is the ability to cancel the booking within the update mask using the auxiliary **Cancel Reservation** button shown in 7.8a. This feature has not been integrated into the mobile app for reasons of limited screen space and to minimize the amount of information displayed to avoid cluttered screens.
For the interaction with the relevant client apps, the **Update Reservation** process could be called on behalf of the following REST interfaces listed in Table 7.2.

Table 7.2: URL with path parameter for initiating an update using the respective REST resource.

Resource Identifier	HTTP Method
`/reservations/{id}`	`PUT`



(a) Update a reservation in the web application.

(b) Update a reservation in the mobile application.

Figure 7.8: Implementation of the user interfaces of the mobile and web applications to update a reservation.

7.3.2.3 Cancel Reservation

Derived from the **Update Reservation** procedure described in 7.3.2.2, the **Cancel Reservation** operation enables the requestor, the revocation of a scheduled or already ongoing arrangement as illustrated in Figure 7.9. Like its predecessors, the initiation of the downstream logic in the `ReservationService` undergoes the initial validation checks after request delegation by the `ReservationRouter`. This ensures, that the incoming inquiry aligns according to the defined criteria within the solution and mitigates unintentional behavior, resulting from malformed user input. By not ending the process prematurely due to erroneous data, the actual cancellation functionality within the service method `cancelReservation` is initiated. Starting with the retrieval of the record from the database using the provided ID, the validation of the life cycle transition of the current status, as discussed in 6.1.2, follows. This ensures that only those objects that are suitable for cancellation, can be cancelled by the user and the system, otherwise, the process is terminated. If the scheduled arrival time is already reached, the corresponding CS is contacted, resulting in an immediate cancellation at that station. After updating the status to `Cancelled`, in the case of a cancelled reservation, and successfully validating it once more, the entity is saved to the database again. The act of withdrawing a booking concludes with an outgoing cancellation notification from the `NotificationHandler` to the appropriate system user.

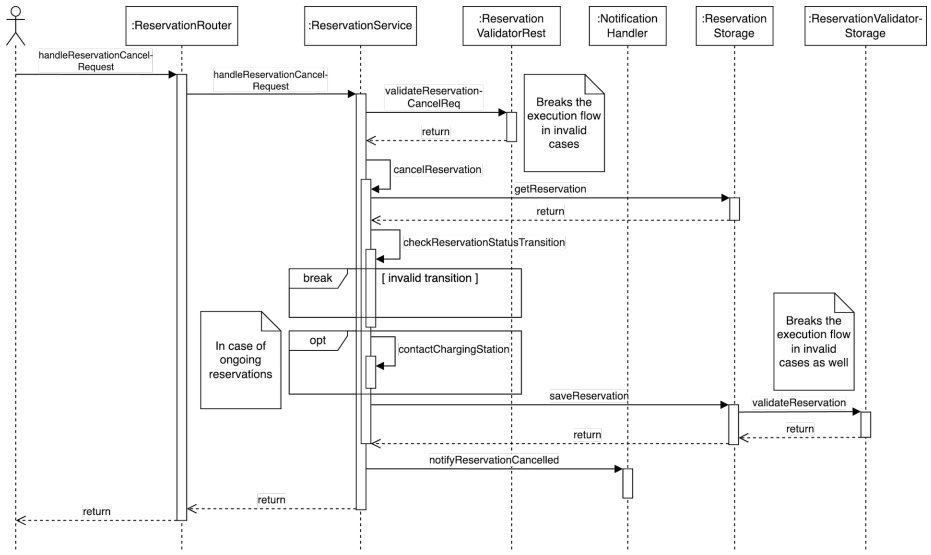
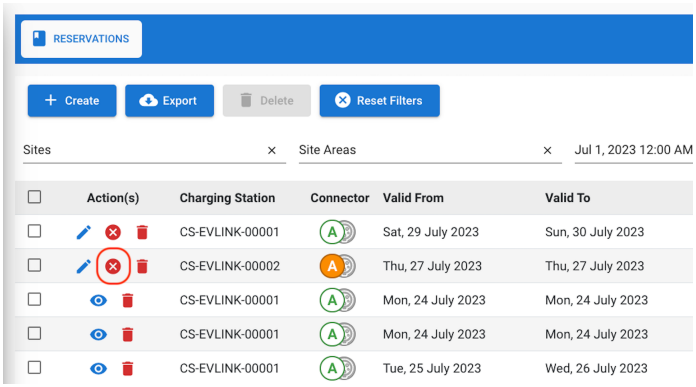


Figure 7.9: The engagement of the different components during the cancellation of a reservation as proposed in the design.

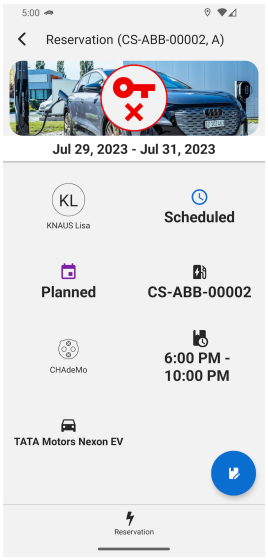
In terms of its impact on the entity and its existence within the application life cycle, the deletion of a reservation is more invasive than cancellation, which represents a practically identical atomic function, that could be realized through one single button. This thought led to the conclusion that such a button should be placed on each booking item on the relevant screen. Accordingly, these buttons should indicate that pressing them results in the associated cancellation of that particular booking. In contrast to the mobile app, the webpage version permits the possibility of withdrawing already within the list overview. Enabling the user to interact more quickly with the appropriate entities, based on their current role and the assigned sites. An example of the list view with the quick action buttons displayed at the list item level is shown in Figure 7.10a, including an indicator pointing to the relevant button.
To mark a booking for revocation, the related functionality could be executed using the REST methodology on behalf of the following endpoint in combination with the matching ID as a path parameter in the URL.

Table 7.3: Provided REST endpoints.

Resource Identifier	HTTP Method
`/reservations/{id}/cancel`	`PUT`



(a) Cancel a reservation in the browser version of the app by using the ⊗ icon.



(b) Cancel a reservation on the mobile version from the details view of the booking.

Figure 7.10: Implementation of the user interfaces of the mobile and web versions relating to the cancellation operation.

7.3.2.4 Delete Reservation

Extending the **Cancel Reservation** functionality, **Delete Reservation** not only cancels the reservations that are currently ongoing within the system or the CSs, it also removes them from the database entirely. As depicted in Figure 7.11, when a deletion request is received from the user at the `ReservationRouter`, the `ReservationValidatorRest` validates it again and returns the process execution to the `ReservationService`. The latter continues the deletion process, once the data entered is correctly validated. With respect to the attached ID, that denotes the booking the user intends to delete, the `ReservationStorage` retrieves the entity from the database and thus verifies its existence this way. In addition to verifying the existence of the record, the current booking status is assessed to initiate additional measures for managing ongoing reservations. This includes preventing the deletion of bookings in progress without notice, for example. To manage ongoing appointments, the system contacts the relevant stations by using the `contactChargingStation` method to release the respective CSs and connectors. Due to the lack of a specific delete operation in the OCPP standard, the deletion encapsulates the cancellation operation for a standardized communication with the CS. As the station does not possess the ability to store history, the cancellation could be described as equivalent to a deletion at the charging infrastructure level. After resolving the dependencies of the booking with the infrastructure, the `ReservationStorage` ultimately deletes the record in the database.

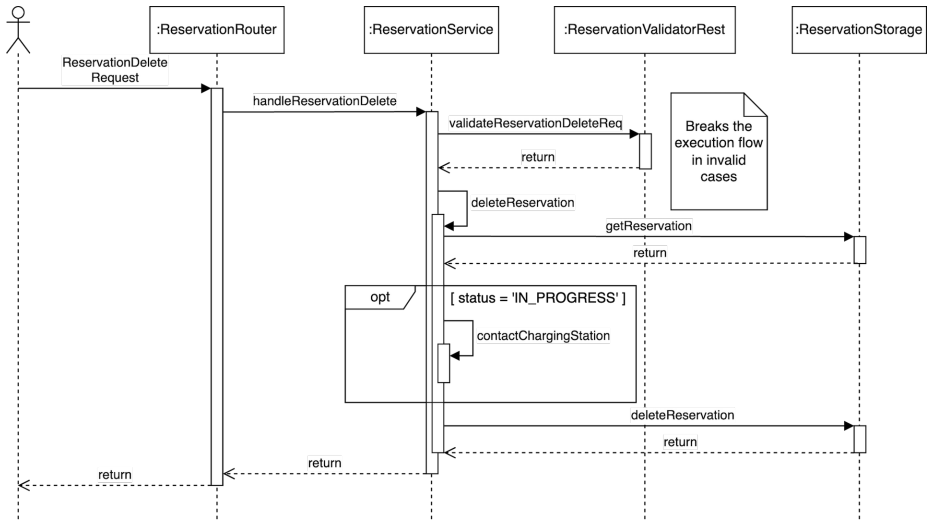
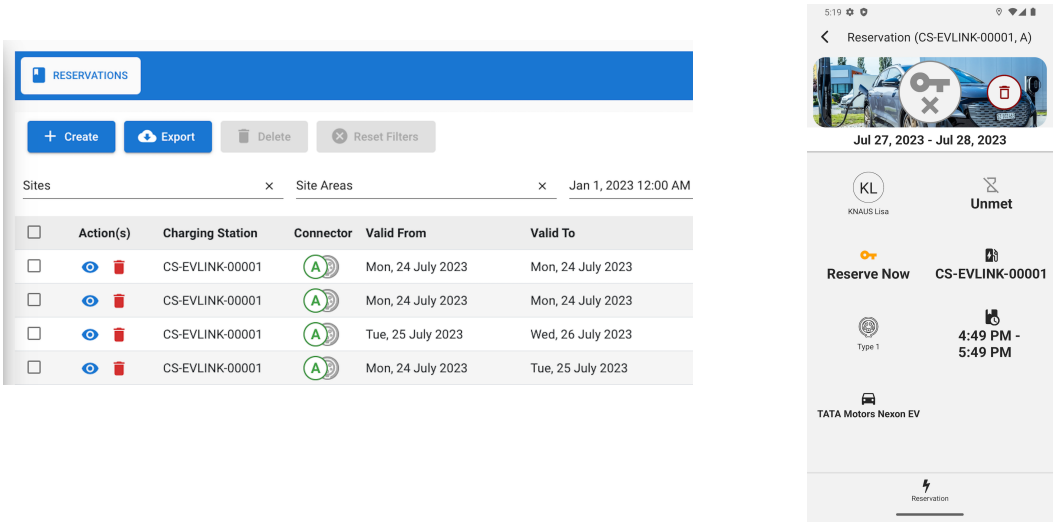


Figure 7.11: Component interaction when deleting a reservation record considering the design drafts.

