POLITECNICO
MILANO 1863

eMall - e-Mobility for All

Requirements **A**nalysis and **S**pecification **D**ocument

Riccardo Motta
Pierluigi Negro

January 8, 2023

Version 1.1

Contents

Chapter 1

Introduction

Nowadays, electric vehicles are becoming very popular. The need to reduce pollution to save our planet is compelling, and anything we can do to reduce the effects of climate change is moving towards the use of clean energy sources, and everyone can do their part.

Many people are changing their cars to electric ones, and in many cities charging stations are being installed for these vehicles. The main aim of the eMall project is to limit the carbon footprint of our mobility by making easy and efficient charging electric cars, focusing on planning the charge.

The eMall project also focuses on the energy needed to accomplish charging cars, as it's important to manage it most efficiently. This is done to reduce the cost of the primary resources and to make the service even more competitive and cheap. The eMall project tries to simplify also this aspect of the charge chain.

1.1 Purpose

The purpose of this document is to provide a general, complete, high-level view of the eMSP and CPMS systems, and of their interactions.

This document comprehends:

- The description of the main goals of the project, with the identification of the interesting phenomena. These phenomena are selected among the ones that might indirectly influence the system or that the system has knowledge of.
- The main scenarios the system can handle. Each of them is then precisely depicted more formally thanks to the use of diagrams.
- The system's requirements.
- A formal analysis of the requirements' feasibility.

1.2 Scope

eMSP | Simplify the charging process of electric vehicles Charging an electric vehicle can be a challenging task to do. Nowadays, it's not so easy to find charging stations where to plug the vehicle. So, also looking at the future where this kind of station will be more present in the territory, it's fundamental to have a system like this one to carefully plan the charge of the vehicle, in order to minimize the time spent on this task, so that it doesn't have any impact on drivers' lives. Thanks to this system it's possible to search for a charging station nearby, check prices and sales, and, of course, book a charging slot for the vehicle.

CPMS | Optimize the energy usage and supply To be competitive, it's also important to minimize the costs of the whole infrastructure, paying careful attention to the price of the electricity and to the mix of primary sources from which it was derived. On the market, there are possibly many DSOs, each one with different supply capabilities, prices, and energy mixes. Choosing from which source to buy the energy is crucial, especially if the CPO can store it in batteries at the charging station.

1.3 Goals

More formally, here are formalized all the goals that the system should archive.

1.3.1 eMSP goals

This first part focuses on the goals related to the eMPS.

G1	Allow the user to see all the charging stations
The user can see and look for all the available charging stations with all the important information, like the cost of the charge and any special offers.	

G2	Allow the user to book a charge
Every user can reserve a charge if any slot is available. The user can select a charging station, the starting time, and the duration of the charge. Moreover, s/he selects the type of socket to use.	

G3	Allow the user to start the charge
Every user who has booked can start the charging process during the booked period. This is done automatically by recognizing the connected vehicle once the socket is plugged in.	

G4	Let the system notify the user when the charge is finished
End users are informed every time one of their vehicles charging in one station ends the recharging process through a notification.	

G5	Allow the user to pay for the service
Every user can pay for the obtained charging process.	

1.3.2 CPMS goals

This second part, instead, focuses on the goals related to the CPMS.

G6	Let the system have information about charging stations
Know the position and the characteristics of all the various charging stations, like sockets, batteries, available energy, and occupation.	

G7	Allow for cars to charge
Start the charge of a vehicle, monitoring the process.	

G8	Fetch information from DSO and decide which DSO to use
Acquire the price of the electricity from the providers and decide from which to buy, if more options are available.	

G9	Define energy mix
Decide the mix of electricity to provide to the sockets, deciding whether to buy energy, use the one stored in batteries or buy energy to store in batteries.	

G10	Manage special offers
Create new special offers on prices for charging.	

1.4 Phenomena

This part focuses on the description of the various interesting phenomena that are important in order to correctly develop this project.

1.4.1 World phenomena

World phenomena are the ones that are proper of the world's domain, and therefore invisible to the modeled machine, but still, it's important to point them out since they influence the system.

Identifier	Description
WP1	The user looks for a nearby station (with its cost and offers).
WP2	The user brings the vehicle to the charging column for the booked charging slot.

1.4.2 Shared phenomena

This first part points out the shared phenomena between the world and the machine that are world-controlled.

Identifier	Description
SP1	The user books a charging slot in a charging station.
SP2	The user plugs in the vehicle for the charge.
SP3	The user unplugs the vehicle after the charge or to stop it.
SP4	The user pays for the obtained service.
SP5	The DSO changes the price of the electricity.
SP6	The electricity for charging the vehicles is provided.

This second part, instead, focuses on the machine-controlled shared phenomena.

Identifier	Description
SP7	The system starts the charge of a vehicle.
SP8	The system notifies the user when the charging process is finished.
SP9	The system changes the charging level of the batteries, storing or consuming their energy.
SP10	The system manages the distribution of energy to the vehicles.
SP11	The system acquires information about the current price of the energy offered by DSOs.

Moreover, some shared phenomena can be both world-controlled and machine-controlled. These are listed down here.

Identifier	Description
SP12	The cost of a charge is changed.
SP13	A special offer is added or removed.
SP14	The energy mix for a specific vehicle is decided.
SP15	The decision whether to use the battery energy or to store it in one is taken by the user.
SP16	The decision of the DSO from which buying the energy is taken by the user.

1.5 Definitions, acronyms and abbreviations

1.5.1 Definitions

Word	Definition
eRoaming	The *eRoaming* is an external service that the system can query in order to obtain information about the various connected eMPSs and CPMSs.

1.5.2 Acronyms

Acronym	Description
API	*Application Programming Interface*: is a set of instructions and interfaces that is standardized in order to facilitate the communication between different pieces of software.
CPMS	*Charge Point Management System*: the CPOs management system that supports the process of charging vehicles and acquiring energy from the various DSOs.
CPO	*Charging Point Operator*: the name of the operators managing the charging points and providing the actual charging service.
DBMS	*DataBase Management System*: it's the system that sits between the physical data structures on disk and the user, allowing to create, read, update and delete data.
DSO	*Distribution System Operator*: the name of the operators that provide the actual energy used during the charging process.
eMSP	*e-Mobility Service Provider*: the name of the providers of the service to the end users, acting as intermediaries between the users and the backing CPOs.
ES6	*ECMAScript 6*: is a JavaScript standard which ensures interoperatibility between different web browsers.
JWT	*JSON Web Token*: is a standardized method for securely representing claims between two parties.
UML	*Unified Modeling Language*: is a modeling language used to formally specify the design of a system.

1.5.3 Abbreviations

Abbreviation	Description
eMall	eMall - e-Mobility for All.

1.6 Revision history

Version 1.0 released on December 22, 2022: original release.

Version 1.1 released on January 8, 2023: uniformed section names, fixed alloy worlds, updated UC5, UC6 and UC8, added Credits section and fixed typos.

1.7 Reference documents

1. **M. Camilli, E. Di Nitto, M. G. Rossi**; *eMall - e-Mobility for All project* (2022)

1.8 Document structure

Chapter 1: Introduction (page 4) This first part provides a general introduction to the project and describes the main purpose and goals it tries to archive, defining the scope of the project and pointing out some definitions, acronyms, and abbreviations used in this document. Moreover, there are all the revisions of the document with all the references used for writing it.

Chapter 2: Overall Description (page 9) This second part focuses on a formal description of the project, explaining all the possible scenarios in which the application is used. The formalization part in this section is done by taking advantage of various UML diagrams, which won't cover every possibility since, as stated in this chapter, some domain assumptions limit the scope. Furthermore, it's provided a brief but precise description of the characteristics of the various types of users, with their respective needs, that interact with the system.

Chapter 3: Specific Requirements (page 15) This third part, instead, is concerned with all the specific requirements that can be useful when developing the system. In particular, are pointed out all the interfaces that should be present (user, hardware, software, and communication interfaces), together with functional and performance requirements. Also, there are some design constraints and all the reliability, availability, security, maintainability, and portability attributes of the system.

Chapter 4: Formal Analysis (page 35) This fourth part covers the formal description of the whole system thanks to the use of a modeling language such as Alloy that allows proving the feasibility of the goals, starting from all the constraints and requirements.

Chapter 5: Conclusions (page 49) This last part contains the last notes about the project and concludes this document, pointing out the effort that each of the authors of this document spent in order to release it.

Chapter 2

Overall Description

This chapter describes the whole system, focusing on use cases, scenarios, functionalities, domain assumptions, and users thanks to the use of formalization diagrams, like UML.

2.1 Product perspective

In this section are listed all the possible scenarios that the system should be able to handle, and there is also an overview of the class structure of the project that is used to accomplish these needs.

2.1.1 Scenarios

These first scenarios are seen from the point of view of the end user (consumer)[1] who wants to charge the vehicle.

S1	**Registration and setup**
Mathew Assafo has just bought a new electric car, but he finds out that there are only a few charging stations in the city where he lives. After talking to some friends that already bought an electric car, he finds out that there exists an eMSP service that allows one to search for charging stations and book a charge. So when he arrives back home, he decides to look into it. Once he finds the eMSP, he looks for the sign-up page and inserts all the required data. Once filled up, he receives an email containing a link for confirming the account. After having confirmed the address, he is redirected to the login page where he inserts his username (email address) and password. Since this is the first time he logs in, he is prompted to choose which one he prefers as the default method for receiving notifications. He can choose between receiving an in-app notification or an email. He chooses the first option and the system warns him that whenever the system cannot reach any logged-in device, that notification will be sent via email.	

S2	**Booking a charge**
Mathew Assafo has just completed the registration procedure and has already set up the application (he has set the preference for notifications). Now he can finally start planning his car's charges. First, he opens the vehicles section where he can insert all the data about his new electric car, then he opens the map to look for all the available charging stations. The system immediately asks him if he wants to enable the geolocalization of the device to easily see all the nearby stations. He grants permission and the map focuses on the surroundings of his house, finding out that there is a relatively near station that he has never seen before. He is happy since he doesn't have to move too much from home, making it not a problem if it rains or if it's cold. Furthermore, he clicks on the station icon to see all the details and to book the first charge. A popup appears showing him all the details of that charging station, including the 20% discount on his first charge. Since it has few available slots, he presses on the booking button and selects an available slot of one hour for the next Sunday (while the vehicle to charge is already chosen by the system since it's the only available one). He already knows that next weekend he'll have to go with his wife to buy something for Christmas since it's already December, and they have no idea which gifts to buy for their children, so he decides that an extra charge to the vehicle would be worth it. He also marks the station as "favorite", so that he'll be able to find it immediately next time. After he does this, a star appears in the corner of the map and on the charging station. Just to be sure, he clicks on the star on the map and finds out that the charging station he has just marked as "favorite" is there.	

[1] A precise description of the different types of users (consumer and CPO's) is provided in section 2.4 User characteristics (page 14).

S3	Booking a charge and editing a reservation

Mathew Assafo has just booked a charge for his vehicle in the charging station near his house. Since he needs his car also for going to work (lamentably, there are not so many public transport facilities where he lives) he clicks on the search bar and types in the address of his working place. Since there are a few stations near his workplace, he decides to switch from the map view to the list view, so that he can sort the results according to the price they offer, the distance from his workplace, and the general availability of the stations. After doing some tests, he finds out which station satisfies him. He marks it as "favorite" and books a few charges for the next few weeks. After booking these charges, he goes back to the home page where he finds a summary of all the future charges, in a list of cards with some brief details. He opens that section and scrolling down the list he realizes that the one he booked for Sunday coincides with a family meeting. So he clicks on the reservation and sees that, near the time section, there is an edit button (he also notices the red recycle bin on top, but he imagines that it'll probably delete the reservation). So he clicks on edit and a calendar with all the available slots (with the one he already booked that is marked) appears. He selects a previous slot so that the car will be fully charged before the family meeting and confirms.

S4	Start of charge

It's Sunday morning, and Mathew Assafo wears his winter coat and brings the car to the charging station where he booked the charge. Before getting there, he receives a notification telling him the exact charging slot in which he has to park, so he stops there and connects the charging cable to the column. Immediately he notices that his car has started charging. He's surprised about the speed and how easy it is. Satisfied, he walks back home, in time for the start of the Formula 2 Grand Prix of that morning.

S5	End-of-charge notification

It's Sunday morning and Mathew Assafo is chilling on the sofa watching the Formula 2 Grand Prix. Suddenly his smartphone receives a notification. He gets up, takes the phone, and sees that the notification is from the eMSP application and is telling him that his vehicle has just finished charging. Even though he charged his car a couple of times during the previous working week, he is still surprised about the efficiency of the whole process. He takes his winter coat and goes to the charging station. He unplugs the vehicle and a notification immediately informs him that he can pay for the charge. He opens the application and clicks on the charge that is highlighted on the home page (the one he has to pay). When he starts the payment procedure in the application, he realizes that he doesn't remember some payment data, so he puts it back into his pocket and takes out his debit card. He goes near the charging column he used for charging the car and follows the instructions on the screen for paying the charge. Once done, he receives a notification of success, sits in his electric car, and goes back home, ready for the family meeting.

The following scenarios are seen from the point of view of the CPO's authorized user who accesses the CPMS.

S6	Changing DSO

Cameron Dya(z) is the owner of charging stations in the city of Paderno and discovers that, due to political allegations in the ongoing war, it would be beneficiary to the image of her company to not affiliate with DSOs owned by the attacking country, and decides to carry out the idea by changing DSOs provider. She turns on her computer, logs into the website of the CPMSs, heads to the section regarding the choice of the DSO, and clicks on "Deactivate automatic DSO choice", then confirms the action. Afterward, she selects a new DSO from the ones operating in her city, making sure that the company of choice has nothing to do with the ongoing conflict. She once again confirms the action, logs out from the website, and turns off the computer, hoping that her actions will not be unnoticed.

S7	Changing the energy mix (case 1)

The manager of the charging stations in Campobasso, Ahmed Ici, while listening to the weather forecast on the television, discovers that later that day, due to a downpour, the solar panels are not going to produce much energy. Knowing that without the help of the solar panels a lot of energy will be forcefully bought from the DSO, he opens his laptop and logs into the website of the CPMSs. He's in luck: by checking on the outgoing situation, he discovers that not a lot of cars are at the stations at the moment. This means that redirecting the energy produced by the solar panels to fill the station batteries will not have a huge impact in terms of costs. The batteries will help tamper with the absence of solar energy later that day, during the heavy rain, being a secondary source of the energy provided by the DSO. Hence, he goes to "Change energy mix", and connects "solar panels" and "batteries", effectively rerouting the energy flow. He confirms his choices and logs out of the website, knowing that once the batteries will be fully recharged the system will re-route the energy flow to a better usage of solar energy.

