

IoT - Smart Charging System

Matteo La Licata (M. 4855798)

Filippo Siri (M. 4819642)

Claudio Stana (M. 4819653)

February 2024

Contents

1	Introduction	3
2	OCPP	3
3	Client	3
3.1	Description	3
3.2	From Client To Server	3
3.2.1	Authorize	3
3.2.2	Boot Notification	3
3.2.3	Heartbeat	4
3.2.4	Meter Values	4
3.2.5	Start Transaction	4
3.2.6	Status Notification	4
3.2.7	Stop Transaction	4
3.3	From Server To Client	4
3.3.1	Cancel Reservation	4
3.3.2	Remote Start Transaction	4
3.3.3	Remote Stop Transaction	4
3.3.4	Reserve Now	4
4	Backend	4
4.1	Description	4
4.2	Database	5
4.3	Authentication	6
4.4	Station	6
4.5	API List	7
4.5.1	Auth	7
4.5.2	Station	7
4.5.3	User	8
5	Dashboard	8
5.1	Description	8
5.2	Home	8
5.2.1	States of charging stations	9
5.3	Adding and editing a charging station	9
5.4	Current year's report	10
5.5	Admin user management	10
5.6	Access Policy	11
6	Mobile Application	12
6.1	Description	12
6.2	Register and authenticate	12
6.3	Map of charging stations	12
6.4	Reservation of a charging station and initiation of charging	13
6.5	Start and Stop charging scanning QRCode	13

6.6	Status of last reservation or recharge	14
6.7	Available credit and profile update	14
7	Further Developments	15

1 Introduction

The project consists in the development of a smart system for managing and monitoring Charging Stations (CS) for Electric Vehicles (EV).

The EV Driver uses an APP to be authenticated and to start or stop charging and to reserve a station.

The APP communicates with the CSMS via the HTTP protocol and the CS communicates with the CSMS via the OCPP 1.6 protocol.

The Driver can also start charging by scanning a QR code on the CS.

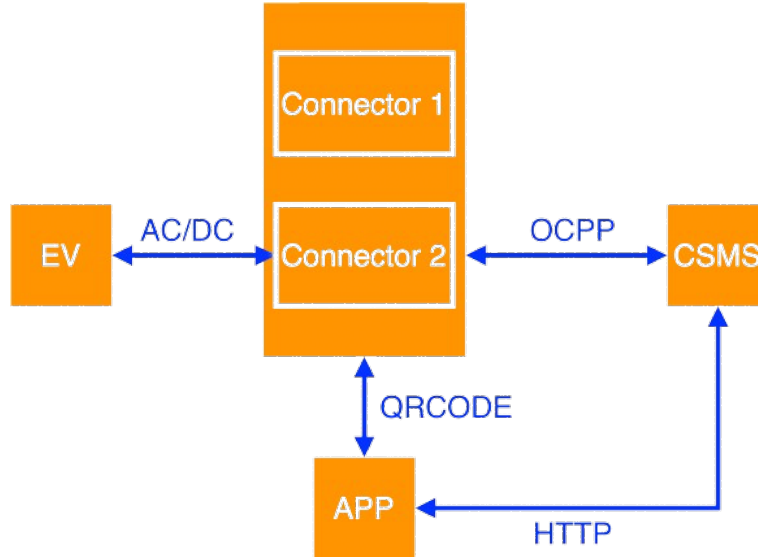


Figure 1: System

2 OCPP

The Open Charge Point Protocol (OCPP) 1.6 is a widely adopted standard in the electric vehicle (EV) charging industry, facilitating communication between charging stations and central management systems (CMS). This protocol defines a set of messages and data structures that enable functionalities such as start and stop charging sessions, retrieve charging status, and manage authentication processes. OCPP 1.6 supports both local and remote interactions, allowing for seamless integration of charging infrastructure with networked management systems. With its standardized approach, OCPP 1.6 promotes interoperability among different charging station manufacturers and CMS providers, fostering a more efficient and scalable EV charging ecosystem.

3 Client

3.1 Description

The client was developed in NodeJS and uses the OCPP-RPC module to manage OCPP 1.6 requests.

Represents a single charging station, manages the reservation, start and end of a charge and its monitoring.

Currently the user cannot interact with it directly since all operations start from the mobile application and therefore go through the server first.

3.2 From Client To Server

3.2.1 Authorize

Before the owner of an electric vehicle can start or stop charging, the Charge Point has to authorize the operation. The Charge Point SHALL only supply energy after authorization.