As well as the referred counterpart, the delete operation, is also designed in the form of a single button. It is included within every screen showing a particular booking, enabling the user to delete at least one entity inside the system. In terms of a larger GUI for interaction, the browser app allows the user to delete multiple reservations at once, as indicated by the provided REST endpoints in the Table 7.4. With reference to further development for the mobile versions, it may be feasible to implement a multi–select interface to enable the inclusion of multiple list entries within a single delete request.
To allow both atomic and batch deletion of entities, the REST methodology recommends introducing the endpoints listed in Table 7.4. For distinguishing between both operations, in order to delete a single object, the user must reference it by its ID. In the case of deleting multiple entries, the user has to submit multiple entries, omitting the ID path parameter.

Table 7.4: REST endpoints for deletion purposes.

Resource Identifier	HTTP Method
`/reservations/`	`DELETE`
`/reservations/{id}/`	`DELETE`



(a) Delete a reservation in the browser.

(b) Delete a reservation on the hand–held device.

Figure 7.12: Implementation of the user interfaces of the mobile and web application relating to the deletion of reservations using the 🗑 icon.

7.3.2.5 Enable Reservations

Representing the final predefined management capability established during the preceding design phase, the **Enable Reservations** function enables the system administrator to activate the extension developed in this work. This feature is part of the `Tenant Component` modules, which stand for different feature sets, the tenants can be enabled for. Due to the varying requirements and challenges faced by each organization, as identified by their dedicated tenant, the range of usable functionalities differs significantly. Which only requires subsets of the system's feature sets to be enabled simultaneously. However, certain dependencies exist between the various components, which necessitate activating specific ones, in order to enable the usability as a whole. The reason for this is that each module determines the amount of data and information available to the tenant, therefore, activating a certain group of components is logical. For example, the tenant component `Reservation` depends on the corresponding `Organization` component to get access to the CSs on the different sites of the organization and the relevant users. Otherwise, the system denies to enable this component on itself. Components such as the `Car` component may be activated upon request, to offer a wider range of features to the particular component. In the case of the `Reservation` component, this enables the integration of cars into the reservation. The example configuration within the administration area of the web frontend for enabling components is displayed in the following Figure 7.13.

If the `Reservation` component is not active, the adjusted solution provides solely the basic OCPP reservation feature set, using the standard *ReserveNow* and *Cancel Reservation* operations [54]. This does not include the extension of recurring and reservations in advance, as discussed in Section 7.1.

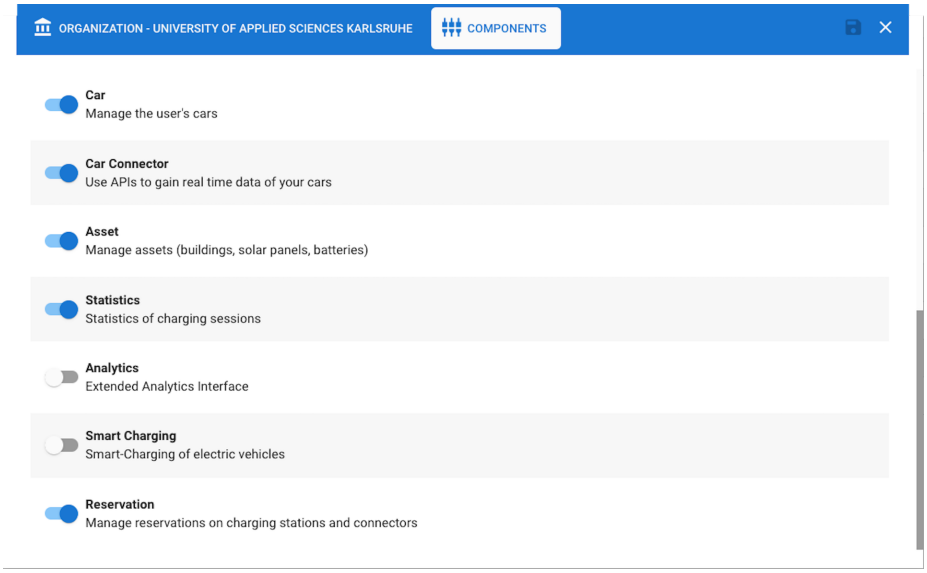


Figure 7.13: Implementation of the user interface for enabling the reservation module.

7.3.3 Scheduling Capabilities

Complementing the management capabilities covered in the previous Subsection 7.3.2, the ability to schedule and process pre–configured tasks automatically without any interaction is a key aspect of developing a system that manages distributed infrastructure facilities. To maximize automation, the proposed scheduling capabilities comprise a set of batch processing jobs, that are intended to run sequentially and can be enabled on demand within the JSON configuration file `src/assets/config.json` included with the project. In addition to task activation and ensuring the specified sequential order, this file also allows for specifying the execution intervals in the form of `*/1 * * * *`, employed as the syntax for Cron jobs in the *UNIX* world, to schedule job processing at specific times. For the execution to behave as expected, the reservation component module introduced in 7.3.2.5 must be enabled. Otherwise, regardless of the activation, these tasks are not running. The pivotal function for handling these tasks is represented by the `TenantSchedulerTask` already mentioned as one of the key components for the working system in 7.3.1.2. As the top–level entity overseeing all available tasks for execution, it executes the activated ones at their designated time intervals on the server. The recommended sequence for executing the subsequent jobs to ensure correct operation is defined as follows within the context of this work:

Schedule Reservation:	`*/5 * * * *`	*'Executed every five minutes.'*
Expire Reservation:	`*/4 * * * *`	*'Executed every four minutes.'*
Free Reserved Connectors:	`*/1 * * * *`	*'Executed on a minute basis.'*

7.3.3.1 Schedule Reservation

Due to the fact that the user can create reservations in the future, the corresponding CSs and the system itself must be notified when the reserved time period arrives.
To do this, the **Schedule Reservation** task shown in Figure 7.14, implemented as `SynchronizeReservationTask`, loads all reservations with status `SCHEDULED` for the next 15 minutes from the database. Also, to enable recurrent events, this comprises bookings with the status `IN_PROGRESS` as well. In the further processing of the combined booking types, the `synchronizeWithChargingStation` method checks for currently running charging sessions on the respective stations and connectors. If these sessions are not assigned to the RFID tag, that belongs to the reservation, the method stops them immediately. To address this, the `ChargingStationClient` is used, which is a product of the `ChargingStationClientFactory`, that controls instantiation using factory methods, to communicate directly with the CS. Otherwise, the reservation is made directly on the CS and if it has a `SCHEDULED` status, it is adjusted accordingly. A notification is then sent to the user, via the `NotificationHelper`, to inform that a booked charging session is forthcoming. Besides saving the updated record, its ID is assigned to the associating connector on the CS in the database, using the `updateConnectorWithReservation` method.

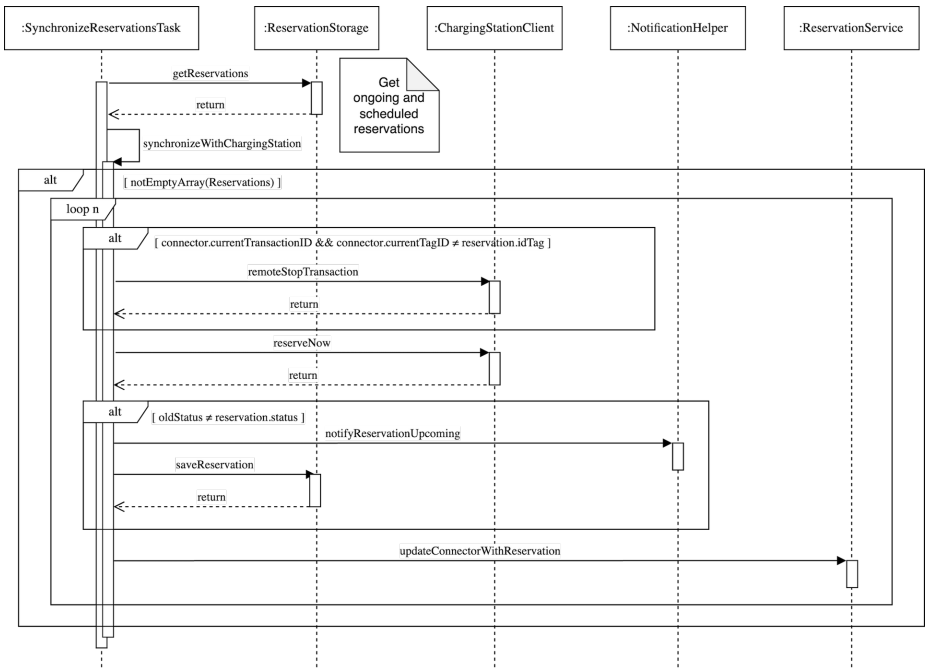


Figure 7.14: Interaction between relevant components to synchronize the reservations with the associated CSs taking into account the underlying process design.