S8	Changing the energy mix (case 2)

To better combat battery aging, the CPO operator Joanna Sedini decides to let the batteries in the charging station that she manages discharge, seen as lately no cars bought a charge in the evening hours, and during the day the solar panels manage to cover the whole energy requirement. She, therefore, opens her laptop, logs into the website, and selects "Change energy mix", connecting "Batteries" and "Outlet". She then logs out and turns off her computer.

S9	Adding a new special offer

It's Christmastime, and the CPO CEO Carol Frat has a great idea: there should be a special Christmas offer for charges at her charging station. This will help persuade people to leave their electric cars charging while they go through their Christmas shopping spree, and maybe tempt other people into buying an electric car, seeing as charging it would come at a low price. Finalizing this idea is also fairly easy: she just goes into the website, logs in, selects "Create new special offer", and inputs the parameters. She sets the starting date at the 8^{th} of December of the current year, the end date at the 6^{th} of January of next year, and the discount at 10%.

S10	Monitoring of crowd situation at the stations

Bruce He(l)m wants to renew one of his charging stations with a new set of batteries. Since all charging stations are similar in terms of battery status and performance comparisons, so he decides to play a quick game: he'll renew the most crowded station at the time of checking. So he opens a browser, goes to the website, and logs in. Then he proceeds to check the external situation of the stations one by one, and there is a single station currently winning the race: it's the only one with a lone car charging, all the others are empty! Unhappy with this result, he decides to change metric and looks for the station where the batteries are at their lowest charge. Funnily enough, the same station has a slightly lower percentage, so the decision can be easily taken.

2.1.2 Class diagram

For graphical purposes, no attributes are shown and all the various handlers are not connected to all the components in the same package that uses it, but they are well integrated into the system. The same stands for the various APIs.

An example of this is the `PaymentService`. The `PaymentHandler` of the eMSP is the one that manages the transaction. It's connected to the `User` for getting data for notification, but also to `CPMShandler` since it has to notify the CPMS of the payment.

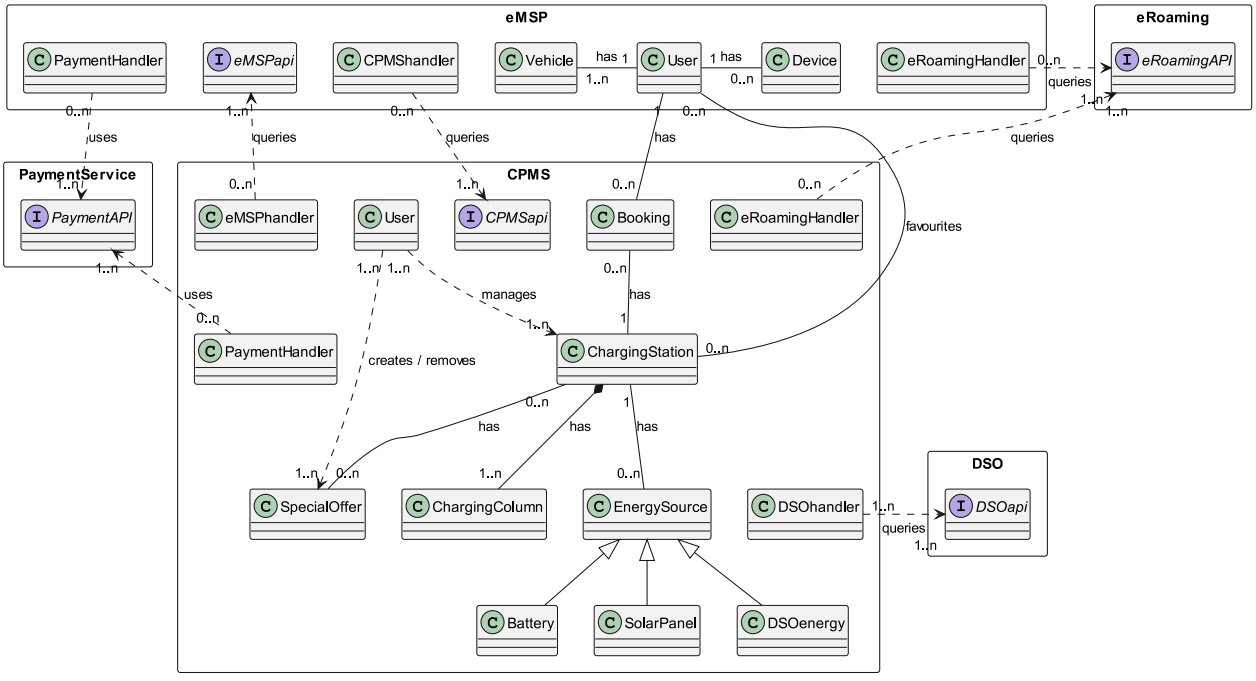

Figure 2.1: the class diagram of the various systems with the main interactions.

2.2 eMSP functions

This section explains all the different functionalities that the eMSP provides to the end users (consumers).

Note All the notifications that are mentioned in this section are in-app notifications (or emails, if the user selected this as the preferred notification method) that can be sent to the user only if there is at least one user's logged-in device with his/her credentials, otherwise, the email is chosen for the notification. If the notification derives from direct user interactions within the application, in-app notifications are sent only to the logged-in devices of the user which aren't the ones s/he has used to perform that action.

2.2.1 Registration and login/logout

Every new user who wants to register is asked to provide some basic information, like name, surname, birthdate, an email address that will be used as the username, and a password (which has to be given twice).

In case the email address is already present in the database or the two password fields don't coincide, the sign-up request is aborted, and the user is notified about this. If otherwise, the request ends successfully, the user receives an email at the specified address (username) with a confirmation link that expires within 24h from the emission. If the user doesn't confirm the address, the registration is automatically deleted. When the account is confirmed it becomes accessible through the login procedure and a confirmation email is sent to the user.

Every registered user can log into the system by providing username and password. If both are present in the database and the account has been confirmed, the user is granted access to his/her private area in which s/he can look for a charging station, book a charge or pay for it (and a few more other functionalities). Moreover, there is the possibility to log out. This deletes all the information stored locally on the user's device preventing him/her from accessing the account before logging in again.

2.2.2 Look for nearby stations

Every logged user can access a page with all the charging stations which can be sorted by distance (manually inserting the location or geolocalizing the user's device), by price (taking into account also any eventual discounts), or by availability. The user is also given the possibility to select a charging station, in which case the system will provide all the information related to that station, like the name, the precise location (address), the price for the charge, the number of parking slots for recharging and a calendar with all the availabilities (the available slots per period of time). Moreover, here is where the user is given the possibility to book a charge (which is described below in the next subsection).

2.2.3 Book a charging station for a certain timeframe

The booking of a charging slot is to be intended as the prosecution of the preceding subsection.

When the user wants to book a slot in a specific station, s/he selects the car s/he wants to charge (if more than one is associated), and the system gives him/her all the available consecutive time slots[2] for every day (with a fixed maximum look ahead from the current date). The system will try, as much as possible, to organize the physical slots according to the reservations, so it gives the user only an aggregated view of the availabilities. In any case, if the reservation ends successfully, the system sends a notification with the details of the reservation and the possibility to add it to the calendar.

2.2.4 Receive important notifications

15 minutes before the booked charge (or immediately if the user books a charge for that moment), the user receives a notification indicating the exact charging slot s/he has been assigned by the system for the charge.

Moreover, once the CPMS notices that the charging process of the user's vehicle is finished[3], it informs the eMSP which sends a notification to the user, telling that the charge is finished, and that s/he can proceed in removing the socket and the vehicle from the charging zone and paying for the service.

[2] Available consecutive time slots are adjacent time slots at a specific charging column. As an example, if there are only two available adjacent time slots related to two different charging columns at the same station, they aren't shown as a unique slot since the user would be forced to move the vehicle in the middle of the charge between the two columns.

[3] This, of course, works only if the user booked that charging slot, otherwise, the system is structured in a way that it forbids the user to recharge the vehicle by simply not charging it.

2.2.5 Pay for the service

After the charge is finished, the user is allowed to pay for the obtained service. This can be done through an apposite section in his/her personal area in the application or through a physical in-place device[4]. In any case, the user is notified once the payment is completed (both in case of success and failure). If the payment fails, the user can retry it, and this process repeats until the user successfully pays for the service.

2.2.6 Query external services

The system periodically queries an eRoaming service that informs it about all the available CPOs that subscribed to that network. From that moment on, the eMSP can interact with all their specific CPMSs, exchanging information thanks to the use of standard APIs.

2.3 CPMS functions

This section, instead, covers all the functionalities that the CPMS should provide to the authorized users of the CPO.

2.3.1 Login/logout

Every registered user can log into the system by providing a username and password. If a match is found in the database, the user is granted access to the area where s/he can manage the allowed charging station. Having logged in, the user can at any point log out from the website.

2.3.2 Manage energy mix

After login, users can define a new energy mix to be used by the system for a specific charge station. This includes defining from which sources energy is provided to sockets and if batteries are to be recharged, used, or ignored. Users can also let the system automatically define energy mixes that can change over time.

2.3.3 Manage DSO choice

After login, users can select what DSO is being used by the system, or let the system choose automatically based on the lowest rates. DSO rates can be queried from a list of available DSOs.

2.3.4 Get info about a charging station's internal/external status

After login, users can query information about a charging station's external status. This includes all available charging columns and relative sockets and how many cars are present. Users can also query information about a charging station's internal status. This includes amounts of energy available in its batteries, and for each charging vehicle, the amount of power absorbed and time left to the end of the charge.

2.3.5 Manage special offers

After login, users can create a new special offer by specifying a start date, an end date, and a discount amount. Special offers can be expanded in future versions of the system. Users can also delete an existing special offer, to make up for mistakes in the making of a new special offer.

[4]The physical in-place device can be a POS (like pretty much on every vending machine at Politecnico di Milano) or another kind of device that allows payments.

2.4 User characteristics

In this section are described all the various classes of users who are interacting with the system[5].

2.4.1 Consumer (end user)

The consumer is the end user of the eMSP system: s/he's the one that wants to charge an electric vehicle. The consumer can look for a charging station thanks to the use of a map integrated into the mobile application, and when s/he identifies a satisfying one (s/he can look for availability, price...), s/he can book a charge in a specific time slot at that station (if there is still any available). Once the charge is finished, s/he receives a notification from the application on the smartphone or, if preferred, an email. When s/he takes away the vehicle after the charge, s/he can pay for the obtained service, directly at the charging station or through an apposite function on the application.

2.4.2 CPO's authorized user

The CPO's authorized user is the end user of the CPMS system. S/he owns charging stations and can manage the charging columns via the CPMS. S/he can make decisions regarding which DSO to acquire energy from, change the cost of charges, set special offers, and decide how to direct the flow of energy in presence of multiple sources (for example if solar panels or batteries are available). S/he can also monitor the external and internal situation of charging stations, as to infer information about bottlenecks in the management of the system (for example if a station is always crowded it would be wise to consider expanding it with new charging columns) and information regarding the health of components (like in the case of local batteries).

2.5 Domain assumptions

These are all the domain assumptions done in order to write the specifications of this system.

Identifier	Description
D1	Users can connect to the internet and access the service.
D2	Consumers know how to manage the vehicle charging process, plugging and unplugging the vehicle.
D3	Consumers remove their vehicles by the end of the booked charging slot.
D4	Consumers provide true and correct information when registering on the platform.
D5	Every vehicle that uses the system has a certificate used for authenticating it at the charging column and the consumer can access it.
D6	DSOs provide true and correct data to CPOs.
D7	No one abusively parks in the slots.
D8	No physical harm is done to the equipment at the charging stations.
D9	No unauthorized person can alter the state of the charging station (plugging and unplugging cables, opening the stations...).
D10	Batteries are durable/stable during the charging process (both in cars and charging stations).
D11	Charging stations are designed to not have energy shortages, even in case of full load.
D12	All sockets types are compatible, but charge speed is not guaranteed for different types.
D13	Sooner or later the charging process of every vehicle ends (the vehicle is fully charged, or the time slot is ended).
D14	There is a reliable network and structure to connect all the components outside the managed infrastructure.
D15	All payments are handled by a third party and go through correctly (either commit or abort, notifying the result to both the user and the system).
D16	Uniform APIs already exist and are already implemented in the other systems in order to simplify the communication between all the different service providers.
D17	Only users authorized by CPOs can be given credentials to access the CPMS and manage it.
D18	There exists an eRoaming service, already configured on the systems, that allows them to discover each other.

[5] For the sake of simplicity, the obvious superusers (the administrators) of each system are not presented here.

Chapter 3

Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

eMSP The system should be available to the end users in form of a web page and an accessible mobile application (which will be able to receive in-app notifications). In both cases, the user has to be connected to the internet.

CPMS The system should be available to the end users in form of a web page. The ease of use is not of the outermost importance, and it's not to be preferred over a more complete and condensed view. Priority should be given to offering all functionalities in the quickest possible way and showing relevant information in the clearest way. Users are required to know how to use the system to fully be able to operate all functionalities (this can be done through the design of a user manual).

3.1.2 Hardware interfaces

eMSP The eMSP doesn't have any particular hardware constraint, but its frontends should be run on any mobile device and computer (in this case only through a browser). Its backend only needs servers able to respond to the user's queries.

CPMS The only hardware interface needed to access the CPMS is a computer with an installed browser since all functionalities are being offered through a website.

3.1.3 Software interfaces

APIs The communication between the different systems should happen thanks to the use of standard APIs. All the systems described in this document should be also able to communicate with external systems which expose the same APIs.

eRoaming There exists an eRoaming service that is accessible thanks to an API that allows eMSPs to query about available CPOs and their location. It also provides the eMSP with the endpoint address of every selected CPMS (CPO's backend).

OpenStreetMap The eMSP application uses the OpenStreetMap API to provide the map of all the connected charging stations of the various CPOs.

3.2 Functional requirements

3.2.1 Requirements

eMSP requirements

Identifier	Description
R1	The system must allow the user to sign-up.
R2	The system must check the sign-up data.