3.2.2 Boot Notification

After start-up, a Charge Point SHALL send a request to the Central System with information about its configuration (e.g. version, vendor, etc.). The Central System SHALL respond to indicate whether it will accept the Charge Point.

3.2.3 Heartbeat

To let the Central System know that a Charge Point is still connected, a Charge Point sends a heartbeat after a configurable time interval.

3.2.4 Meter Values

A Charge Point MAY sample the energy meter or other sensor/transducer hardware to provide extra information about its meter values. It is up to the Charge Point to decide when it will send meter values.

3.2.5 Start Transaction

This operation initializes a charging session between an electric vehicle and a charging station. It typically involves the authentication of the user, the allocation of charging resources, and the beginning of the charging process.

3.2.6 Status Notification

A Charge Point sends a notification to the Central System to inform the Central System about a status change or an error within the Charge Point.

3.2.7 Stop Transaction

Once a charging session is complete or needs to be terminated prematurely, the "Stop Transaction" operation is used. It involves concluding the charging process, finalizing the transaction details, and making the charging spot available for other users.

3.3 From Server To Client

3.3.1 Cancel Reservation

If a user decides not to proceed with a previously made reservation, the Central System sends a cancellation request to the Charging Station. The Charging Station then releases the reserved resources, making them available for other users or transactions.

3.3.2 Remote Start Transaction

This operation allows the Central System to remotely initiate a charging session without physical interaction with the Charging Station. It involves sending a request from the Central System to the Charging Station to begin the charging process, providing flexibility and convenience for users and operators.

3.3.3 Remote Stop Transaction

The Remote Stop Transaction operation enables the Central System to remotely terminate an ongoing charging session. Upon receiving the request, the Charging Station stops power delivery, finalizes transaction details, and communicates the end of the session to the Central System.

3.3.4 Reserve Now

The Central System sends a Reserve request to the Charge Point and both reserve the specified charging resources, ensuring availability at the scheduled time.

4 Backend

4.1 Description

The backend was developed in NodeJS and MySQL using the express framework to handle HTTP requests. The OCPP-RPC module was used to manage OCPP requests..

Starting from the `src` folder, we can find the following project structure:

- **middleware:** functions that are called before processing the HTTP request to verify that authentications are met

- **models:** classes that represent the entities in the db by providing the ability to perform read and write operations
- **routes:** classes that manage the routes of the different APIs
- **services:** classes that implement the business logic needed in the various API calls
- **views:** HTML pages returned by the server. They are used to implement the password reset.

4.2 Database

To implement the database, it was decided to use MySQL.

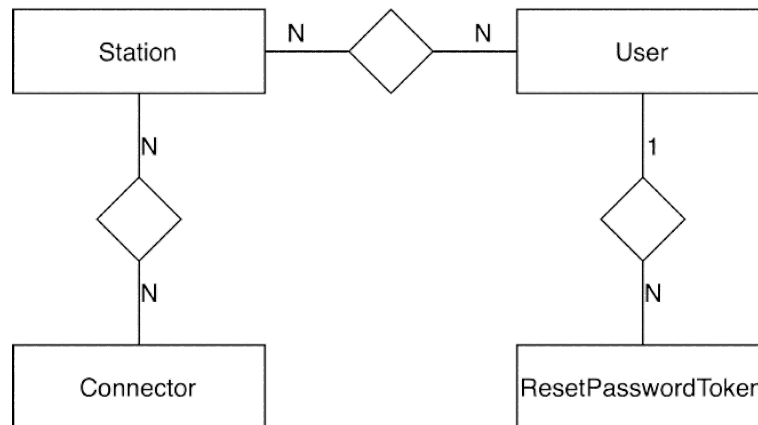


Figure 2: ER schema of the database

The database consists of the following entities:

- **Station:** List of stations in the system. Each station is characterized by the following fields:
 - **name:** Station name
 - **lat e lon:** Latitude and longitude of the station
 - **price:** station price per kWh expressed in cents
 - **dismissed:** flag indicating whether the station has been removed
 - **last_heartbeat:** timestamp of the last heartbeat received
 - **notes:** notes visible only by admins
 - **description:** description visible to users
- **Connector:** List of connectors in the system. Each connector is characterized by the following fields:
 - **name:** Name of connector
 - **power:** Connector power
- **User:** List of users registered in the system. Each user is characterized by the following fields:
 - **name e surname:** User's first and last name
 - **email:** User email
 - **password:** hash of the user's password
 - **token_reset_time:** timestamp Updated every time the user changes the password. Allows previously generated authentication tokens to be invalidated
 - **balance:** user credit expressed in cents
 - **is_admin:** flag indicating whether the user is an admin
- **ResetPasswordToken:** List of tokens generated to reset the password. Each token is characterized by the following fields:

- **email:** User email
- **token:** Token generated
- **generated_time:** timestamp indicating when the token was generated
- **used:** flag indicating whether the token has already been used

We also find the following many-to-many relationships that have been represented through other entities:

- **StationConnector:** Allows you to assign a connector to a station
 - **station_id:** station id
 - **connector_id:** connector id
- **StationUsage:** Allows a user to represent a station usage. Each usage is characterized by the following fields:
 - **user_id:** user id
 - **station_id:** station id
 - **start_time:** timestamp indicating when the recharge started
 - **end_time:** timestamp indicating when the recharge ended
 - **reservation_time:** timestamp indicating when the station reservation was made
 - **kw:** kWh used in charging
 - **price:** Price of electricity per kWh when charging was done
 - **deleted:** flag indicating whether the reservation has been cancelled

4.3 Authentication

Authentication is implemented through JWT.

There are 4 different types of authentication levels:

- **Everyone:** are APIs that can be called by anyone and do not require any kind of token
- **ResetPassword:** are APIs that can only be called by users who have requested a password reset and require that a token of type password reset be passed
- **User:** are APIs that can only be called by users who have logged in and require that a token representing a user be passed
- **Admin:** are APIs that can only be called by an admin user and therefore require the `is_admin` flag present in the token and set to true.

4.4 Station

Within the system, the status of the stations is determined based on the value of `last_heartbeat` and the most recent record inside `StationUsage` (with the `deleted` flag = false).

A station therefore can have six different states:

- **Free:** it is a working station that has not been booked and is not being used
- **Reserved:** this is a working station that has been booked
- **Used:** This is a working station that is currently used
- **Dismissed:** this is a station that has been removed from the system and is no longer visible to users. It is still maintained in the system to preserve the historical
- **Broken:** it is a station whose last 3 heartbeats were not received and is reported to users as broken
- **Undefined:** this is a station that has never communicated with CSMS having never sent a heartbeat. This is a station that has probably yet to go into operation and is therefore not visible to users

4.5 API List

4.5.1 Auth

- **/auth/register:** API that can be called without authentication via POST method. This API registers a new user in the DB, takes as input a json containing the user's info, and returns the access token or an error message
- **/auth/login:** API that can be called without authentication via POST method. This api allows logging into the application/dashboard, takes as input a json containing the user's credentials (email and password) and returns the access token or an error message
- **/auth/validateToken:** API that can only be called by authenticated users via GET method. This API allows the access token to be validated by returning the message "Valid token"
- **/auth/resetPasswordToken:** API that can be called without authentication via POST method. This API allows you to request a password reset, takes as input the user's email address to which to send the password reset, and returns a confirmation or error message
- **/auth/resetPasswordPage:** API that can only be called by authenticating by sending a password reset token via GET method. This API returns the HTML page containing the form to enter the new password
- **/auth/resetPassword:** API that can only be called by authenticating by sending a password reset token via POST method. This API takes the new password and email as input and returns a confirmation or error message.

4.5.2 Station

- **/station:** API that can be called without authentication via GET method. This api allows receiving the list of all stations in the system
- **/station/report:** API that can only be called by users authenticated as Admin via GET method. This API allows receiving a report containing the current year's consumption and earnings by returning it as JSON
- **/station/{id}:** API that can be called without authentication via GET method. This api allows you to receive the information of a single station, takes as input the station id and returns a JSON containing the station information
- **/station/{id}/last_charge:** API that can be called without authentication via GET method. This api allows receiving the information of the last recharge made at a station, takes as input the id of the station and returns a json containing the recharge information
- **/station/{id}/last_reservation:** API that can be called without authentication via GET method. This api allows receiving the information of the last reservation made at a station, takes as input the station id and returns a json containing the reservation information
- **/station/{id}/reserve:** API that can only be called by authenticated users via POST method. This API is used to reserve a free station, takes as input the id of the station to be reserved and returns a JSON containing the reservation information or an error message. Within this API, a request of type ReserveNow is also sent to the station
- **/station/{id}/cancel_reservation:** API that can only be called by authenticated users via POST method. This API is used to cancel a reservation made by the same user who authenticated, takes as input the id of the station where to cancel the reservation and returns a JSON containing the reservation information or an error message. Within this API, a request of type CancelReservation is also sent to the station
- **/station/{id}/start_charging:** API that can only be called by authenticated users via POST method. This API is used to start a recharge at a free or reserved station by the same user who has authenticated, it takes as input the id of the station where to start the recharge and returns a JSON containing the recharge information or an error message. Within this API, a request of type RemoteStartTransaction is also sent to the station
- **/station/{id}/stop_charging:** API that can only be called by authenticated users via POST method. This API is used to stop a recharge at a station used by the same user who authenticated, takes as input the id of the station where to end the recharge and returns a JSON containing the recharge information or an error message. Within this API, a request of type RemoteStopTransaction is also sent to the station