7.3.3.2 Expire Reservation

After having automatically synchronized the reservations with the corresponding CSs, a routine to clean up the expired ones and to fulfill certain aspects of the householding tasks is implemented as `CheckReservationStatusTask`. Designed as a counter piece to **Schedule Reservation**, this job deals with entities reaching their expiration date, as demonstrated in Figure 7.15. To identify the concerned bookings, it loads each record with a status of `IN_PROGRESS` or `SCHEDULED` and an expiry date that is overdue by the current time and date. If any record is found, the task then iterates through the list of reservations and removes the reservation information from the associated connector, using `resetConnectorReservation`. As a complement to the `updateConnectorWithReservation` function mentioned in the previous process, this method is used when an arrangement still blocks a connector after it is successfully cancelled or deletion. This is often due to the system disconnecting from the CS during the latter stages of the process. Considering this case, it is utilized to invalidate expired bookings, change their status to `EXPIRED` in the database, and notify the involved profile with the `NotificationHelper`.

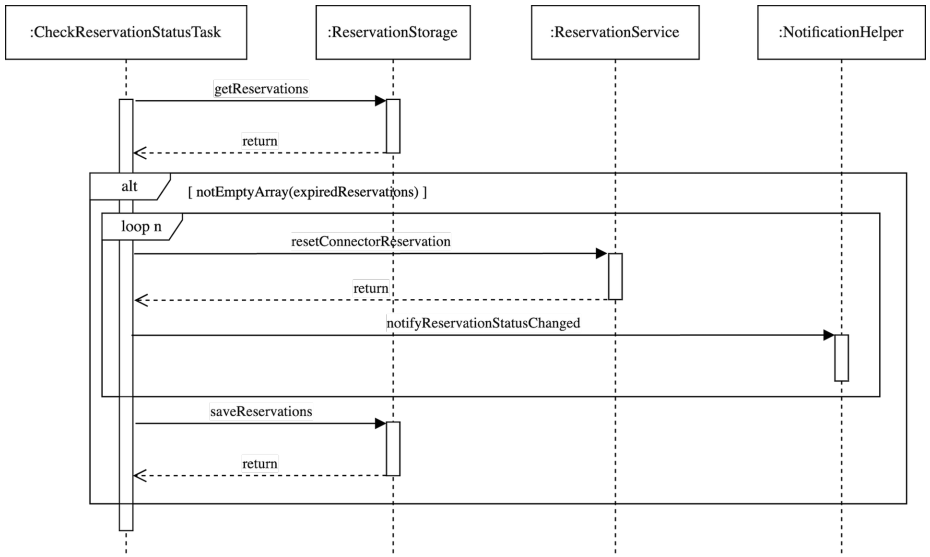


Figure 7.15: Sequential flow of the components interacting with each other, to invalidate expired bookings using the considered process templates.

7.3.3.3 Free Reserved Connectors

While the **Expire Reservation** function looks after the expiry as a natural element of the booking life cycle, the **Free Reserved Connectors** task has a more regulative purpose. Taking the position of an imaginary steward, it monitors all ongoing engagements on the CSs and identifies the ones, where the user has not arrived after a certain period of time after the supposed arrival time. Following the OCPP standard, the arrangement lasts until the expiry time is reached. As the expiration time is undefined, it could be several hours later and block the station during this time. To mitigate this issue in the handling of reservations, the proposal provides a background task, implemented in `CancelUnmetReservationsTask` and described in Figure 7.16. As a result of this custom implementation, it is possible to handle unfulfilled reservations, due to a certain threshold defined by the constant `THRESHOLD`, which is set to **15** minutes in the context of this work. Every booking with an arrival time exceeding this threshold is identified by the `ReservationStorage`. If the associated connector is still in the `RESERVED` state, the blockage is cancelled to free the reserved charging unit. Otherwise, the next one is evaluated by using these constraints. After setting all identified entries meeting these criteria to the `UNMET` status and saving them again, the relevant individuals are informed of their undesired conduct.

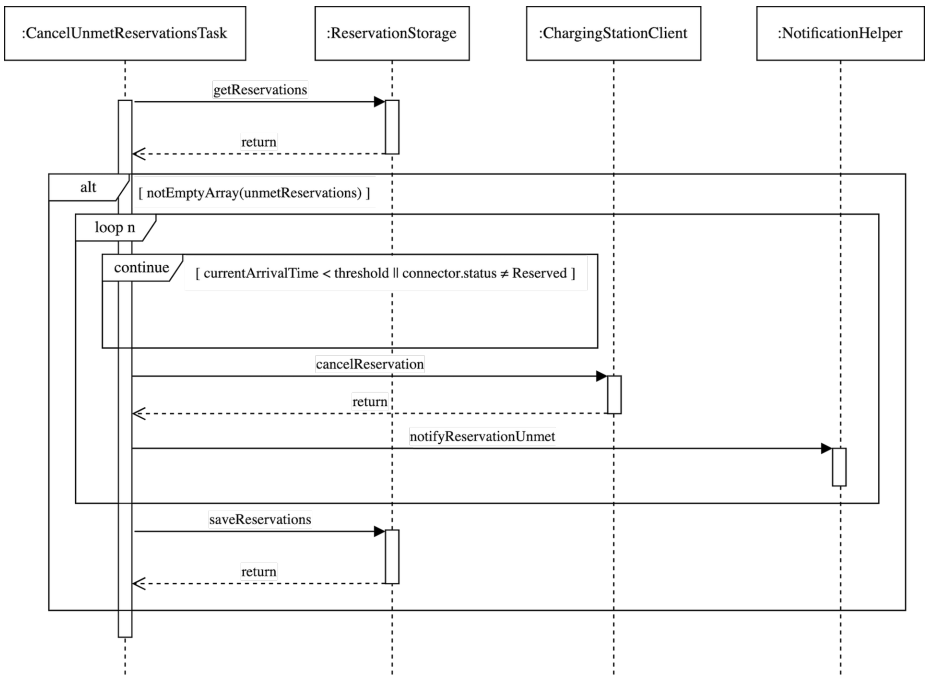


Figure 7.16: Required component interactions for removing unmet appointments on the respective CSs in order to implement the underlying design proposals.

7.3.4 Notification Capabilities

With regard to the notification types, introduced to cover different scenarios in the application and arrangement lifetime, this subsection provides a brief overview of which notification types are actually used within the implemented features. This may serve as a reference point for future implementations and as a possibility to evaluate new scenarios not covered by this implementation. Implied by the listing of involved packages in Sub–subsection 7.3.1.1, particularly by viewing the `notification` package, the system can send notifications in various forms. Alongside using the SMTP protocol [41] primarily targeting adopters of the browser app with assumed easier access to their mail clients, for the mobile apps the service utilizes *Google's Firebase* [19] PaaS and IaaS offer to publish information via push notifications. During the development phase as part of the extension for the mobile component of the project, this work has identified missing functionalities related to background notifications, when the application is closed or not opened. To address this specific feature gap, adjustments to the appropriate logic and the `notifee/react-native` package from *Notifee* [53] are added. Seizing the suggestions represented by **Scheduling Capabilities** in Subsection 6.2.3 of the design chapter, the notification types such as `ReservationStatusChanged`, `ReservationUpcoming`, `ReservationCreated`, `ReservationCancelled` and `ReservationUnmet` are utilized in the current tasks to convey crucial information about the processed entities to the end users and facilitate the information flow. This should ensure that the EVU can make informed decisions based on up–to–date information provided by the system. In the case of the `ChargingStationBlocked` notification, no suitable location could be identified. An integration within the **Schedule Reservation** process would lead to a simultaneous notification of the user blocking the CS, as well as the user of the reservation. Possibly resulting in the parallel switching of drivers to a new station and the situation that this notification was intended to prevent.
In terms of the support for multiple languages both the notifications and the labels of the GUI elements, which normally adapt to the language set in the user's profile, are only translated into English and German. Other language options, including Spanish, French, and Italian, are not part of this approach and further work needs to be done to allow these languages to be used properly.

7.3.5 Additional Capabilities

During development, more ways to enhance the reservation approach with extended features beyond the defined use cases could be identified. These capabilities, since they have no direct use affecting the basic functionality in interaction with the system, are listed as **Additional Capabilities** and are mentioned here as part of an informative section, to possibly be considered as useful for other scenarios, that may arise during other studies in this area.

7.3.5.1 Export Reservation

By examining the backend service features and frontends' specific functions, the potential to export certain entities of the system in the form of generated CSV reports stands out. Despite the fact that it was not considered during the design of the system, the author of the work assumes that this feature could be useful for the end users of the platform in the context of data recovery and analysis purposes. Representing a very basic functionality, which only requires extracting all specified entries from the database and transforming them into a CSV compliant format, no additional process or sequence flow is elaborated. Implemented as a button on the reservation overview interface within the web GUI, the **Export Reservation** functionality loads each booking the user has access to from the database and writes the entries into a CSV compliant file that can be downloaded via the browser. If the database crashes or there is a risk of unwanted data loss, these files could be the basis for populating new instances of the system or recreating a clean application state.
As a dedicated endpoint on the web server, this functionality is provided, using the REST methodology and the relevant resources as part of the service portfolio and is presented in the subsequent Table 7.5.

Table 7.5: Endpoint for exporting the booking entries inside the database.