Identifier	Description
R3	The system must send the user an email for activating the account after a successful sign-up.
R4	The system must discard any account activation request that arrives after 24 hours from the sign-up procedure.
R5	The system must consider any signed-up user who hasn't already activated the account as an unregistered user.
R6	The system must notify the user of the sign-up result, sending an email if it's successful.
R7	The system must allow the user to log in.
R8	The system must check the login data.
R9	The system must allow the user to change his/her password after logging in.
R10	The system must allow any registered user to reset his/her password by sending an email to the specified address.
R11	The system must discard any password change 24 hours after the request.
R12	The system must discard any duplicated usage of the same link for password change after a successful one.
R13	The system must allow the user to logout.
R14	The system must allow the user to set the preferred notification method.
R15	The system must allow the user to add a new vehicle.
R16	The system must allow the user to remove an associated vehicle, only if there is at least one.
R17	The system must allow the user to edit any associated vehicle's details.
R18	The system must allow the user to look for nearby stations.
R19	The system must allow the user to sort the nearby stations according to distance, price, and availability.
R20	The system must allow the user to mark and unmark stations as "favorite".
R21	The system must allow the user to book a charge in a charging station if any slot is available.
R22	The system must ask the user the vehicle s/he wants to charge every time s/he books a charge.
R23	The system must notify the user of any successful booked charge.
R24	The system must allow the user to add any booked charge to the calendar.
R25	The system must allow the user to show all the booked charges, both past, and future (there might be limits).
R26	The system must allow the user to edit or delete a booked charge before the end of the booked slot.
R27	The system must notify the user of the charging slot to use before the charge.
R28	The system must notify the user when the charge is finished.
R29	The system must allow the user to pay for the obtained service.
R30	The system must notify the user after every payment attempt (both in case of success and failure).
R31	The system must periodically contact an eRoaming service for obtaining the list of available CPOs (and respective CPMSs).
R32	The system must be able to connect to any CPMS that offers standard APIs.
R33	The system must be able to communicate with any connected CPMS through standard APIs.
R34	The system must allow any registered user to do everything specified, except signing up.
R35	The system must allow any unregistered user to only sign up and log in.
R36	The system must send the notification only to the user's authenticated devices if the notification preference is by in-app notification.
R37	The system must send emails instead of in-app notifications if that's the user's preferred method or if there is no available user's device.

CPMS requirements

Identifier	Description
R38	The system must check all input data correctness.
R39	The system must allow the user to log in.
R40	The system must allow the user to change his/her password after logging in.
R41	The system must allow any registered user to reset his/her password by sending an email to the specified address.
R42	The system must discard any password change 24 hours after the request.
R43	The system must discard any duplicated usage of the same link for password change after a successful one.
R44	The system must allow the user to logout.
R45	The system must provide an API function to get information about all bookings present for any charging station.
R46	The system must provide an API function to add/delete a booking for a charging point in a charging station.
R47	The system must provide an API function to get information about all charging points and respective sockets in a charging station.
R48	The system must provide an API function to get information about prices relative to a charging station and socket types.
R49	The system must start charging a car when it is first plugged in if it corresponds to the car for which a charge was booked in the corresponding time and place (same charging point/socket as well).
R50	The system must stop charging a car once it recognizes that the car battery has reached full capacity.
R51	The system must stop charging a car if the booking corresponding to that car ends and the car is still plugged in.
R52	The system must automatically select the DSO that offers the best energy price if the "automatic DSO choice" is selected.
R53	If a battery was charging and reaches a full charge, the system must stop charging the battery.
R54	The system must automatically change the energy mix if the "automatic mix choice" is selected.
R55	The system must automatically notify the eMSP of the assignment of a charging column.
R56	The system must automatically notify the eMSP of the end of the car charge.
R57	The system must periodically update the database with new data coming from sensors present in charging stations.
R58	The system must allow the CPMS user to view all bookings present for any charging station.
R59	The system must allow the CPMS user to view all charging points, relative socket types, and if they are occupied or not, for any charging station.
R60	The system must allow the CPMS user to view the batteries' charge levels if present.
R61	The system must allow the CPMS user to create and delete special charge offers.
R62	The system must allow the CPMS user to view all DSO available.
R63	The system must allow the CPMS user to view all DSO prices.
R64	The system must allow the CPMS user to change the energy mix.
R65	The system must allow the CPMS user to start recharging batteries if they are present.
R66	The system must allow the CPMS user to activate and deactivate "automatic DSO choice".
R67	The system must allow the CPMS user to activate and deactivate "automatic mix choice".
R68	The system must allow the CPMS user to modify the automatic DSO payment method.

3.2.2 Goal mapping

eMSP goals

Goal	Domain assumptions	Requirements
G1	D1, D4, D14, D16, D18	R1-R14, R18, R19, R20, R31, R32, R33, R34, R35
G2	D1, D4, D5, D14, D16	R1-R14, R15-R17, R21-R27, R33, R34-R37
G3	D2-D5, D7-D9, D11, D12, D14, D16	R15-R17, R33, R34, R35
G4	D1, D4, D13, D14, D16	R1-R14, R15-R17, R28, R33, R34-R37, R55, R56
G5	D1, D4, D8, D13-D16	R1-R14, R15-R17, R29, R30, R34-R37

CPMS goals

Goal	Domain assumptions	Requirements
G6	D5, D7, D8, D10, D13, D14	R21, R25, R26, R27, R45-R48, R57, R58-R60
G7	D5, D7-D9, D11-D14, D16, D17	R45, R49-R51, R55, R56
G8	D1, D6, D14, D16, D17	R38-R44, R52, R62, R63, R66, R68
G9	D1, D6, D10, D11, D14, D16, D17	R38-R44, R53, R54, R64, R65, R67
G10	D1, D14, D16, D17	R38-R44, R48, R61, R63

3.2.3 Users' use cases diagrams

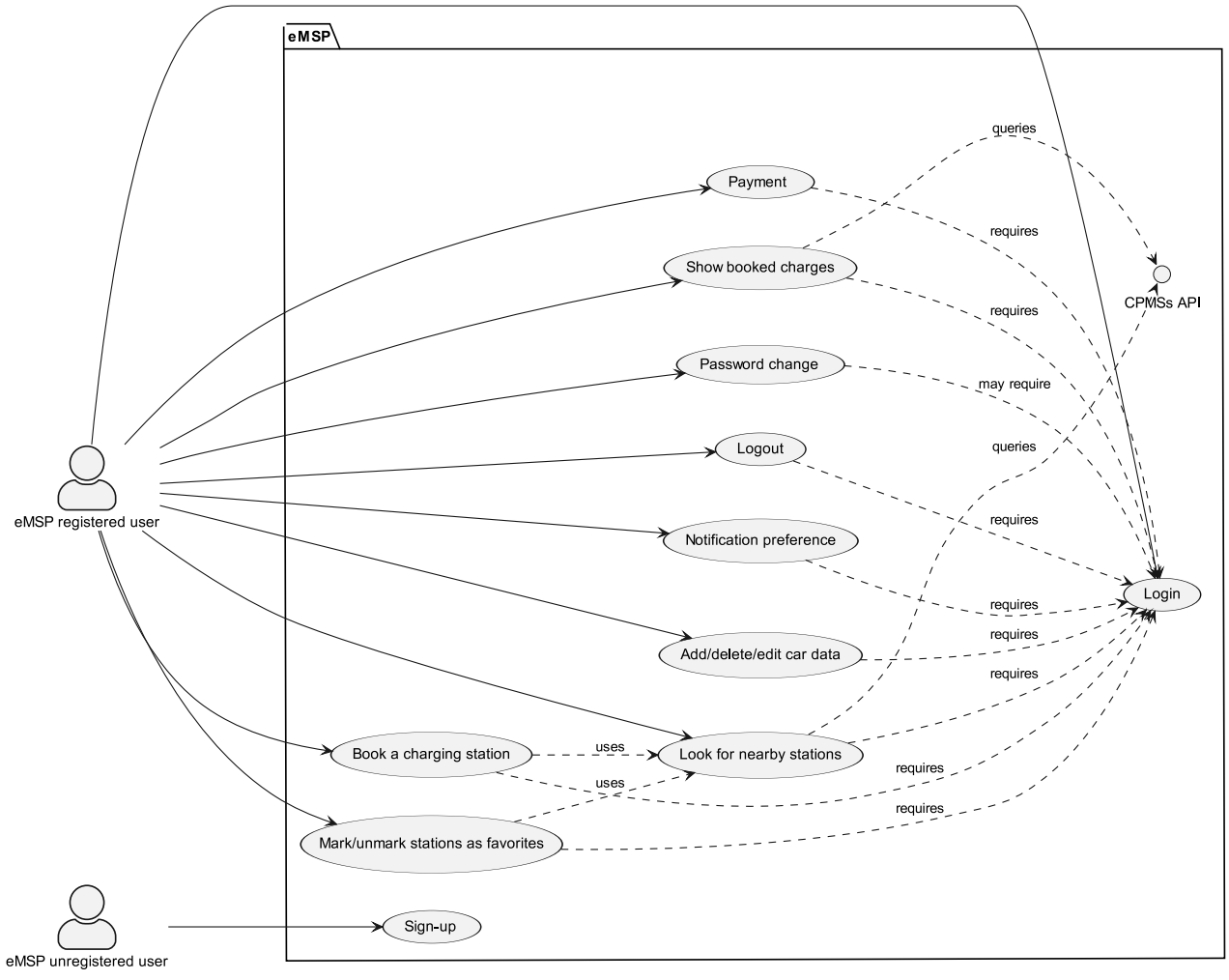

Figure 3.1: eMSP users' use cases.

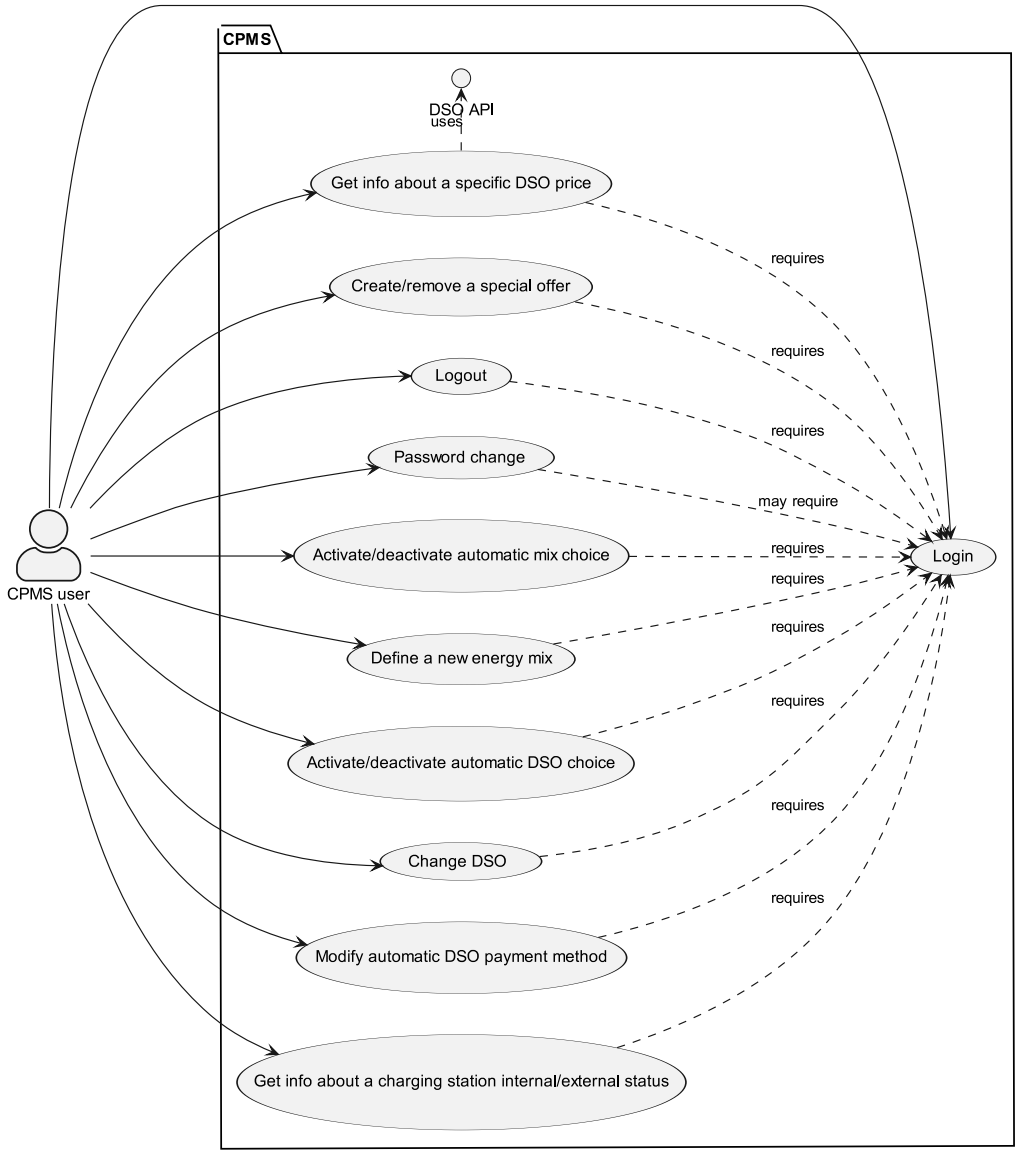

Figure 3.2: CPMS users' use cases.

3.2.4 Use cases

Every time "consumer" or "user" appears in a UML diagram, it's intended that s/he is interacting with the application or web page that interacts with the system.

For the sake of simplicity, many data checks and the opening of the application or web page, according to the use case, are not illustrated, but they are present in the system. Moreover, some cases are not depicted here:

- *Notification preference*: the user is asked to choose between in-app notifications or email ones after his first login. Furthermore, the user can change this preference at any time under his/her profile.
- *Logout*: simply removes the token[1] from the client.
- *Password change*: it works like most of the password changes implemented nowadays. The first option is that the user clicks on the "Forgot password?" link, receives an email with the link for changing it, and changes the password. The second one is that s/he goes to the user's details page and changes the password directly from there. In both cases, s/he has to log back in on all his/her connected devices.

Also, even if it's not specified in the diagrams, if any component fails, the user is notified and the action is automatically aborted, reverting its state to the original one. The same happens in case of user's network failures (which are "detected"[2] through a timeout).