Resource Identifier	HTTP Method
`/reservations/action/export`	`GET`

7.3.5.2 Reservable Charging Stations

Due to the assumption, that the individuals create their reservations directly by selecting the CSs from the map view, considered as the entry point of the mobile application, or by using the various filter options within the web app, a pre–filtering mechanism to narrow down the options for available stations during a specific time in the future was not included in the design. However, the proposed solution provides a method for making bookings without pre–selecting the CS, which leads to the requirement of choosing a suitable station, while setting the reservation constraints. To promote a more convenient recommendation mechanism, the creation process should be supported by minimizing the number of stations presented, that are not otherwise occupied at the given time. Taking

this into account, the initial concept of the underlying logic resembles a simple intersection, similar to the one shown in Figure 7.17. In this way, the appropriate subset of connectors available for reservation could be easily identified. Hence, the code loads all the entries for the specified period and all the EVSEs registered in the system, in order to perform an intersection of the two sets and select only the stations inside the **blue area**, as shown in the diagram below. These stations comprise the subset of options to be offered by the input fields. If other EVUs create reservations during this process, that would affect the offered selection, the service should detect any resulting overlaps and handle them appropriately as usual.

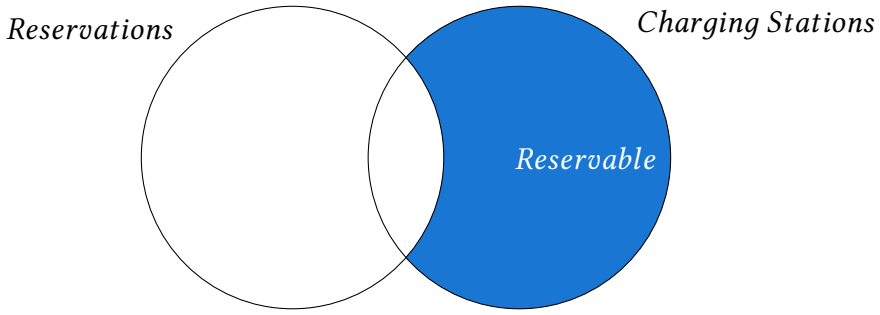


Figure 7.17: Isolation of reservable CSs utilizing an intersection with already reserved ones.

As the client applications must pre–populate input fields in the appropriate GUI windows, this supporting functionality is triggered by the respective endpoints listed in the Table 7.6. To implement this feature, the selected design approach requires expanding the relevant `charging-station` REST resource, which is managed by the `ChargingStationService` and the upstream `ChargingStationRouter`.

Table 7.6: Added REST endpoint to provide the subset of reservable CSs.

Resource Identifier	HTTP Method
`/charging-stations/reservation/availability`	`GET`

7.3.5.3 Reservation Retrieval

Besides the aforementioned feature sets and their related endpoints, a web service following the REST architectural model requires additional interfaces, e.g. for information retrieval. Consequently, according to this approach, endpoints must be supplied for so–called `READ` operations. The function's connotation implies, that these endpoints are accessible for requesting data when calling the corresponding URL. Following the naming convention of the REST method, the resource for collecting reservations is declared, as its plural form of the handled entity name in lowercase. Given the proposed solution and the reservation as the underlying entity, the resource provided by this system is called `reservations`. Resulting in the interfaces listed in Table 7.7 and accessible through the reservation API, mainly employed for providing clients access to the internal system information.

Table 7.7: REST endpoints for performing `READ` operations retrieving reservation entities.

Resource Identifier	HTTP Method	Functionality
`/reservations`		
`/`	`GET`	Retrieve all reservations available
`/{id}`	`GET`	Request for a specific reservation by its ID

7.3.6 Role Concept

To assign the necessary functionality to each system role, as described in the chapter of requirements engineering, particularly Section 3.4, the capabilities required to fulfill their purpose must be considered according to the principles of least privilege, which are explained in [46]. Additionally, to supplement the mapping of feature sets to roles, the solution distinguishes between specific scopes that describe the range of information available to that particular role. Roles assigned with the *Own* scope only have access to their own data and cannot view reservations made by other profiles, for example. That falls within the scope of the *Site*, allowing the assigned role to access all data pertaining to that specific location. With this privilege, the respective user can create, update, or cancel bookings for all users assigned to this site, for instance. An exception to this privilege set is the *Admin* role, which is not limited to a dedicated site and allows the user to manage all entities, related to the entire tenant. Concerning the use cases from Chapter 3 and their associated actors, the final mapping of the developed processes is presented in Table 7.8 as defined in the file `AuthorizationsDefinition.ts` as part of the backend project.

Table 7.8: Mapping of the designated reservation functionalities to the corresponding roles in the system.

Role	Scope	Functionality
Basic	Own	Create Reservation, Update Reservation, Cancel Reservation
Admin	-	Create Reservation, Update Reservation, Cancel Reservation, Delete Reservation
Site Admin	Site	Create Reservation, Update Reservation, Cancel Reservation
Site Owner	Site	Create Reservation, Update Reservation, Cancel Reservation
Super Admin	-	Enable Reservations
Demo	Own	Create Reservation, Update Reservation, Cancel Reservation

Afterwards, the assignment decisions in the context of the *Delete Reservation* and *Enable Reservations* actions are revisited, for a better understanding. The choice to only allow the *Admin* user to perform the *Delete Reservation* action is not solely based on the use case design. This work assumes, that disruptive actions, such as entity deletion, should only be carried out when excluding system users and removing their information, which can be classified as an administrative task and solely limited to system administrators. Concerning basic users, it is more likely that they use the cancel operation to properly revoke their reservations than to delete them completely. In the case of the *Enable Reservations* task, only the *Super Admin* can access the relevant sections of the administration dashboard to configure the tenants, thus, making this role the exclusive entity for assigning this feature. On the other hand, this profile is unable to manage entities within the particular tenants itself, hence the need to assign it additional privileges for managing reservations is mandatory.

7.3.7 Failure Indication

The introduction of new features not only creates new possibilities for the interaction with the audience but also opens up a wider surface for possible unexpected circumstances, that may arise along certain scenarios. Most of these exceptions may occur in a way that only the system can recognize and are typically hidden from the user, who is simply experiencing the consequences, resulting from these malfunctions. To alleviate these uncertainties, a software solution should always inform the user of its current status within the process it performs. This involves notifying the client, on behalf of the user, of any unexpected behavior that could interrupt the current operation or lead to a bad result. Due to the nature of distributed systems, the client and the corresponding server are decoupled, and the only way the server could keep the client informed of the above scenarios is to use the HTTP status codes as part of each request to exchange information. Apart from failures within the system itself in the form of outages or targeted attacks, the charging infrastructure and the associated CSs, should also be considered as an extra source of errors. To effectively identify and manage exceptions in both the backend and frontend applications, this study translated each potential exception into a status code for meaningful reservation error handling.
The list below only reflects the current state of implementation, concerning the aforementioned range of capabilities, as well as the current state of the OCPP standard, which is not meant to be complete. Potentially, additional scenarios and relevant exception–handling mechanisms need to be incorporated in the future.

600 Reservation Already Exists

601 Reservation Collision

602 Reservation Not Supported *

603 Reservation Rejected *

604 Reservation Faulted *

605 Reservation Occupied *

606 Reservation Unavailable *

607 Multiple Reserve Now

608 Invalid Status Transition Error

To distinguish the exceptions that may arise from the charging infrastructure, the respective entries in the above list are marked with the * symbol. For a detailed explanation of these particular error scenarios, please refer to the official OCPP standard [54].

8 Analysis and Validation

Referring to the final design concept and its implementation within the practical part of this work, the proposed reservation system and its capabilities, which are considered necessary for the efficient management of the charging infrastructure, are observed in an evaluative manner and finally reviewed. Therefore, the assumptions formulated during conceptualization, as well as the pre–defined goals and their achievement, are considered to give a reasonable assumption of whether the system is capable of handling reservations on CSs in an efficient manner. Alongside the stated objectives for fulfillment, this chapter discusses open issues, that emerged during the different phases and which could not be integrated into the development process, due to the limited resources available. However, these characteristics are deemed essential to facilitate situations, that would add value to this work. To conclude the capabilities listed and further possibilities for feature implementations, the limitations of such a system, described in the final part of this chapter, are elaborated as a set of challenges in the context of certain scenarios.

8.1 Achievement of Objectives

With the aim of elaborating on a reservation process implemented insight a suitable system for handling the relevant charging infrastructure, this thesis stated several objectives the system needs to satisfy. Formulated in Section 3.5, describing the *Goals* within the preliminary *Design* chapter, the fulfillment of these goals is subsequently evaluated. Beginning with a brief summary of each goal, the capabilities addressed are evaluated in relation to their originally proposed range of functionality. Regarding the regulation and management of entities within a software solution, it requires specific basic operations to carry out the tasks, mainly known as *Management Capabilities*. As stated in the first objective, a reservation system's crucial feature is the ability to manage various types of reservations. This includes criteria to prevent overlaps and collisions, along with the flexibility to modify, reschedule, or cancel individual appointments. Since not every user in the system has the same privileges based on the role assigned to their profile, rules governing access to the range of functionality provided, as well as the associated insight into the data available, are part of this consideration and should be enforced by the system. With access control rules in mind, the concept of *Self–Healing and Autonomous Processes* arising from a set of rules covering the description of a self–regulated behavior that allows the system to handle pre–conceived scenarios, without the intervention of a user, is defined as the second objective. In addition to mitigating undesired actions of individuals, the transition of managed entities, such as changing their states according to specific constraints, is conceivable. Consequently, both the convenience of the system user, who no longer needs to approve and process each step of the entity life cycle and the consistency of the data

could be improved. Beyond the capabilities, which focus on the systems' functionalities and the support of generic functions to enable the user to achieve his goals, the environment and the communication partners, that exchange data with the solution, need to be addressed. Due to the emphasis on the charging infrastructure in the context of e–mobility and the recharging of EVs, the industry and other players in the field have developed a common basis as a guideline for communication between the various entities, involved in the domain. Representing these guiding principles, which include the definition of a basic feature set in combination with the essential steps to achieve successful interaction with the relevant devices, standards such as those mentioned in 2.1.4 need to be considered. Covered by the *Support of relevant standards*, the intention is to provide an interoperable solution that is not restricted to a particular type of infrastructure but can manage all devices that implement the generally accepted standards. After gathering the required functionalities and referencing the relevant standards that the system must consider, the objective of *Modular Design* focuses on addressing the design of the solution in terms of software components and the necessary abstraction from the remaining system landscape. Resulting in the ambition to provide as much decoupling as possible, which promotes both resiliency of the overall system in case of failure, and easier carve–out of functionality for individual service deployments. Furthermore, this permits integration within other services or the potential for reuse within other deployment scenarios.

Considering the above objectives and their relation to the current output, the following conclusions can be drawn with regard to the achievement of the stated targets. It is important to note, that the purpose of this section is not to provide a detailed summary of each functionality and label it as complete or incomplete. Instead, it assesses the level of accomplishment and the overall design intentions, achieved from the perspective of the author of this work, with a view to possible application within a real–world scenario, similar to the one elaborated in 3.1. Taking into account the existing functionality within the system and that provided by the current standards, the implementation and its design proposal extend the feature sets in several ways. Apart from the ability to create bookings for dedicated connectors and the CSs in advance, the functionality is not covered by the solution and is not covered by the standard used. Furthermore, the option of recurring bookings over several days is a newly introduced concept and is still not found in the existing literature, during the processing time of this work. By applying the implementation to the custom scenario and considering its application within a public or semi–public charging infrastructure, the developed capabilities present the chance to reserve vital infrastructure, in an efficient manner. For example, it enables the creation of longer–lasting reservations, suitable for visitors to a particular site, without the unnecessary hassle of making multiple arrangements for the same user, resulting in more convenient use of the applications. This could be further improved with the help of the self–healing and autonomous background processes described later. Assessing the level of fulfillment of the *Management Capabilities*, this work assumes an almost complete coverage of these system capabilities. Despite the absence of features required for use in smart charging scenarios, which are addressed in the following section listing the open issues identified during the development process, the system provides a sufficient set of features in terms of management tasks for the most plausible scenarios. Aside from promoting convenience

in using the system, the *Self–Healing and Autonomous Processes* cover the most essential states within the underlying life cycle concept, introduced by this work. Considered the most important ones are the synchronization with the CSs, due to the upcoming reserved period, the expiration in terms of reaching its end, as well as the handling of the case of a non–appearing driver. However, this list is not considered complete and permits the addition of further scenarios, taking into account the varying charging infrastructures and their respective constraints. From the authors' point of view, this does not allow a direct statement to be made about the degree of completion. As the focus of this collection is to serve as a guide for future work, it should evolve over time to cover more viable scenarios in the future. Regarding the *Support of relevant standards*, the utilized OCPP standard, as previously mentioned, served as the base implementation for enabling the communication with the charging infrastructure and the respective CSs. Besides OCPI, mainly used for roaming scenarios, and OICP, developed as a proprietary protocol, the OCPP standard, as a free and extensible implementation, is assumed to be the most suitable choice for use in systems in the e–mobility context. Due to the protocol's existing implementation in the underlying solution, the decision to use it in the context of this development could be described as mandatory. Though it cannot be excluded, that the other standards could also be applicable to suit the used scenarios and their requirements. Therefore, the designed entities abstract from the templates introduced by OCPP in a manner that allows a switch to another standard without further adaption of the reservation logic itself. Referring to the elaborated `Reservation` entity, a derived set of properties could be used within a more general scenario, in which the mere reservation of CSs is not the sole focus. Thus, it could be assumed that the proposal would be supported in conjunction with other existing standards. Nevertheless, initiating the desired behavior when calling functionalities as part of an existing application or through an interface, provided by a CSMS, requires parameter adjustments. Concluding the assessment of the project's objectives, the *Modular Design* evaluates the architectural metrics, particularly the coherence of the system, as well as the interaction of the introduced components with the existing ones, located within other logical areas of the application. From the very beginning of the conceptualization of the entities, care was taken to keep the dependencies, on other objects within the system, as limited as possible. For this purpose, only the essential relations, such as the reference to the relevant CSs and the RFID tag of the user who created the booking request, are integrated, and are also mandatory, in terms of the underlying OCPP standard. Examining the ability of the system to selectively activate functions, the modular design of the reservation subsystem also allows its complete deactivation by the superior administrator. As previously stated, deactivating this module restricts the system's users to the booking options, according to the established standard. To guarantee this functionality is enforced, the relevant components implement this logical toggle to prevent unauthorized use by tenants, that are not enabled for this operation. In terms of dependencies referencing other components, that are not directly part of the subsystem, only the connection to the dedicated storage classes, for retrieving user and information about the CSs, is established. This includes the integration of the component responsible for generating and sending notifications to the appropriate client devices. Deployments that target smaller service units, commonly referred to as microservices, aim to extract each subsystem to operate independently from others. In this context, only the user and CS information needs to

be retrieved from other service instances. To enable notification capabilities, extra steps must be taken. This led to the conclusion that looking at the logic of the service alone, it maintains only a few connections to external components, that could not be precisely described as part of the design, which resulted in a more loosely coupled system. Nevertheless, these dependencies can be easily abstracted and can be considered as a result of the chosen form of extension.

After outlining the objectives and the corresponding degree of completion, the next section presents features and feature sets in the form of capabilities, that are not part of the current state of the extended service. Despite that, they should be mentioned here for future elaboration.

8.2 Open Issues

Because of the variety of possible ways in which pure reservation systems could be implemented in order to allow their users to book the managed limited resources, not all features and capabilities, could be integrated into the final stage of the application. The aim of this section is to give a comprehensive outline of the capabilities that the actual system does not cover, in terms of managing the reservations themselves, as well as the use of methodologies, that are highly relevant in the context of e–mobility and the use of EVs. It is important to note that this section does not offer in–depth implementation guidance for incorporating these features, but rather presents them concisely for potential inclusion in future work.

In the case of the OCPP standard, the specification enables the use of a parent ID tag, which represents a subset of RFID tags, covered by the superior entity. Every tag belonging to this group is capable of initiating the charging session on a reserved connector. Covered by the underlying *Open e–Mobility* backend [28], only `SiteArea` entities offer the possibility to nest site areas within another and organize them into a ranking, omitting the associated RFID tags. Possibly, the tags within the parent areas may also have the ability to charge at the stations within the specific child areas. However, this behavior could not be confirmed in the current study and was not found to be part of the logic, governing the initiation of the charging session. Resulting in a more detailed examination of the charging process, this capability was not taken into account during the design stage but was introduced as a property within the `Reservation`, for later implementation. As part of this work, the typical user, who represents the majority of the EVUs, only considers creating reservations for himself or on behalf of certain individuals.
Along with the ability to declare a parent tag, to allow all its children to charge on a reserved connector, the CSs facilitate the reservation of an unspecified logical connector, through the usage of the dedicated ID `0` in the booking request. This requires the CS to possess at least one connector, accessible for charging, within the designated time slot. Despite implementing this feature during the extension of the CS simulator in developing the system prerequisites, the actual frontend applications do not support both the selection and utilization of this non–physical entity on the relevant stations. This

involves reserving the specific connector or activating it, using the *configuration key* `ReserveConnectorZeroSupported` as defined in the OCPP standard [54]. In order to introduce this capability, the relevant frontend interfaces necessitate further modification to represent this particular type of connector. Failing to do so, renders the implementation of the reservation system being of little use, in its current state.
Remaining within the context of particular frontend functionalities, configuring each CS so far is only possible, by adjusting the *configuration key* key–value pairs, within the OCPP parameter section of the CS. For instance, enabling the support of bookings on logical connector zero could be achieved, by modifying or adding this value as a free–form string in the browser version of the service offering. Apart from a good level of technical knowledge, this requires an understanding of the concept of these keys in combination with the values they support. The same principle applies to adjusting the available features at the station. Through the use of so–called *feature profiles* [54], functionalities such as reservation support, can be fully disabled at the respective station. Representing the corresponding features enabled on the station, their simple existence as string values inside an array demolishes the chance of editing such data structures freehand without typing errors. As suggested here, and to facilitate easier use, the administrative user should be able to easily adjust these parameters, by selecting the desired capability on the dedicated station, using a feature toggle or a checkbox in the corresponding part of the GUI.
Regarding the `Tenant Component` entity, introduced in the *Implementation* chapter within Sub–subsection 7.3.2.5, there are additional opportunities to integrate the existing modules, into the reservation system proposal. Apart from integrating the **Car** component to assign cars of the EVUs to the corresponding reservations, another option is, to utilize the **Fees and Payment** component. For instance, the standard consumer has to pay for the electricity used to charge a vehicle, as well as to compensate the company providing the service. Thus, payment options and pricing calculations are crucial for effective accounting by both parties. Considering the actual properties of the booking entity, parameters such as arrival and departure time, combined with the current price available from the power supplier, could be used to estimate the later price of the charging session, allowing the user to know the actual cost of their charging before booking. Furthermore, the addition of the present battery capacity and the intended ultimate battery level could enable a more accurate computation.
Widening the perspective from pure cost estimation to include current electricity prices and taking into account the resulting load of connected vehicles on the underlying power grid, the concept of **smart charging** opens up a new area of problem statements. This includes in particular, addressing grid stability as well as the power distribution during peak demand. Apart from mitigating power outages, and assisting end users to plan their charging sessions more effectively, based on the optimization algorithms utilized in smart charging, there could be significant benefits in incorporating reservations as well. By providing inputs such as arrival and departure times in combination with the amount of energy required, these algorithms could work with more reliable values than those previously predicted and should lead to better results. As an extension to the `Reservation` entity, details on the current battery charge of the used car and the aforementioned pricing capabilities could be incorporated as well. This could permit an intelligent charging system to negotiate the best option for charging the user's vehicle, taking into account

the available information, as suggested in [57]. From the author's perspective, this could open the door to more advanced smart charging strategies, including V2G scenarios.