[1] An explanation of the usage of the token can be found in the eMSP's login use case (UC2).
[2] This is a simplification, since in a distributed environment with possible byzantine failures things are a bit more complex.

eMSP | Registration

Identifier	UC1
Actor	Consumer (end user)
Entry condition	The user is not already registered
Event flow	1. The user opens the application or webpage 2. The user fills out the registration form 3. The system checks the inserted data 4. The system sends an email to the user for confirming the account 5. The user clicks on the link for activating the account 6. The system notifies that the account is confirmed
Exit condition	The process ends without errors
Exceptions	• There is already a registered user with that email address • The passwords don't coincide • The user doesn't click on the link in the email within 24 hours
Special requests	The user needs to have access to an email address

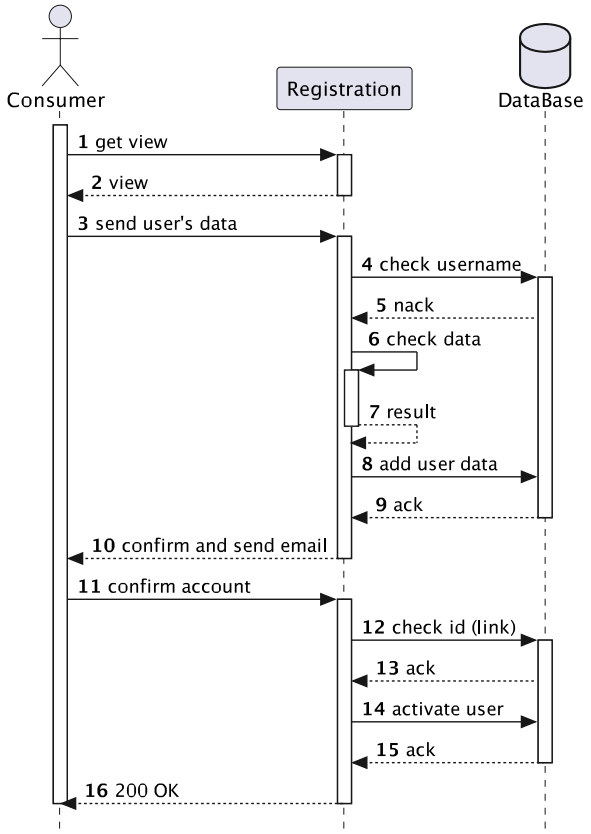

Figure 3.3: registration of the eMSP user (consumer).

eMSP | Login

Identifier	UC2
Actor	Consumer (end user)
Entry condition	The user is already registered
Event flow	1. The user opens the application or webpage 2. The user fills out the login form 3. The system checks the inserted data 4. The system logs in the user sending back a token with all the information
Exit condition	The process ends without errors and the token is sent
Exceptions	• The inserted username corresponds to a non-activated account • The username doesn't exist • The password hash doesn't coincide

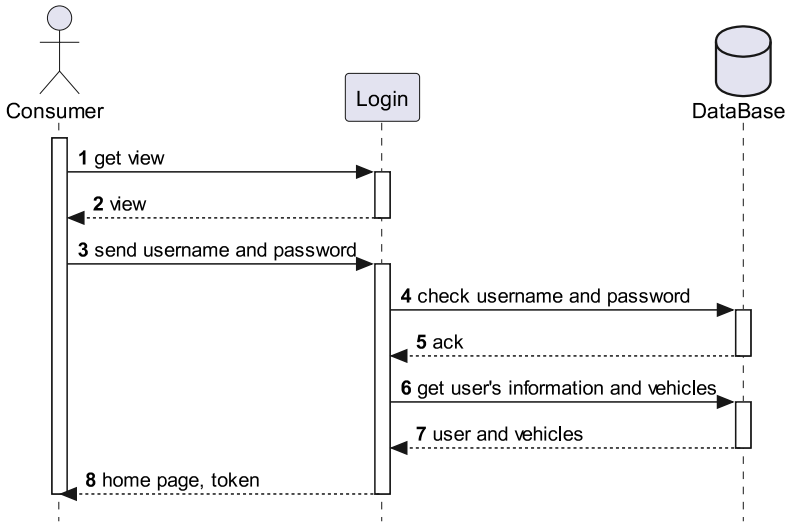

Figure 3.4: login of the eMSP user (consumer).

Please, note that the token present in the scheme is to be intended like a JWT with all the information of the user (which can fit into the token because of their small size) that has to be sent every time. This explains why certain pieces of information are not queried every time they are needed.

eMSP | Vehicles editing

Identifier	UC3
Actor	Consumer (end user)
Entry condition	The user needs to add/remove/edit one or more associated vehicles
Event flow	1. The user opens the list of vehicles 2. The user adds a new vehicle (inserting all the data) or clicks on one 3. If the user clicks on a vehicle, he can choose to edit or remove it 4. The system confirms the operation, sends back an updated token, and updates any future reservation (deleting it if the vehicle is deleted or editing it with the new certificate)
Exit condition	The process ends without errors
Exceptions	• The user tries to remove some information from a vehicle while editing
Special requests	If the user wants to add a new vehicle or to change the certificate of any, s/he has to be able to upload the certificate

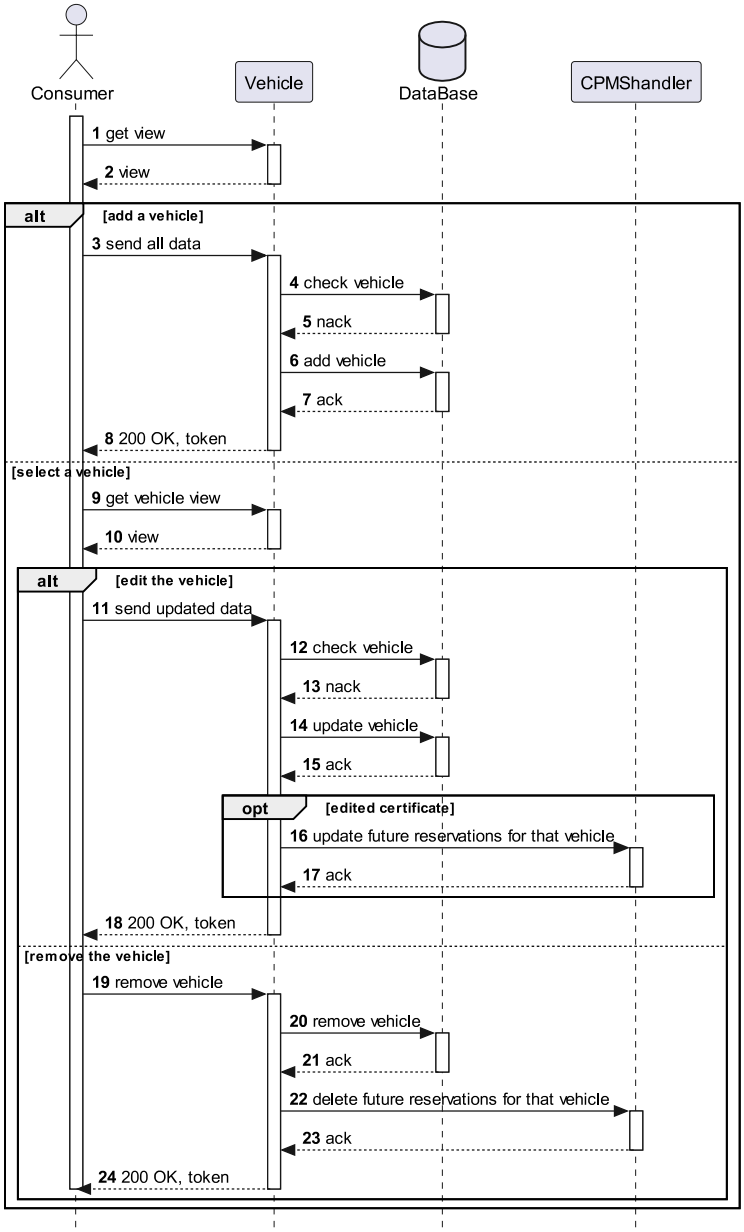

Figure 3.5: the user manages his/her vehicles.

eMSP | Look for nearby stations

Identifier	UC4
Actor	Consumer (end user)
Entry condition	The user is already logged in
Event flow	1. The user opens the map page (if s/he selects the favorites, skips to the last point) 2. The user activates the geolocalization or searches for a place 3. If wanted, the user clicks on the list view (s/he can switch at any time) 4. If the user is in the list view, s/he can sort the stations based on some parameters 5. The user opens the details of the charging station, eventually toggling favorite
Exit condition	The process ends without errors
Exceptions	• The user inserts a nonexistent location

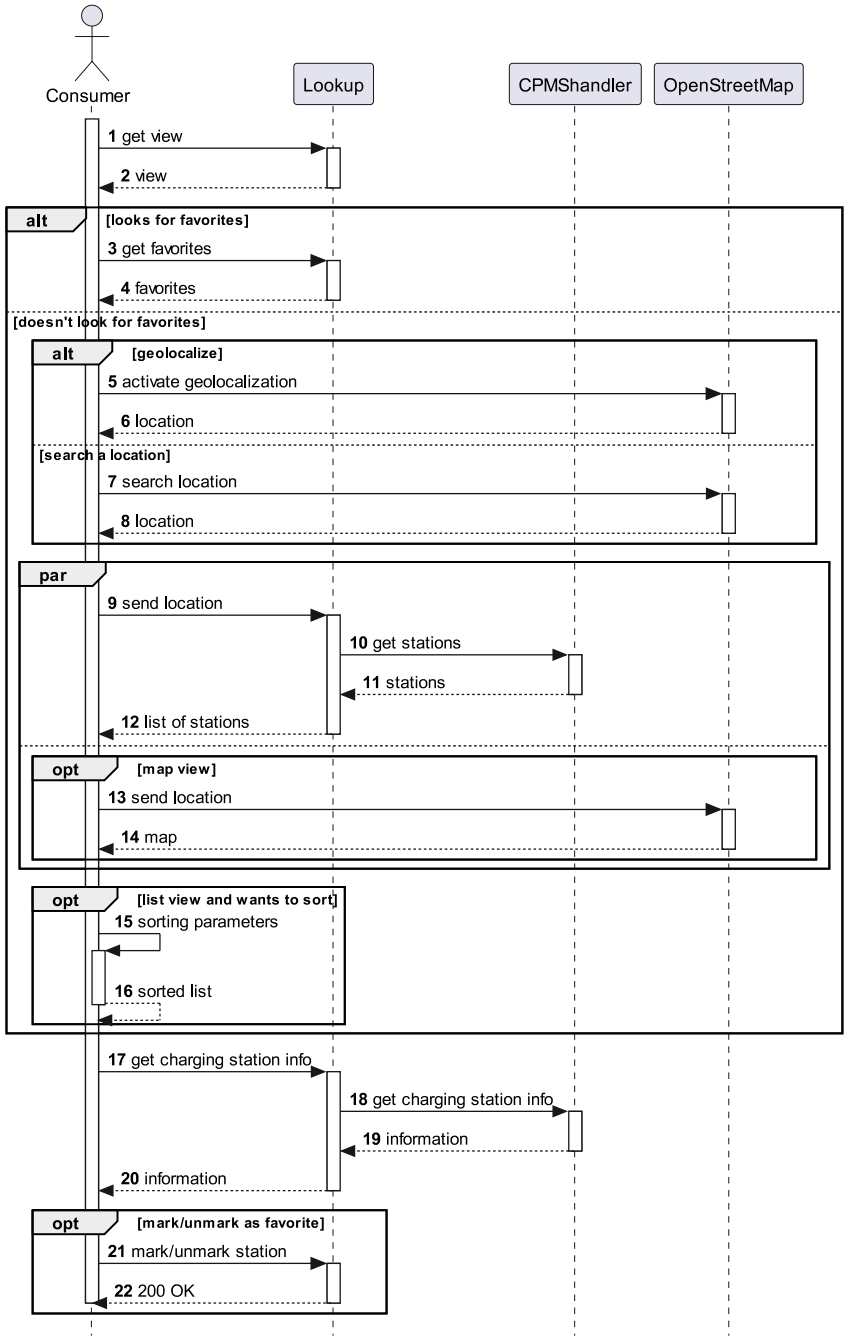

Figure 3.6: the user looks for a charging station and marks/unmarks it as "favorite".

eMSP | Book a charge

Identifier	UC5
Actor	Consumer (end user)
Entry condition	The user has already opened the information of a charging station
Event flow	1. The user opens the book functionality 2. The user selects the vehicle to charge (only if more than one) 3. The user selects some available slots 4. The system notifies the user of the successful operation and sends all the details
Exit condition	The process ends without errors
Exceptions	• The user has no associated vehicle • There are no available slots for that charging station and socket • The user selects a duration that exceeds the maximum duration for that slot • Some other user, meanwhile, has booked even a small part of the selected slot

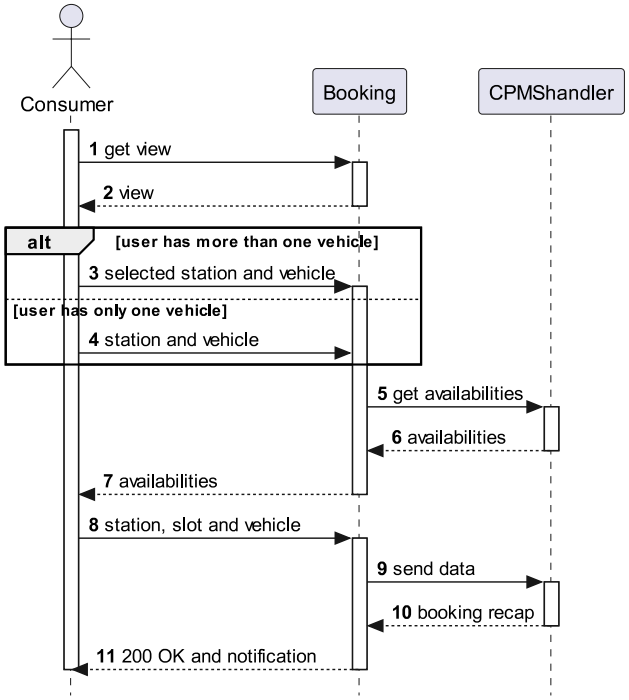

Figure 3.7: the user books a charge.

eMSP | Edit/delete a reservation

Identifier	UC6
Actor	Consumer (end user)
Entry condition	The user has already booked a charge and needs to edit or delete it
Event flow	1. The user accesses the list of future charges 2. The user selects a charge 3. The user edits (following steps 2-4 of UC5) or deletes it 4. The system notifies the change
Exit condition	The process ends without errors
Exceptions	• The user tries to delete a past or current charge • An exception from the booking process ones (UC5) arises