Despite the wide range of functionalities available, such systems are also limited in their ability to prevent certain situations or enforce correct functioning. These situations are addressed as part of the final section of this chapter.

8.3 Limitations

The fact that a software solution is described as part of the non–physical world, places certain limitations on the capabilities of such solutions. This is particularly true for reservation systems, that aim to connect physical and virtual entities and attempt to ensure specific user behavior, with respect to the established rule set. By considering the booking of suitable charging sessions in public and semi–public parking facilities equipped with CSs, various scenarios may arise where the user may be prevented from charging their vehicle, due to the misbehavior of other drivers or system users. Concerning these situations, the system will not always be able to prevent them from happening. To name just a few examples, starting with the most trivial one of a vehicle blocking the reserved parking space. Thus, the reservation does not allow the vehicle to be charged at this particular station, the parking space itself can not be blocked for this driver. From the perspective of some researchers in the literature, the introduction of physical barriers that block these parking spaces, until the appropriate driver arrives is essential, as suggested by the authors in [7]. But considering the expenses involved in installing such facilities, the efficiency in terms of cost and maintenance is beyond the scope of many CSOs and the providers of parking space.
Another inevitable scenario arises when a vehicle commences charging, prior to an approaching reservation. Therefore, this study proposes a preventative approach, by alerting the user of the forthcoming reservation, but the vacant connector on the CS could be used without restrictions until the reserved time slot arrives. These are just two examples of misbehavior, a system like this cannot stop. Further examples could be added, but the critical point to acknowledge is, that the application depends on a user, who behaves in a manner, that supports the overall concept and does not constrain other users. Otherwise, additional regulatory entities would be necessary, to supplement the system's rules in direct collaboration with the EVUs.

9 Resume and Outlook

Summarizing the results achieved in the development of a comprehensive reservation system for the management of charging infrastructure, this final chapter provides a resume concluding the tasks of this work. Furthermore, based on the achieved results, a consideration of the contribution to the proposals, elaborated in the literature, is given. Together with these retrospective views, this chapter concludes by providing an outlook on future possibilities for this category of system and proposing directions for future work. Especially, the administration and maintenance of charging infrastructure face several constraints, that are not determined by existing standards or technical limitations. A frequently underestimated fact refers to the process–related impediments to ensure the intended behavior from the system, the charging environment, and the driver. Beyond the behavior of the individual system parts and the corresponding actors, the exchange of information between these components is another aspect, that needs to be considered in the process of designing these types of systems. In order to provide a sophisticated experience, it should allow interconnectivity with stations from several manufacturers, interoperability with existing services, and ease of use by users. Therefore, the proposed design and development of a comprehensive solution aims to address all these concerns, by extending the aforementioned OCPP communication standard. By creating this PoC, the proposal should demonstrate, that it is feasible to coordinate charging session bookings for immediate as well as future sessions, by incorporating a basic set of functionalities. Concerning the fulfillment of these goals, initially stated at the beginning of this work, the resulting design proposal is at least capable of handling charging appointments, according to the OCPP standard in version 1.6. Thus, it is applicable for being implemented within a real–world scenario, dealing with the problem of coordinated CS allocation, presupposed the stations support the version mentioned above of the standard. In order to create the bookings, the system offers besides the standard *ReserveNow* method, which only permits immediate blocking of a connector, an extension of the OCPP functionality, to allow users to make arrangements in advance. Enabling EVUs to plan future charging sessions and recurring bookings, is not accommodated by other system approaches at the time of this research, as far as the author of this work is concerned. By using this solution, it is possible for drivers to make reservations over an extended period of time, in a more convenient manner. This proposal alleviates the inconvenience of making a booking for every single day the car needs to be charged. Concerning the elaborated scenario, this also caters to the requirements of the stakeholders. Particularly in the suggested situations this study considers, where designated user groups have specific parking spots they are associated with, which offer only limited available charging points. In a broader context, these scenarios could also apply to car parks, owned by providers or companies, who need to explicitly regulate the individual parking spaces, equipped with CSs. The range of domains, this proposal is applicable to, in order to allow advanced reservation life cycle management, as

well as a more detailed way of monitoring, is diverse. Thus, the author assumes, that this approach, in terms of contributing to the existing research for CS management, at least gives a better understanding of the obstacles as well as the opportunities, by introducing such solutions.
Taking into account the above–listed aspects, the reviewed proposal for a systematic approach for CSs allocation allows leastwise the following contributions in terms of supporting the existing research. In order to extend the range of scenarios considered in other projects, the introduction of further requirements, especially in terms of background processes broadens the ratio of possible unintended behavior. This for example includes the mitigation techniques, such as managing non–arriving EVUs and initiating mitigation processes, prior to the start of a charging session at stations with pending reservations. Alongside highlighting problematic situations, this documentation also presents suggestions for resolving such conflicts. For this purpose, it solely considers the options typically available to such systems, without the need for external factors, mostly not accessible in other scenarios. This ensures a higher degree of enforcing arriving reservations and a more automated approach, to managing infrastructure availability in a generic way. Furthermore, it avoids unnecessary blockages, fostered by the background processes mentioned above. Combined with the additional life cycle states, describing the validity of a booking, a detailed consideration of relevant error cases, is elaborated. Besides the exceptions from the designed processes, this collection of possible misbehavior, includes the potential errors produced by the charging infrastructure as well. Using this new terminology in conjunction with the understanding of the edge cases that have already been identified, there is potential for elaborating scenarios that require more complex combinations of features. This allows further extensions to be developed in combination with the elaborated entities, which serve as a guideline for encapsulating the standardized request objects.
Switching the perspective from solely focusing on managing stations or infrastructure at a specific location. By considering the possibility of connecting different infrastructures to create a network of interconnected stations, communicating via a common backbone, the problem most highlighted in the context of e–mobility, also known as 'range anxiety' [64], could also be addressed. As already suggested in the literature by [71], the use of roaming capabilities, to build a heterogenous network of CSs, may be capable of alleviating such fears. Through the enablement of EVs to share data on their current battery status and charging needs directly with the station, the ability to reserve charging sessions along the route would be possible. Without the intervention of the driver, the car should be capable of planning its charging sessions, based on the given destination and guarantee its owner the possibility to recharge. Especially during longer trips, as addressed by 'range anxiety', and an unknown environment, this offers more convenience in using EVs. Moreover, linking several CSOs, CPOs and EMSPs would lead to a higher acceptance of their services by the EVUs, and thus to an increase in both usage and profits. This facilitates both the development of new areas and offers numerous business prospects, by supporting the installation of stations and expanding the existing infrastructure. Nevertheless, it is important to take into account the increased load on the grid, that would be caused as a result. However, by knowing the amount of time, that the EV will eventually park at the CS location, combined with the current battery state, the amount of energy required to

recharge to a full state of charge can be determined. The remaining unneeded power can be used through smart charging scenarios such as V2G, which aims to support overall grid stability using excessive power from EVs. Other applications could include V2B to return energy to the building, where the power unit is located, V2H to replace the building by the driver's home, or V2X to cover various targets. In order to further promote the adoption of EVs, enhance the corresponding infrastructure, and provide support for all the scenarios mentioned, the need for future development work in this area is obvious.
Regarding the purpose of subsequent elaborations, below are some examples that the author of this work considers significant and possibly useful for future research. First to be mentioned, are the fundamental constraints, imposed by the state of the grid, which require special attention. Without a functional grid to power the relevant stations and vehicles, further developments in energy storage and infrastructure observation are pointless. Therefore, the simulations already in existence, taking into account the different types of elaborated reservations to measure their impact on the journey duration and overall comfort could be used. Introduced by Basmadjian et al. in [7, 8], these results could be used as groundwork, to create scenarios specifically for simulating the effects of reservations, to cushion peek loads using V2G. Utilizing the existing software tooling like *AgentPolis* [4], used in the above simulations to create the infrastructure and agent–based drivers, all that needs to be added is the power grid. Alongside the effective prevention of parking space blockades by users without bookings, using sensor–based solutions, the consideration of mobile CSs, as movable units, requires further investigations. Based on the existing literature, only loose papers take this type of station into account for application within reservation scenarios. Abstracting from the issues of continuous power demand from stationary units, mobile stations in the form of vehicles could potentially address the problems of limited parking spaces equipped with CSs. Considering the integration of reservations into more mobile scenarios can offer new opportunities, as already examined by [72]. Putting all these deliberations in a nutshell, the field of reservation systems for charging infrastructure management offers many opportunities and approaches for improvement. Apart from the creation of new standards, to introduce another more unified communication protocol, the use of machine learning algorithms, for extended optimization techniques, salvages huge potential as well.
The concrete results of this master's thesis and the necessary changes made to the applications during this work are available in the corresponding public forks of the original repositories, within the '*reservation–process*' branch. The relevant URLs as hyperlinks, are deposited in the following declarations for the **backend**, **mobile** and **web** components of the *Open e–Mobility* project.

Bibliography

[1] Sithara S. G. Acharige et al. “Review of Electric Vehicle Charging Technologies, Standards, Architectures, and Converter Configurations”. In: *IEEE Access* 11 (2023). Conference Name: IEEE Access, pp. 41218–41255. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3267164.

[2] Aswad Adib et al. “E-Mobility — Advancements and Challenges”. In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 165226–165240. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2953021.