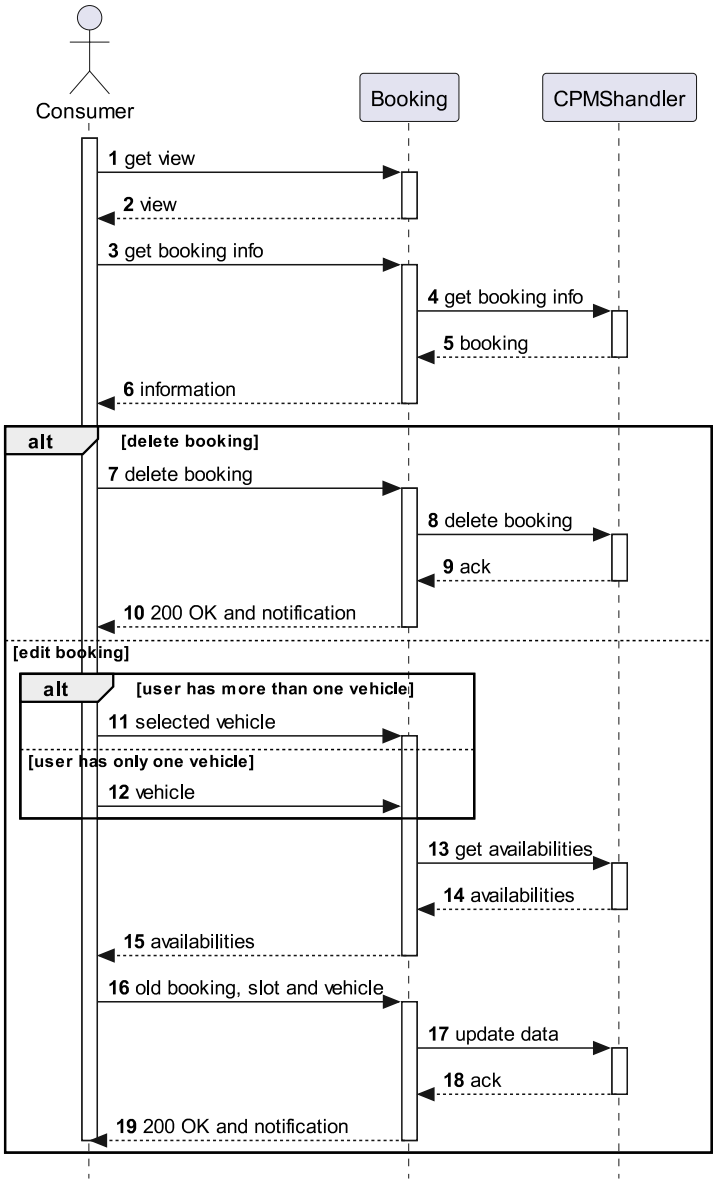

Figure 3.8: the user edits or removes a booked charge.

eMSP | Get notified

Identifier	UC7
Actor	Consumer (end user)
Entry condition	The system is informed about an important fact for the user (assigned socket, end of charge, or payment result)
Event flow	1. The system checks the user's notification preferences 2. The system notifies the user
Exit condition	The notification is sent to the user

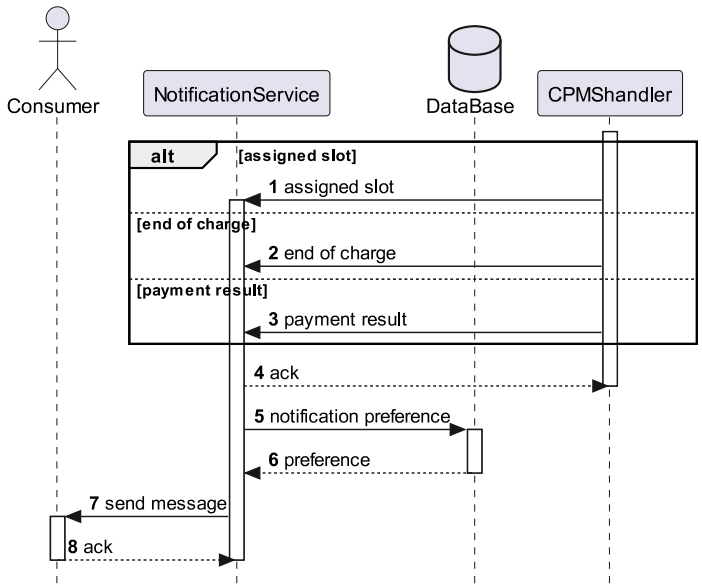

Figure 3.9: the system notifies the user about an important fact from the CPMS.

Since these three cases are close to each other, they are represented together in the same use case.

eMSP | Pay for the obtained service

Identifier	UC8
Actor	Consumer (end user)
Entry condition	After completing a charge, the user has to pay for it
Event flow	1. The user selects the charge to pay 2. The system redirects the user to the payment page 3. The user inserts the data 4. The system confirms the payment and notifies the CPMS and the user
Exit condition	The process ends without errors
Exceptions	• The user inserts wrong payment data
Special requests	The user has a valid payment method

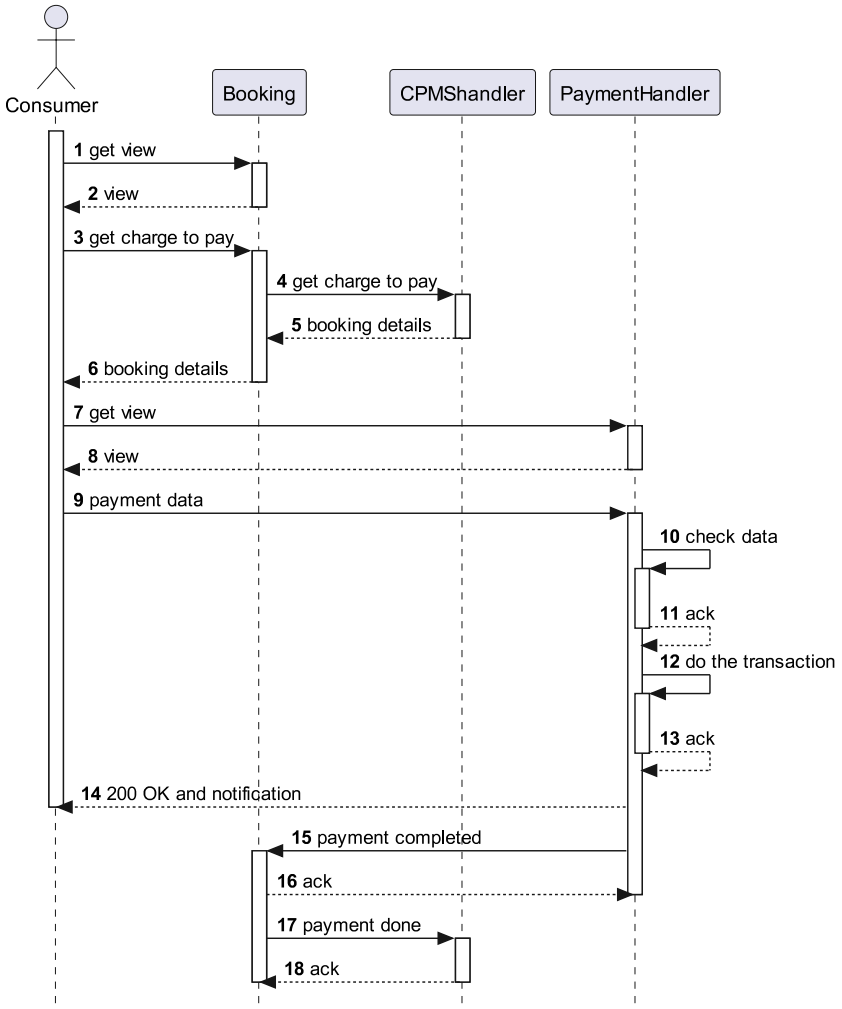

Figure 3.10: the user pays for the charge.

Similar to this, the in-loco payment requires the user to insert his/her payment data (or to use a payment card, according to how it's implemented) and in this case it's the CPMS who notifies the eMSP, and not the other way round.

CPMS | Login

Identifier	UC9
Actor	CPO allowed user
Entry condition	The user is at the login page in the website
Event flow	1. The user enters login data 2. The system checks if input data matches database content 3. The system queries the DBMS for home page data 4. The system sends back the home page with all relevant information
Exit condition	The process ends without errors
Exceptions	• Missing or wrong input data is sent from the user

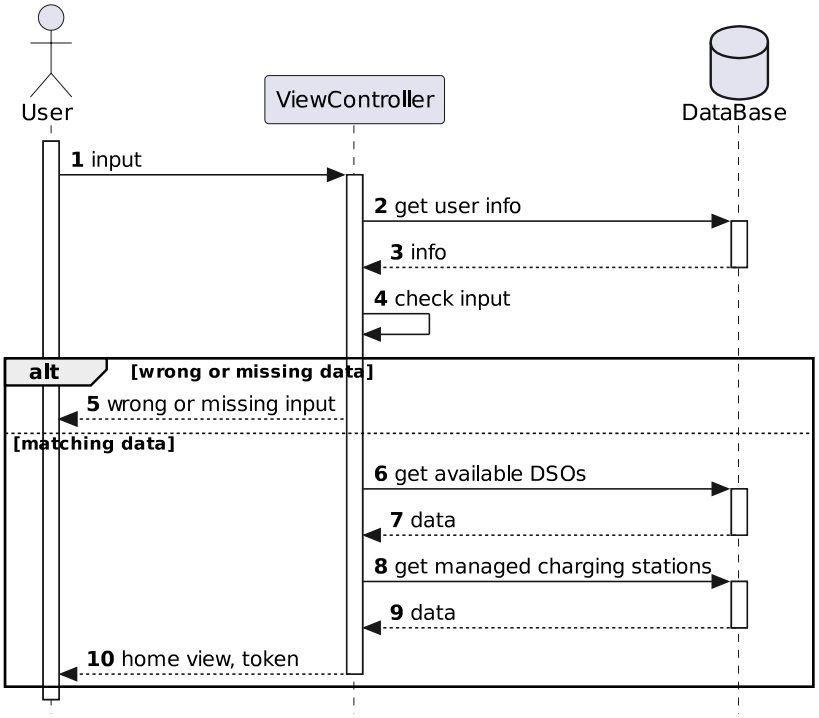

Figure 3.11: login of a registered CPO user.

CPMS | Check charging station status

Identifier	UC10
Actor	CPO allowed user
Entry condition	The user is at the home page in the website
Event flow	1. The user selects a charging station 2. The user selects "Check status" 3. The system queries the DBMS for all relevant information 4. The DBMS returns all relevant information 5. The system sends back the new page with all relevant information 6. The system periodically sends new information to the website
Exit condition	The process ends without errors
Exceptions	• A non-existing charging station is selected • The selected charging station is not managed by the user

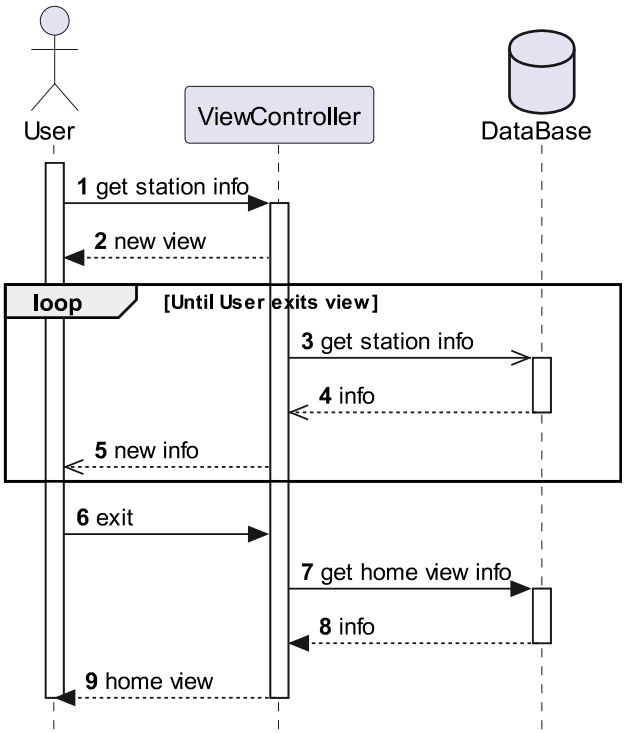

Figure 3.12: change view from the home page to charging station info.

As stated in requirement R57, the system periodically has to update the database with information regarding all changing parameters coming from charging stations. These include the station's battery levels, incoming energy from the solar panels, occupied columns in the station, and eventually more data. This allows the view controller to simply fetch data from the database, instead of intercepting all data streams, to communicate to the website in the corresponding views.

CPMS | Change energy mix

Identifier	UC11
Actor	CPO allowed user
Entry condition	The user is at the home page in the website
Event flow	1. The user selects "Change energy mix" 2. The user defines a new energy mix 3. The system communicates the new energy mix to the charging station 4. The charging station communicates the components the change
Exit condition	A new energy mix is defined without errors, the user is notified
Exceptions	• The new energy mix could not be attained • The database could not be updated • The user could not be notified
Special requests	The user has to input a reasonable energy mix

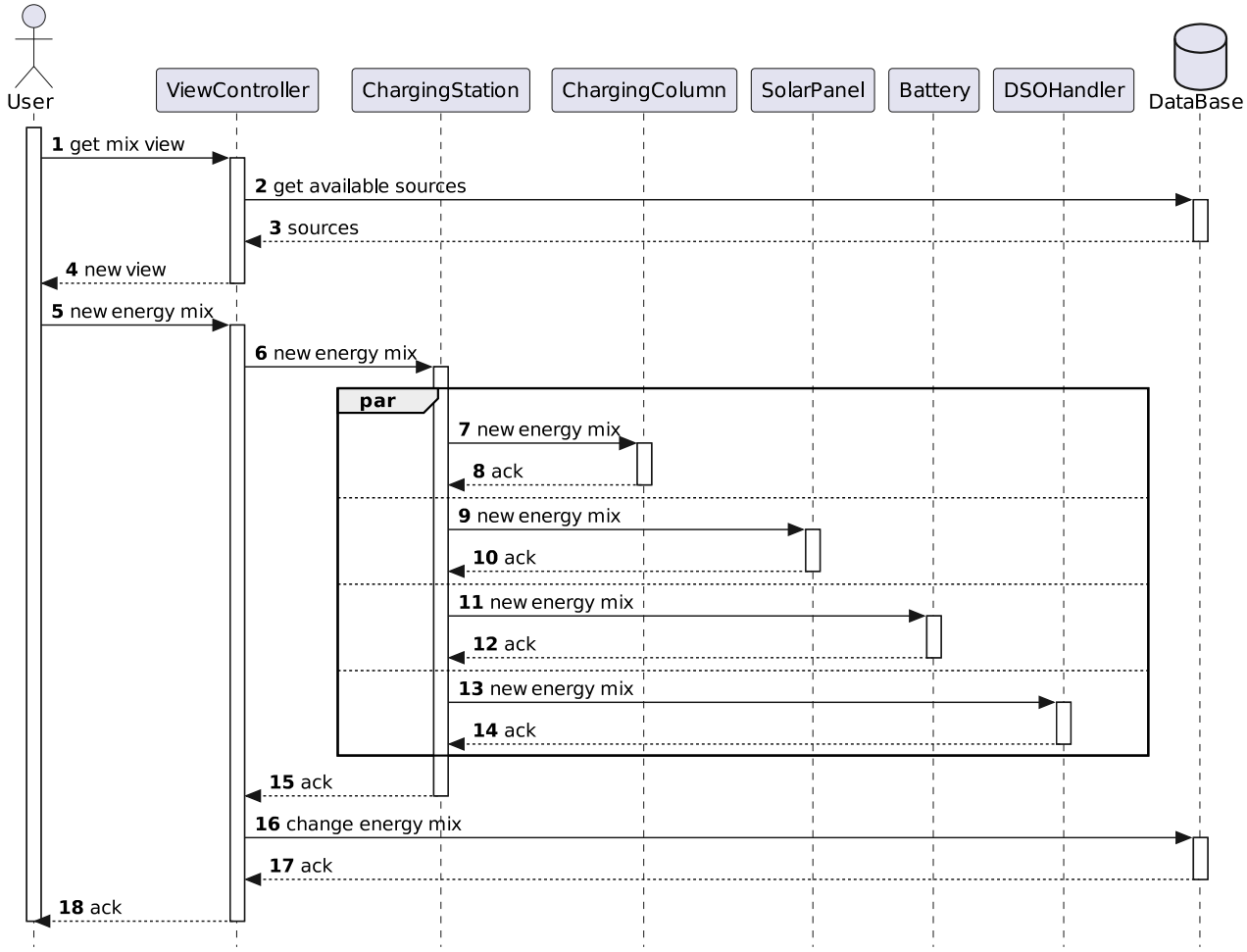

Figure 3.13: user changing energy mix of a charging station.