[3] Shahab Afshar et al. “A Literature Review on Mobile Charging Station Technology for Electric Vehicles”. In: *2020 IEEE Transportation Electrification Conference & Expo (ITEC)*. 2020 IEEE Transportation Electrification Conference & Expo (ITEC). ISSN: 2377-5483. June 2020, pp. 1184–1190. DOI: 10.1109/ITEC48692.2020.9161499.

[4] *Agentpolis*. original-date: 2017-07-20T13:47:14Z. May 1, 2022. URL: https://github.com/aicenter/agentpolis (visited on 09/05/2023).

[5] *Angular*. URL: https://angular.io/ (visited on 08/28/2023).

[6] *Art. 17 GDPR - Right to erasure (’right to be forgotten’)*. GDPR.eu. Section: Uncategorized. Nov. 14, 2018. URL: https://gdpr.eu/article-17-right-to-be-forgotten/ (visited on 08/22/2023).

[7] Robert Basmadjian et al. “A Reference Architecture for Interoperable Reservation Systems in Electric Vehicle Charging”. In: *Smart Cities* 3.4 (Nov. 21, 2020), pp. 1405–1427. ISSN: 2624-6511. DOI: 10.3390/smartcities3040067. URL: https://www.mdpi.com/2624-6511/3/4/67 (visited on 03/13/2023).

[8] Robert Basmadjian et al. “An Interoperable Reservation System for Public Electric Vehicle Charging Stations: A Case Study in Germany”. In: *Proceedings of the 1st ACM International Workshop on Technology Enablers and Innovative Applications for Smart Cities and Communities*. BuildSys ’19: The 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation. New York NY USA: ACM, Nov. 13, 2019, pp. 22–29. ISBN: 978-1-4503-7015-8. DOI: 10.1145/3364544.3364825. URL: https://dl.acm.org/doi/10.1145/3364544.3364825 (visited on 03/13/2023).

[9] Richard Bemile, Akwasi Achampong, and Emmanuel Danquah. “Online Hotel Reservation System”. In: 1.9 ().

[10] Frank Brosi. *Methode zur Erzeugung eines erweiterten Konformitätstests für Kommunikationsprotokolle am Beispiel der ISO 15118*. Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart. Wiesbaden: Springer Fachmedien Wiesbaden, 2019. ISBN: 978-3-658-27532-7. DOI: 10.1007/978-3-658-27533-4. URL: http://link.springer.com/10.1007/978-3-658-27533-4 (visited on 08/07/2023).

[11] Paul Clements, ed. *Documenting software architectures: views and beyond*. 2nd ed. SEI series in software engineering. OCLC: ocn640079469. Upper Saddle River, NJ: Addison-Wesley, 2011. 537 pp. ISBN: 978-0-321-55268-6.

[12] *ConfigMaps*. Kubernetes. Section: docs. URL: https://kubernetes.io/docs/concepts/configuration/configmap/ (visited on 08/15/2023).

[13] Nicolò Daina, Aruna Sivakumar, and John W. Polak. "Electric vehicle charging choices: Modelling and implications for smart charging services". In: *Transportation Research Part C: Emerging Technologies* 81 (Aug. 2017), pp. 36–56. ISSN: 0968090X. DOI: 10.1016/j.trc.2017.05.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S0968090X17301365 (visited on 08/07/2023).

[14] Sanchari Deb, Mikko Pihlatie, and Mohammed Al-Saadi. "Smart Charging: A Comprehensive Review". In: *IEEE Access* 10 (2022). Conference Name: IEEE Access, pp. 134690–134703. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3227630.

[15] Glenda A. Delizo and Mischelle A. Esguerra. "ONLINE HOTEL RESERVATION AND MANAGEMENT SYSTEM FOR THE COLLEGE OF INTERNATIONAL TOURISM AND HOSPITALITY MANAGEMENT (CITHM)". In: *INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY* 10.1 (July 25, 2013), pp. 1201–1229. ISSN: 2277-3061. DOI: 10.24297/ijct.v10i1.3324. URL: https://www.cirworld.com/index.php/ijct/article/view/3324 (visited on 08/08/2023).

[16] *Die HKA - Die Hochschule Karlsruhe : Green4EVer: Integration der Elektromobilität in nachhaltige Smart-Buildings und Standortmanagement*. URL: https://www.h-ka.de/iaf/green4ever (visited on 08/09/2023).

[17] *Documentation · OData - the Best Way to REST*. URL: https://www.odata.org/documentation/ (visited on 09/08/2023).

[18] Roland Ferwerda et al. "Advancing E-Roaming in Europe: Towards a Single "Language" for the European Charging Infrastructure". In: *World Electric Vehicle Journal* 9.4 (Dec. 2018). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 50. ISSN: 2032-6653. DOI: 10.3390/wevj9040050. URL: https://www.mdpi.com/2032-6653/9/4/50 (visited on 08/07/2023).

[19] *Firebase | Google's Mobile and Web App Development Platform*. Firebase. URL: https://firebase.google.com/ (visited on 08/31/2023).

[20] Radu Flocea et al. "Electric Vehicle Smart Charging Reservation Algorithm". In: *Sensors* 22.8 (Apr. 7, 2022), p. 2834. ISSN: 1424-8220. DOI: 10.3390/s22082834. URL: https://www.mdpi.com/1424-8220/22/8/2834 (visited on 04/05/2023).

[21] *From Inventing the Enterprise Software Sector to Helping the World Run Better*. SAP. URL: https://www.sap.com/documents/2023/01/b8b97eef-5f7e-0010-bca6-c68f7e60039b.html (visited on 09/03/2023).

[22] Erich Gamma et al. *Design Patterns: Entwurfsmuster als Elemente wiederverwendbarer objektorientierter Software*. Ed. by John M. Vlissides. Trans. by Maren Feilen and Knut Lorenzen. 1. Aufl. Frechen: mitp-Verl, 2015. 472 pp. ISBN: 978-3-8266-9700-5.

[23] Moisés Antón García et al. "SGAM-Based Analysis for the Capacity Optimization of Smart Grids Utilizing e-Mobility: The Use Case of Booking a Charge Session". In: *Energies* 16.5 (Mar. 6, 2023), p. 2489. ISSN: 1996-1073. DOI: 10.3390/en16052489. URL: https://www.mdpi.com/1996-1073/16/5/2489 (visited on 04/04/2023).

[24] J. García-Villalobos et al. "Plug-in electric vehicles in electric distribution networks: A review of smart charging approaches". In: *Renewable and Sustainable Energy Reviews* 38 (Oct. 2014), pp. 717–731. ISSN: 13640321. DOI: 10.1016/j.rser.2014.07.040. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364032114004924 (visited on 08/07/2023).

[25] Zacharenia Garofalaki et al. "Electric Vehicle Charging: A Survey on the Security Issues and Challenges of the Open Charge Point Protocol (OCPP)". In: *IEEE Communications Surveys & Tutorials* 24.3 (2022). Conference Name: IEEE Communications Surveys & Tutorials, pp. 1504–1533. ISSN: 1553-877X. DOI: 10.1109/COMST.2022.3184448.

[26] *GitHub - sap-labs-france/ev-dashboard: The Open e-Mobility Charging Station management front-end Angular application (check also ev-server and ev-mobile)*. URL: https://github.com/sap-labs-france/ev-dashboard (visited on 09/03/2023).

[27] *GitHub - sap-labs-france/ev-mobile: The Open e-Mobility Charging Station management mobile React-Native application (check also ev-server and ev-dashboard)*. URL: https://github.com/sap-labs-france/ev-mobile (visited on 09/03/2023).

[28] *GitHub - sap-labs-france/ev-server: The Open e-Mobility Charging Station management backend server (check also ev-dashboard and ev-mobile)*. URL: https://github.com/sap-labs-france/ev-server (visited on 09/03/2023).

[29] *GitHub - SAP/e-mobility-charging-stations-simulator: OCPP-J charging stations simulator*. URL: https://github.com/SAP/e-mobility-charging-stations-simulator (visited on 09/03/2023).

[30] Till Gnann et al. "Fast charging infrastructure for electric vehicles: Today's situation and future needs". In: *Transportation Research Part D: Transport and Environment* 62 (July 2018), pp. 314–329. ISSN: 13619209. DOI: 10.1016/j.trd.2018.03.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S1361920917305643 (visited on 08/03/2023).

[31] Robert Goecke. "The Evolution of Online Booking Systems". In: *Handbook of e-Tourism*. Ed. by Zheng Xiang et al. Cham: Springer International Publishing, 2020, pp. 1–25. ISBN: 978-3-030-05324-6. DOI: 10.1007/978-3-030-05324-6_27-1. URL: http://link.springer.com/10.1007/978-3-030-05324-6_27-1 (visited on 08/07/2023).

[32] Andreas Heinrich and Michael Schwaiger. "ISO 15118 – charging communication between plug-in electric vehicles and charging infrastructure". In: *Grid Integration of Electric Mobility*. Ed. by Johannes Liebl. Proceedings. Wiesbaden: Springer Fachmedien, 2017, pp. 213–227. ISBN: 978-3-658-15443-1. DOI: 10.1007/978-3-658-15443-1_17.

[33] *Helm*. URL: https://helm.sh/ (visited on 08/15/2023).

[34] Sara Hsaini, Mounir Ghogho, and My El Hassan Charaf. "An OCPP-Based Approach for Electric Vehicle Charging Management". In: *Energies* 15.18 (Sept. 15, 2022), p. 6735. ISSN: 1996-1073. DOI: 10.3390/en15186735. URL: https://www.mdpi.com/1996-1073/15/18/6735 (visited on 04/05/2023).

[35] *Ingress*. Kubernetes. Section: docs. URL: https://kubernetes.io/docs/concepts/services-networking/ingress/ (visited on 08/15/2023).

[36] *JavaScript | MDN*. July 8, 2023. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (visited on 08/28/2023).