CPMS | Manage DSO choice

Identifier	UC12
Actor	CPO allowed user
Entry condition	The user is at the home page in the website
Event flow	1. The user selects a charging station 2. The user selects "Available DSO" 3. The system sends a new DSO view 4. The user selects a DSO 5. The system starts using the selected DSO in the charging station
Exit condition	The DSO is being used in the charging station and the user is notified
Exceptions	• The DSO stops being available while the choice is being made

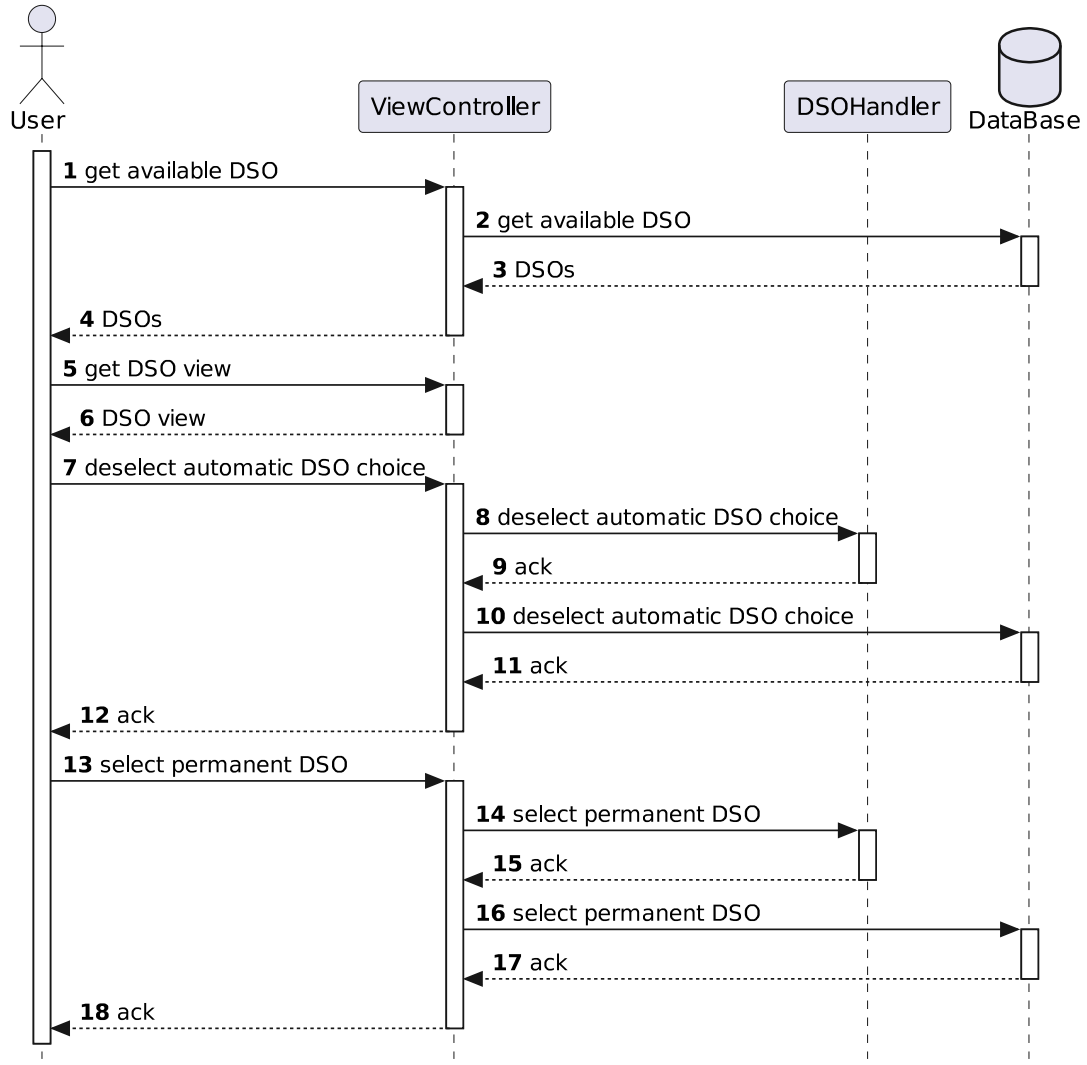

Figure 3.14: Caption

Instead of selecting a specific DSO, the user at the DSO view can also choose to activate/deactivate "Automatic DSO choice", letting the system handle the DSO choice automatically or selecting the current DSO as permanent. In this view, it is also possible to modify the automatic payment method.

CPMS | Create Special offer

Identifier	UC13
Actor	CPO allowed user
Entry condition	The user is at the home page in the website
Event flow	1. The user selects "Special offers" 2. The user selects "Create new special offer" 3. The user inputs all the needed parameters 4. The system creates a new special offer
Exit condition	A new special offer is created
Exceptions	• Missing or wrong parameters

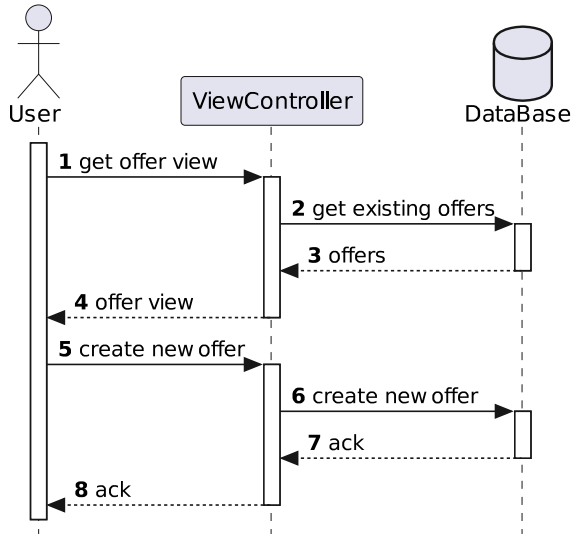

Figure 3.15: a user creating a new special offer. In the special offer view users also see existing offers, and delete them if needed.

3.3 Performance requirements

eMSP The eMSP system should be generally responsive. It should provide answers to simple queries in less than 5 seconds, while more complex ones should complete in less than 30 seconds. If the system is directly interfacing with the user and the response isn't available after a couple of seconds, the frontend should display a loading indicator. A lower waiting time can be archived thanks to a proper database configuration and through the redundancy of components and data. The application performs much of the required computations on the backend, thus letting the application and the web interface only display the contents and doing some basic operations (sorting, checks...).

CPMS Since the CPMS system is more geared towards monitoring, and eventually allows for some actions from the user, availability should be preferred over reliability. It's not essential that the system does not crush (although it is preferred), since all changes are immediately saved in permanent storage. It's more important that if the system crushes it can recover quickly, and all inputs are delivered as soon as possible and made permanent in the database storage. All the computation will take place on the servers of the system, which have to be replicated and distributed to better assert the desired availability and reach a consensus over the input received.

3.4 Design constraints

3.4.1 Standards compliance

Every system should correctly implement all the needed standard APIs for allowing interconnection with different pieces of software that expose those functionalities. Moreover, each system should be compliant with common payment APIs for managing user payments.

3.4.2 Hardware limitations

eMSP For accessing the system, the user can use both the application which should be available in all the main mobile stores and should run on most of the devices present nowadays (e.g. an Android version greater or equal to 4.4 KitKat) or a browser which supports at least ES6 for correctly displaying all the views of the website.

CPMS For accessing the system, any computer/laptop/tablet/phone with an installed web browser, connected to the internet, is sufficient, as a website is the sole user access point.

3.5 Software system attributes

3.5.1 Reliability

eMSP The eMSP should be reliable, and even in case of failures, it should recover quickly. If downtimes, even if small, are frequent, this might make the user experience painful because of probably many failed requests.

CPMS The CPMS website should be reliable enough to let users monitor and perform actions. Reliability is not to be preferred over availability. The CPMS servers on the other hand can be replicated to allow for higher reliability and protection of the data.

3.5.2 Availability

eMSP The eMSP should be available 99% of the time (about 361 days per year). Any longer unavailability period might encourage the user to move to some other competitor.

CPMS The CPMS should be available 99.7% of the time (about 364 days per year). Such a high availability is needed for the server to be responsive toward all end users, for better monitoring, and to secure that input actions are almost always registered.

3.5.3 Security

eMSP The eMSP should provide a high level of security for the information the users insert. Especially, the system must protect all the uploaded vehicles' information.

CPMS The CPMS should provide a high level of security for the information the charging station dispatches. In particular, the system needs to be aware of "man in the middle" attacks where information coming from the stations is tampered with, which would result in incorrect automatic choices and eventually wrong information dispatched to the eMSP. Attention also should be paid to securing charging columns from possible manumissions in the validation of cars' certificates.

3.5.4 Maintainability

The system should follow basic design guidelines to make the whole system more maintainable. This will be described more deeply in the Design Document, but a general indication is to divide components into smaller ones, so that each one can do only a specific action, thus making the system easier to bring up and maintain, allowing also the maintenance of single components at a time.

3.5.5 Portability

eMSP The backend should be able to run well on Linux systems, and it should be optimized to use all the standard Linux and POSIX features. The website should instead work well on all the major browsers, while the mobile application should be able to run on many architectures, allowing it to work on all the supported devices, as described in 3.4.2 Hardware limitations (page 33).

CPMS The backend should be able to run well on Linux systems, and it should be optimized to use all the standard Linux and POSIX features. The database system should be distributed for higher availability.

Chapter 4

Formal Analysis

4.1 Alloy code

This section provides all the alloy code used to prove the correctness of the project. For a better reading, the code is subdivided into the different blocks it's composed.

4.1.1 Entities

```
// +-------------------------------------------+
// |                  General                  |
// +-------------------------------------------+

// Email address
sig Email {}

// Password
sig Password {}

// Booleans
abstract sig Boolean {}
lone sig True extends Boolean {}
lone sig False extends Boolean {}

// Date time, represented as an integer
sig DateTime {
    time: one Int
} {
    time >= 0
}

// Global time
lone sig GlobalTime {
    time: one DateTime
}

// +-------------------------------------------+
// |                  Common                   |
// +-------------------------------------------+

// eMSP user's vehicle
sig Vehicle {
    socket: one Socket,
    certificate: one Certificate
}
```

```
// Vehicle certificate
sig Certificate {}

// Sockets
abstract sig Socket {}
lone sig SocketSlow extends Socket {}
lone sig SocketFast extends Socket {}
lone sig SocketRapid extends Socket {}

// eMSP user's booking
sig Booking {
    chargingColumn: one ChargingColumn,
    start: one DateTime,
    end: one DateTime,
    vehicle: one Vehicle,
    user: one eMSPuser
} {
    start.time < end.time
}

// +-------------------------------------------+
// |                    eMSP                   |
// +-------------------------------------------+

// eMSP user
sig eMSPuser {
    username: one Email,
    password: one Password,
    devices: set Device,
    vehicles: set Vehicle,
    favorites: set ChargingStation
}

// eMSP user's device
sig Device {}

// +-------------------------------------------+
// |                    CPMS                   |
// +-------------------------------------------+

// CPMS
sig CPMS {
    users: set CPMSuser,
    chargingStations: some ChargingStation,
    bookings: set Booking
}

// CPMS user
sig CPMSuser {
    username: one Email,
    password: one Password,
    cpms: one CPMS,
    CSManaged: some ChargingStation
}
```

```
// CPO's charging station
sig ChargingStation {
    location: one Location,
    chargingColumns: some ChargingColumn,
    mix: one EnergyMix,
    dso: one DSO,
    availableDSOs: some DSO,
    automaticDsoChoice: some Boolean
} {
    dso in availableDSOs
}

// Charging station's column
sig ChargingColumn {
    chargingStation: one ChargingStation,
    socket: one Socket,
    occupied: one Boolean
}

// Location
sig Location {}

// CPMS energy mix
sig EnergyMix {}

// DSO
sig DSO {
    price: one Int
} {
    price > 0
}

// Energy source
abstract sig EnergySource {
    chargingStation: one ChargingStation,
    isBeingUsed: one Boolean
}

// Battery as the energy source
sig Battery extends EnergySource {
    chargeLevel: one Int,
    isCharging: one Boolean
} {
    chargeLevel >= 0 and chargeLevel <= 100
}

// Solar panel as the energy source
sig SolarPanel extends EnergySource {}

// DSO as the energy source
sig DSOenergy extends EnergySource {
    DsoUsed: one DSO
}
```

4.1.2 Facts

```
// +-------------------------------------------+
// |                   Common                  |
// +-------------------------------------------+

// Every email address has an owner
fact OwnedEmail {
    all e: Email | (
        (one u: eMSPuser | u.username = e)
        or
        (one u: CPMSuser | u.username = e)
    )
}

// No two distict eMSPusers or CPMSusers have the same username (email address)
fact OneUsername {
    (no disj u1, u2: eMSPuser | u1.username = u2.username)
    and
    (no disj u1, u2: CPMSuser | u1.username = u2.username)
}

// Every password is associated with at least one user
fact OwnPassword {
    all p: Password | (
        (some u: eMSPuser | u.password = p)
        or
        (some u: CPMSuser | u.password = p)
    )
}

// Every date time is used in a booking or is the global time
fact SomeDateTime {
    all d: DateTime |
        (some b: Booking |
            b.start = d
            or
            b.end = d)
        or
            d.time = Now[]
}

// No two date time represent share time
fact NoCommonTime {
    all d: DateTime |
        all d1: DateTime |
            d.time = d1.time
            implies
            d = d1
}

// Every socket is used somewhere
fact SocketUsed {
    all s: Socket | (
        (some cc: ChargingColumn | cc.socket = s)
        or
        (some v: Vehicle | v.socket = s)
    )
}
```

```
// +-------------------------------------------+
// |                    eMSP                   |
// +-------------------------------------------+

// Every device has only one owner
fact OwnedDevice {
    all d: Device |
        one u: eMSPuser |
            d in u.devices
}

// Every vehicle has only one owner
fact EveryVehicleIsOwned {
    all v: Vehicle |
        one u: eMSPuser |
            v in u.vehicles
}

// Every certificate is associated with only one vehicle
fact OneCertificateOneVehicle {
    all c: Certificate |
        one v: Vehicle |
            c = v.certificate
}

// +-------------------------------------------+
// |                    CPMS                   |
// +-------------------------------------------+

// Every charging station is associated to a CPMS
fact OneChargingStationOneCPMS {
    all cs: ChargingStation |
        one cpms: CPMS |
            cs in cpms.chargingStations
}

// Charging stations managed by a CPMS user are owned by that CPMS
fact ChargingStationCPMS{
    all u: CPMSuser |
        u.CSManaged & u.cpms.chargingStations = u.CSManaged
}

// Every location is associated with only one charging station
fact OneLocationOneChargingStation {
    all l: Location |
        one c: ChargingStation |
            c.location = l
}

// A charging column is associated with only one charging station
fact ChargingColumnOneStation {
    no disj c1, c2: ChargingStation |
        some cc: ChargingColumn |
            cc in (c1.chargingColumns & c2.chargingColumns)
}
```

```
// The charging column's station contains that column
fact ChargingStationColumn {
    all c: ChargingColumn |
        c in c.chargingStation.chargingColumns
}

// No battery can be still charging if its level is 100%
fact ChargingBattery {
    all b: Battery |
        b.isCharging = True
        implies
        b.chargeLevel < 100
}

// If automatic DSO choice is enabled, the DSO with the lowest price is selected
fact AutomaticDsoChoice {
    all cs: ChargingStation |
        cs.automaticDsoChoice = True
        implies
        all d: DSO |
            d in cs.availableDSOs
            implies
            cs.dso.price <= d.price
}

// Every DSO has to be associated with at least one charging station
fact OneDSOOneOrMoreChargingStation {
    all d: DSO |
        some c: ChargingStation |
            c.availableDSOs & d = d
}

// Every energy mix has to be associated with one charging station
fact OneEnergyMixOneChargingStation {
    all e: EnergyMix |
        some c: ChargingStation |
            c.mix = e
}

// A DSO energy associated to a charging station uses the dso used by that station
fact DSOUsed {
    all cs: ChargingStation, d: DSOenergy |
        d.chargingStation = cs
        implies
        d.DsoUsed = cs.dso
}

// All charging stations have a DSO energy
fact ExactlyDSOEnergyPerChargingStation {
    all cs: ChargingStation |
        one d: DSOenergy | d.chargingStation = cs
}
```

```
// All charging station have some energy sources that are being used
fact ActiveSourcesPerStation {
    all cs: ChargingStation |
        some e: EnergySource |
            e.chargingStation = cs
            and
            e.isBeingUsed = True
}

// +-----------------------------------------+
// |                 Bookings                |
// +-----------------------------------------+

// Every booking is associated to a CPMS
fact BookingCPMS {
    all b: Booking |
        one c: CPMS |
            b in c.bookings
}

// Every booking is associated to a charging column managed by the CPMS
fact BookingRightCPMSassociation {
    all b: Booking, c: CPMS |
        b in c.bookings
        implies
        b.chargingColumn.chargingStation in c.chargingStations
}

// The same column can't have two overlapping bookings
fact BookNoOverlapOnColumn {
    all disj b1, b2: Booking |
        b1.chargingColumn = b2.chargingColumn
        implies
        b1.start.time > b2.end.time or b2.start.time > b1.end.time
}

// The same vehicle can't have two overlapping bookings
fact BookNoOverlapOnVehicle {
    all disj b1, b2: Booking |
        b1.vehicle = b2.vehicle
        implies
        b1.start.time > b2.end.time or b2.start.time > b1.end.time
}

// The user has to book a column that is compatible with the vehicle's socket
fact BookedRightColumn {
    all b: Booking |
        b.chargingColumn.socket = b.vehicle.socket
}

// For every booked charge, the vehicle is owned by the user who booked
fact BookedChargeVehicleOwner {
    all b: Booking |
        b.vehicle in b.user.vehicles
}
```

```
// If a charging column is occupied, there exists a booking that occupies it at the given
// global time
fact ChargingColumnOccupied {
    all c: ChargingColumn |
        c.occupied = True
        implies
        one b: Booking |
            b.chargingColumn = c
            and
            b.start.time <= Now[]
            and
            b.end.time >= Now[]
}
```

4.1.3 Functions

```
fun Now[]: Int {
    GlobalTime.time.time
}
```

4.2 Alloy generated worlds

Here there are some generated worlds from the alloys source that show some functionalities of the system. Each one of them has a brief description of the depicted situation, the predicate used for showing it, and the generated diagram.

4.2.1 Ongoing charge of a vehicle

This diagram shows the charging process of a vehicle, pointing out the occupation of that charging column by the user's vehicle, while the other one is free.

```
pred OngoingCharge {
    #Device = 1
    #eMSPuser = 1
    #CPMSuser = 0
    #Vehicle = 1
    #DSO <= 1
    #ChargingStation = 1
    #ChargingColumn = 2
    #Booking >= 2
    some c: ChargingColumn | c.occupied = True
}
```

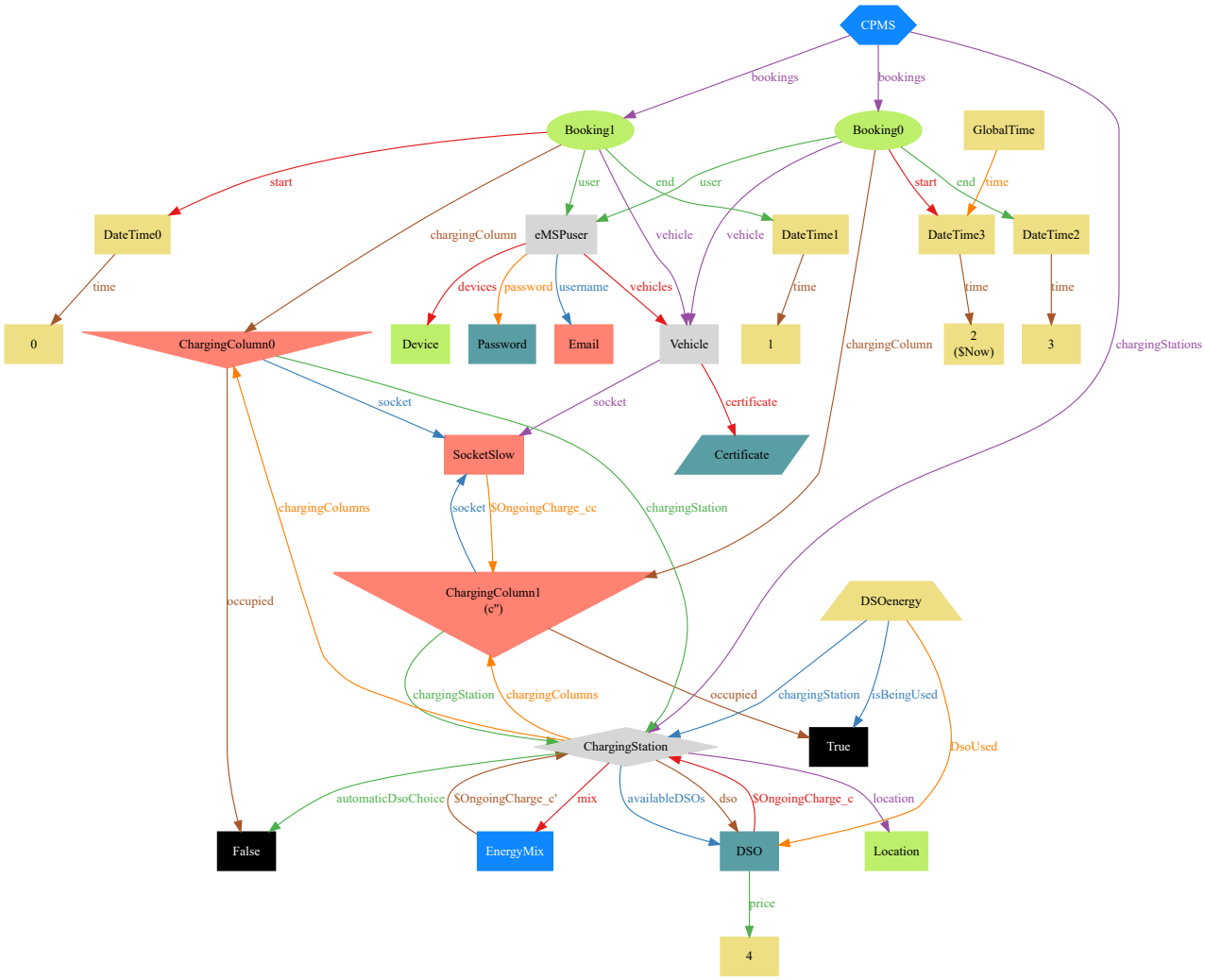

Figure 4.1: the ongoing charge of a vehicle at the charging column.

4.2.2 Creation of a booking

This diagram shows the creation of the first booking by an eMSP user (which is not depicted here) at a specific charging station. The relative column is associated automatically by the system.

```
pred BookCharge[c, c': CPMS, b: Booking] {
    c'.users = c.users
    c'.chargingStations = c.chargingStations
    c'.bookings = c.bookings + b
    #Device = 0
    #ChargingColumn = 1
    #DSO = 1
    #eMSPuser = 1
    #CPMSuser = 0
}
```

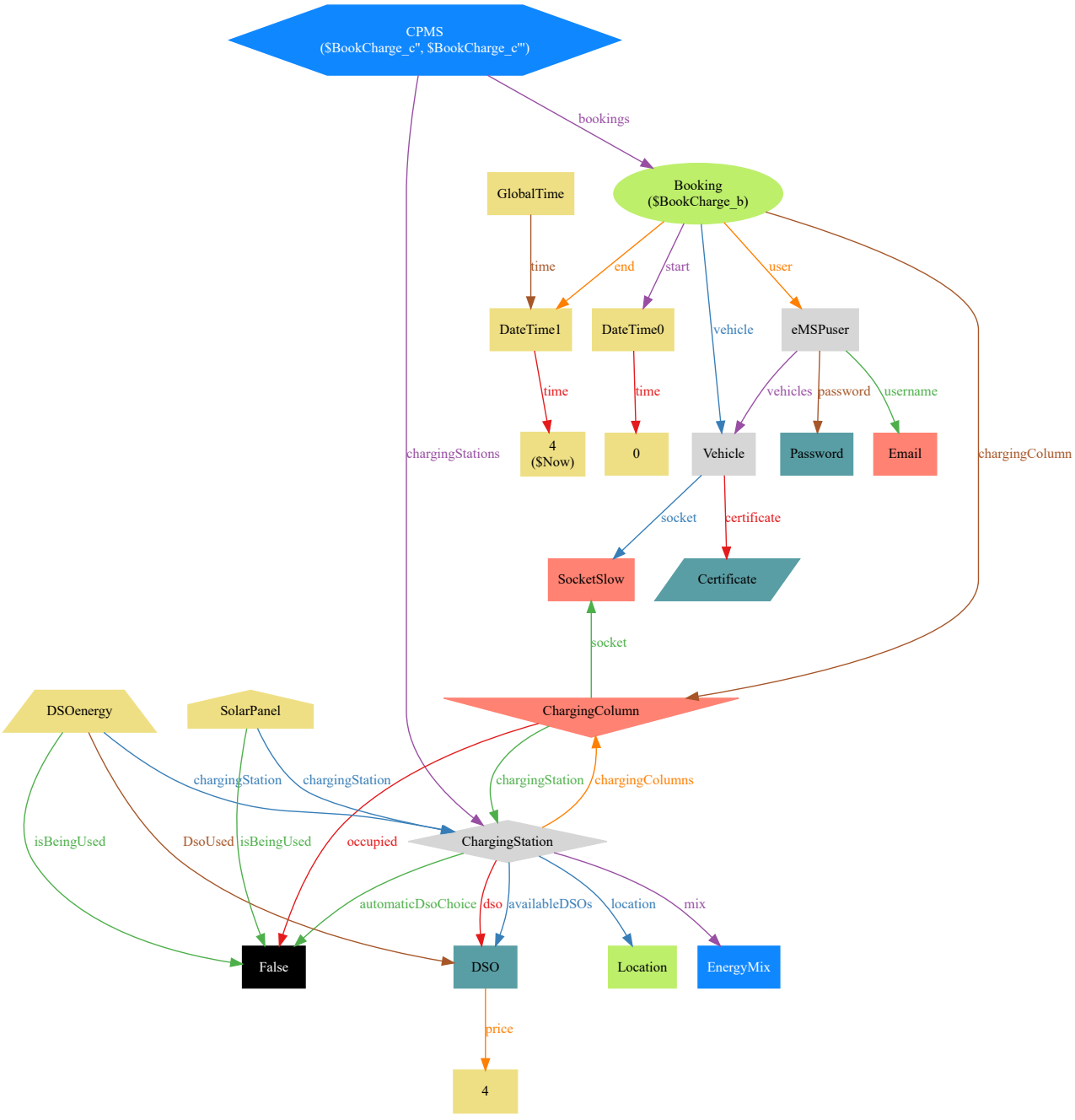

Figure 4.2: the creation of a new booking.

4.2.3 Addition of a vehicle

This diagram shows the addition of a new vehicle by the eMSP user, connecting it to his account, thus making it available when booking a new charge.

```
pred AddVehicle[u, u': eMSPuser, v: Vehicle] {
    u'.username = u.username
    u'.password = u.password
    u'.devices = u.devices
    u'.vehicles = u.vehicles + v
    u'.favorites = u.favorites
    #GlobalTime = 0
    #CPMS = 0
    #Device = 1
    #Vehicle = 2
}
```

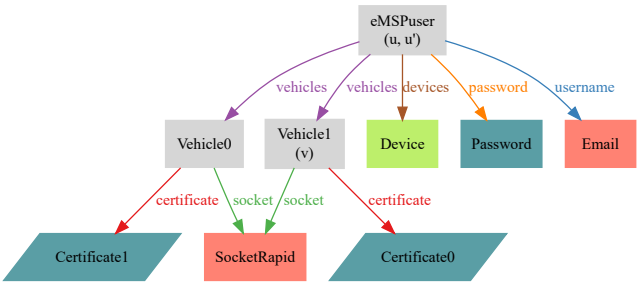

Figure 4.3: the addition of a new vehicle by the eMSP user.