[37] *JavaScript With Syntax For Types*. URL: https://www.typescriptlang.org/ (visited on 08/28/2023).

[38] M. Kathiresh, G. R. Kanagachidambaresan, and Sheldon S. Williamson, eds. *E-Mobility: A New Era in Automotive Technology*. EAI/Springer Innovations in Communication and Computing. Cham: Springer International Publishing, 2022. ISBN: 978-3-030-85423-2. DOI: 10.1007/978-3-030-85424-9. URL: https://link.springer.com/10.1007/978-3-030-85424-9 (visited on 07/28/2023).

[39] Hye-Jin Kim et al. "An Efficient Scheduling Scheme on Charging Stations for Smart Transportation". In: *Security-Enriched Urban Computing and Smart Grid*. Ed. by Tai-hoon Kim, Adrian Stoica, and Ruay-Shiung Chang. Vol. 78. Series Title: Communications in Computer and Information Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 274–278. ISBN: 978-3-642-16443-9. DOI: 10.1007/978-3-642-16444-6_35. URL: http://link.springer.com/10.1007/978-3-642-16444-6_35 (visited on 04/04/2023).

[40] Benedikt Kirpes et al. "E-Mobility Systems Architecture: a model-based framework for managing complexity and interoperability". In: *Energy Informatics* 2.1 (Dec. 2019), p. 15. ISSN: 2520-8942. DOI: 10.1186/s42162-019-0072-4. URL: https://energyinformatics.springeropen.com/articles/10.1186/s42162-019-0072-4 (visited on 03/13/2023).

[41] John C. Klensin. *Simple Mail Transfer Protocol*. Request for Comments RFC 5321. Num Pages: 95. Internet Engineering Task Force, Oct. 2008. DOI: 10.17487/RFC5321. URL: https://datatracker.ietf.org/doc/rfc5321 (visited on 09/01/2023).

[42] Philippe Kruchten. *The rational unified process*. Addison-Wesley object technology series. Reading, Mass: Addison-Wesley, 1999. 255 pp. ISBN: 978-0-201-60459-7.

[43] Junghoon Lee, Gyung-Leen Park, and Hye-Jin Kim. "Reservation-Based Charging Service for Electric Vehicles". In: *Algorithms and Architectures for Parallel Processing*. Ed. by Yang Xiang et al. Vol. 7017. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 186–195. ISBN: 978-3-642-24668-5. DOI: 10.1007/978-3-642-24669-2_18. URL: http://link.springer.com/10.1007/978-3-642-24669-2_18 (visited on 04/04/2023).

[44] Marcel Linnemann and Christoph Nagel. *Elektromobilität und die Rolle der Energiewirtschaft: Rechte und Pflichten eines Ladesäulenbetreibers*. Wiesbaden: Springer Fachmedien Wiesbaden, 2020. ISBN: 978-3-658-30216-0. DOI: 10.1007/978-3-658-30217-7. URL: http://link.springer.com/10.1007/978-3-658-30217-7 (visited on 08/07/2023).

[45] Littlefuse. "Designing Safer, More Efficient, and Reliable EV Charging Stations". In: (June 2020). URL: https://m.littelfuse.com/~/media/electronics_technical/application_notes/resettable_ptcs/littelfuse_designing_ev_charging_stations_technicalarticle.pdf (visited on 07/28/2023).

[46] Xiaopu Ma et al. "Specifying and enforcing the principle of least privilege in role-based access control". In: *Concurrency and Computation: Practice and Experience* 23.12 (2011). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.1731, pp. 1313–1331. ISSN: 1532-0634. DOI: 10.1002/cpe.1731. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1731 (visited on 08/23/2023).

[47] Alexey Melnikov and Ian Fette. *The WebSocket Protocol*. Request for Comments RFC 6455. Num Pages: 71. Internet Engineering Task Force, Dec. 2011. DOI: 10.17487/RFC6455. URL: https://datatracker.ietf.org/doc/rfc6455 (visited on 08/03/2023).

[48] *MongoDB: The Developer Data Platform*. MongoDB. URL: https://www.mongodb.com (visited on 08/15/2023).

[49] Jesús Jaime Moreno Escobar et al. "A Comprehensive Review on Smart Grids: Challenges and Opportunities". In: *Sensors* 21.21 (Oct. 21, 2021), p. 6978. ISSN: 1424-8220. DOI: 10.3390/s21216978. URL: https://www.mdpi.com/1424-8220/21/21/6978 (visited on 08/07/2023).

[50] *Namespaces*. Kubernetes. Section: docs. URL: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ (visited on 08/15/2023).

[51] *NGINX Ingress Controller*. URL: https://docs.nginx.com/nginx-ingress-controller/ (visited on 08/15/2023).

[52] *Node.js*. Node.js. URL: https://nodejs.org/en (visited on 08/28/2023).

[53] *Notifee*. Notifee. URL: https://notifee.app/ (visited on 09/01/2023).

[54] *OCPP 1.6, Protocols, Home - Open Charge Alliance*. URL: https://www.openchargealliance.org/protocols/ocpp-16/ (visited on 03/16/2023).

[55] *Open Charge Alliance - Global Platform For Open Protocols*. URL: https://www.openchargealliance.org/ (visited on 08/07/2023).

[56] *Open Charge Point Interface 2.2.1*. Oct. 6, 2021. URL: https://evroaming.org/app/uploads/2021/11/OCPI-2.2.1.pdf (visited on 04/17/2023).

[57] Simone Orcioni and Massimo Conti. "EV Smart Charging with Advance Reservation Extension to the OCPP Standard". In: *Energies* 13.12 (June 24, 2020), p. 3263. ISSN: 1996-1073. DOI: 10.3390/en13123263. URL: https://www.mdpi.com/1996-1073/13/12/3263 (visited on 08/17/2023).

[58] Simone Orcioni et al. "Electric Vehicles Charging Reservation Based on OCPP". In: *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*. 2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe). Palermo: IEEE, June 2018, pp. 1–6. ISBN: 978-1-5386-5186-5. DOI: 10.1109/EEEIC.2018.8494366. URL: https://ieeexplore.ieee.org/document/8494366/ (visited on 03/13/2023).

[59] *Overview of Docker Compose*. Docker Documentation. Aug. 15, 2023. URL: https://docs.docker.com/compose/ (visited on 08/15/2023).

[60] Sanjay Patni. *Pro RESTful APIs*. Berkeley, CA: Apress, 2017. ISBN: 978-1-4842-2664-3. DOI: 10.1007/978-1-4842-2665-0. URL: http://link.springer.com/10.1007/978-1-4842-2665-0 (visited on 08/08/2023).

[61] *Persistent Volumes*. Kubernetes. Section: docs. URL: https://kubernetes.io/docs/concepts/storage/persistent-volumes/ (visited on 08/15/2023).

[62] *Produktionsreife Container-Orchestrierung*. Kubernetes. URL: https://kubernetes.io/de/ (visited on 08/15/2023).

[63] Hua Qin and Wensheng Zhang. "Charging scheduling with minimal waiting in a network of electric vehicles and charging stations". In: *Proceedings of the Eighth ACM international workshop on Vehicular inter-networking*. Mobicom'11: The 17th Annual International Conference on Mobile Computing and Networking. Las Vegas Nevada USA: ACM, Sept. 23, 2011, pp. 51–60. ISBN: 978-1-4503-0869-4. DOI: 10.1145/2030698.2030706. URL: https://dl.acm.org/doi/10.1145/2030698.2030706 (visited on 08/17/2023).

[64] Nadine Rauh, Thomas Franke, and Josef F. Krems. "Understanding the Impact of Electric Vehicle Driving Experience on Range Anxiety". In: *Human Factors* 57.1 (Feb. 1, 2015). Publisher: SAGE Publications Inc, pp. 177–187. ISSN: 0018-7208. DOI: 10.1177/0018720814546372. URL: https://doi.org/10.1177/0018720814546372 (visited on 08/02/2023).

[65] *React Native · Learn once, write anywhere*. URL: https://reactnative.dev/ (visited on 08/28/2023).

[66] *SAP S/4HANA Cloud, Public Edition*. SAP. URL: https://www.sap.com/germany/products/erp/s4hana.html (visited on 09/06/2023).

[67] *SGAM Toolbox – Modelling aid for the Smart Grid Architecture Model*. URL: https://sgam-toolbox.org/ (visited on 08/19/2023).

[68] Kartik Sharma and Kalpana Chauhan. “Smart Grid System: A Review”. In: *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*. 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE). Bengaluru, India: IEEE, Oct. 9, 2020, pp. 614–618. ISBN: 978-1-72817-213-2. DOI: 10.1109/ICSTCEE49637.2020.9277503. URL: https://ieeexplore.ieee.org/document/9277503/ (visited on 08/07/2023).

[69] *StatefulSets*. Kubernetes. Section: docs. URL: https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/ (visited on 08/15/2023).

[70] ChengJian Xu et al. “Generation and management of waste electric vehicle batteries in China”. In: *Environmental Science and Pollution Research* 24.26 (Sept. 1, 2017), pp. 20825–20830. ISSN: 1614-7499. DOI: 10.1007/s11356-017-9890-8. URL: https://doi.org/10.1007/s11356-017-9890-8 (visited on 08/02/2023).

[71] Azamat Zarkeshev and Csaba Csiszar. “Charging reservation service for electric vehicles using automatic notification”. In: Mar. 25, 2018.

[72] Xu Zhang et al. “Mobile Charging as a Service: A Reservation-Based Approach”. In: *IEEE Transactions on Automation Science and Engineering* 17.4 (Oct. 2020), pp. 1976–1988. ISSN: 1545-5955, 1558-3783. DOI: 10.1109/TASE.2020.2983819. URL: https://ieeexplore.ieee.org/document/9068444/ (visited on 04/06/2023).