4.2.4 Charging station sources

This diagram shows that a charging station can have multiple energy sources, but that according to the energy mix, some of them may not be used at a specific point in time.

```
pred Sources {
    #eMSPuser = 0
    #CPMS = 1
    #CPMSuser = 0
    #ChargingStation = 1
    #EnergySource = 6
    #GlobalTime = 0
    some d: DSOenergy | d.isBeingUsed = False
}
```

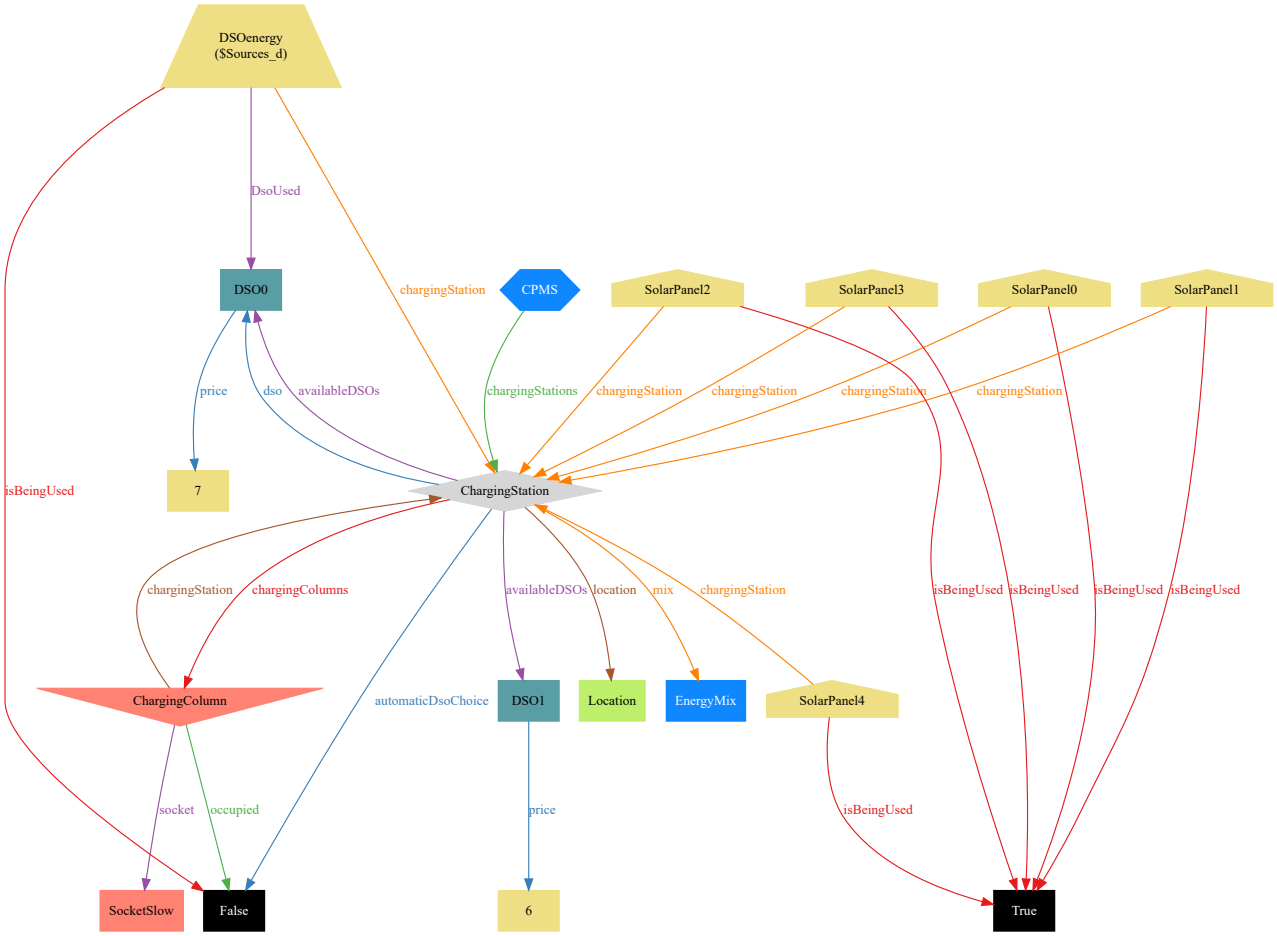

Figure 4.4: the energetic situation at the charging station.

4.2.5 Automatic DSO choice

This diagram shows the choice of a DSO from which buying the energy for supplying the charging stations which have in their energy mix the energy coming from the DSO.

```
pred AutomaticDSOChoice {
    #ChargingStation = 2
    #DSO > 2
    #eMSPuser = 0
    some cs: ChargingStation | cs.automaticDsoChoice = True
    some cs: ChargingStation | cs.automaticDsoChoice = False
    all cs: ChargingStation | #cs.availableDSOs > 1
}
```

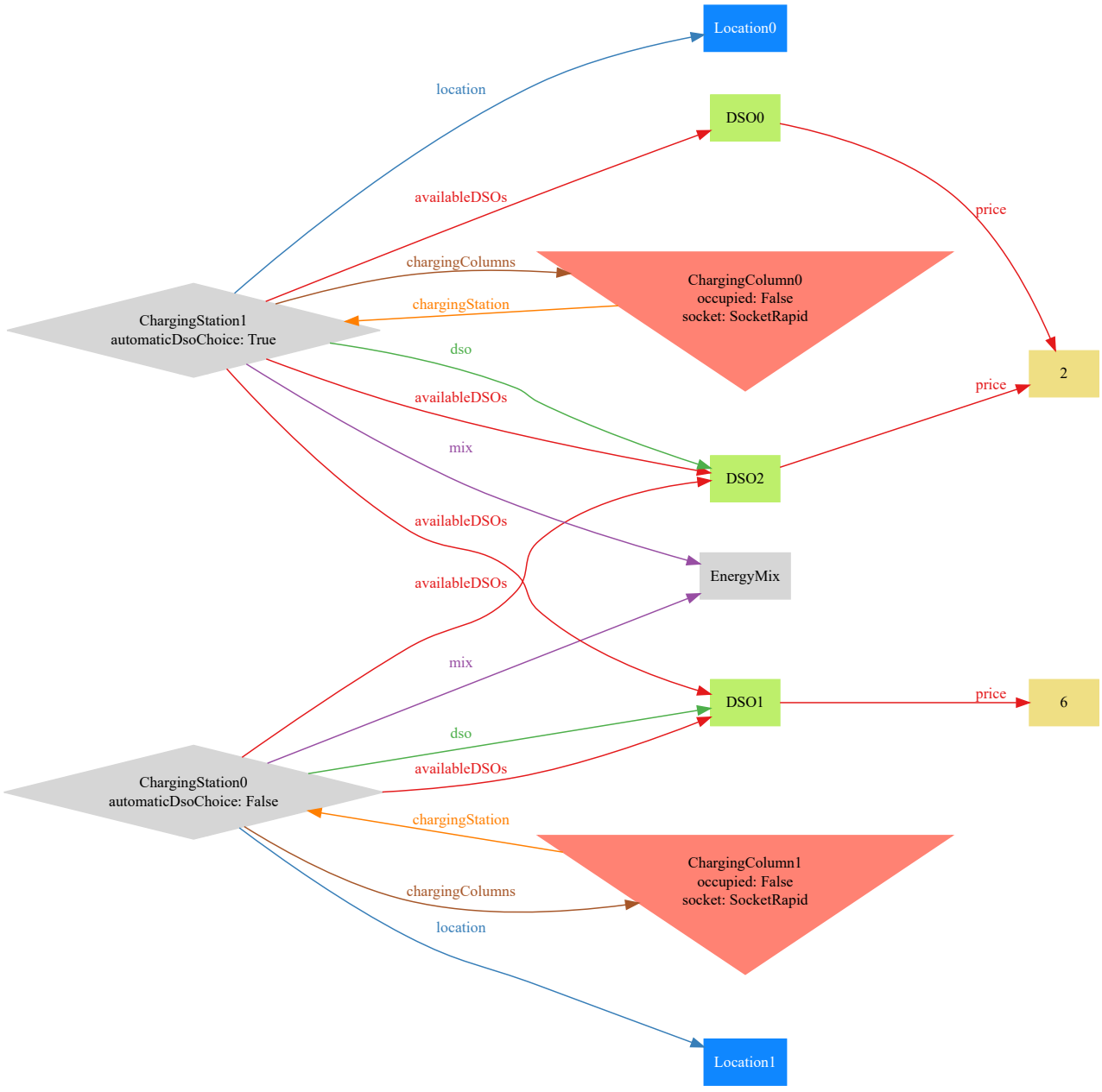

Figure 4.5: the automatic choice of a DSO.

4.2.6 A rather complete world

This diagram shows a world which tries to be as complete as possible.

```
pred GenericWorld {
    #Email = 3
    #Password = 4
    #eMSPuser = 2
    #Vehicle >= 1
    #Device >= 1
    #CPMS = 1
    #CPMSuser = 2
    #ChargingStation >= 2
    #Booking >= 2
    #DSO >= 2
    #EnergySource >= 3
    no u: eMSPuser | #u.vehicles = 0
    some c: ChargingColumn | c.occupied = True
}
```

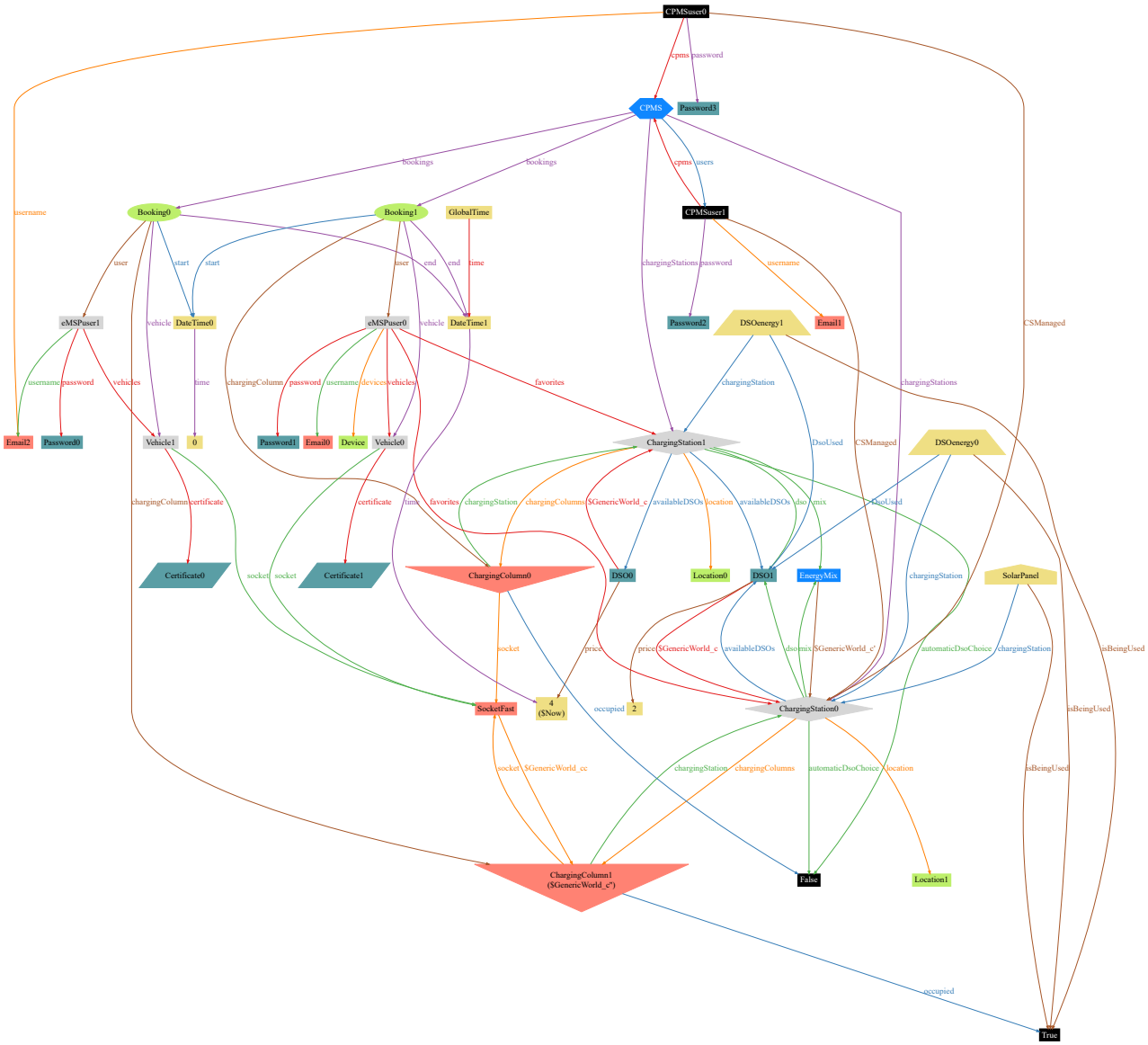

Figure 4.6: a generic, but rather complete, world.

Chapter 5

Conclusions

5.1 Final thoughts

This document was redacted following the guidelines for the project (*eMall - e-Mobility for All project*). Discrepancies between our document and the guidelines are to be considered our hypothesis on how the two systems should behave or be modeled.

Also, these systems have been designed having in mind possible future expansions. Thus, anyone who wants to expand this project is welcome and encouraged to do so.

5.2 Credits

For writing this document, we used different pieces of software from the Internet. This is a rather comprehensive list of them:

- *LaTeX* (with packages) and *Visual Studio Code* (with plugins) for writing the document.
- *Git*, *GitHub* and *Notion* for keeping things organized.
- *draw.io* and *PlantUML* for creating the diagrams.

5.3 Effort

Task	Riccardo Motta	Pierluigi Negro
Starting off	1h	2h
Defining goals and phenomena	2h	2h
Defining scenarios, functions, users, and domain assumptions	3h	3h
Creating class diagram	3h	2.5h
Defining requirements	2h	2h
Goal mapping	2h	2h
Creating use cases	8h	6h
Alloy writing	10.5h	10.5h
Document writing	16h	17h
Updating document for Version 1.1	0.5h	0h
Total	**48h**	**47h**