- **/station:** API that can be called only by users authenticated as Admin via POST method. This API is used to create a new station, requests as input a JSON containing the information of the station to be created, and returns a JSON containing the information of the newly created station
- **/station:** API that can be called only by users authenticated as Admin via PATCH method. This API is used to update an existing station, it requests as input a JSON containing the information of the station to be updated and returns a JSON containing the information of the newly updated station
- **/connector:** API that can be called without authentication via GET method. This API is used to get the information of all connectors, returning a JSON containing the information of all connectors.

4.5.3 User

- **/user/all:** API that can only be called by users authenticated as Admin via GET method. This API allows obtaining the list of all users in the system by returning them as JSON objects
- **/user:** API that can only be called by authenticated users via GET method. This API returns the authenticated user's information via JSON
- **/user/last_usage:** API that can only be called by authenticated users via GET method. This api returns the information of the last use of a station by the authenticated user via JSON
- **/user:** API that can only be called by authenticated users via PATCH method. This api allows the authenticated user's information to be updated, requests as input a JSON containing the user's information to be updated, and returns a JSON containing the newly updated user's information
- **/user/{id}/set_user_admin:** API that can only be called by users authenticated as admin via PATCH method. This API allows setting or removing another user as admin, requires as input the id of the user to be put as admin and a flag indicating whether the user should be admin and returns a JSON with the user's newly updated information.

5 Dashboard

5.1 Description

The dashboard was implemented using React for the frontend and sending HTTP requests to the backend seen above.

The dashboard is accessible only by **admin** users and provides the following functionality:

- Add and edit charging stations
- See the status of the charging stations
- See a report with data for the past year
- Manage the list of **admin** users

5.2 Home

On the homepage of the dashboard it is possible to see a table with all the chargin stations with the possibility of editing them by going to click on the "Edit" button in the row that corresponds to the CS to be edited. Also shown is a map with the location and color of the CSs according to their status.

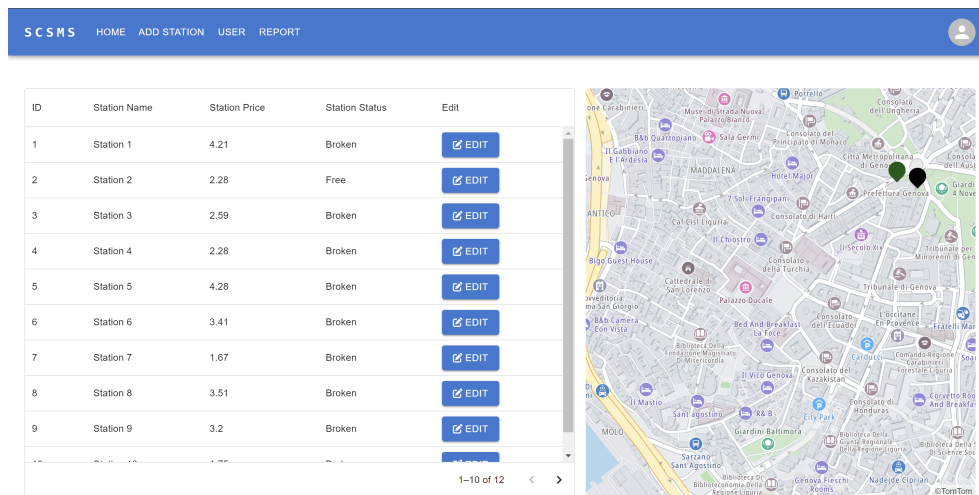


Figure 3: Home

5.2.1 States of charging stations

According to the status of the various CS, colored markers are shown on the map. Each color corresponds to a status:

- **Green:** The station is free;
- **Orange:** The station has been booked;
- **Red:** The station is occupied;
- **Black:**
 - The station is in the dismissed state, i.e., disabled;
 - The station has not sent out any heartbeats in the last period;
 - The station is in an undefined state.

5.3 Adding and editing a charging station

Through the dashboard an admin is able to add or edit a CS. The form for adding/editing a CS is the same with the difference that if the CS already exists the fields will start out enhanced with the Station details.

SC5MS HOME ADD STATION USER REPORT

Add Charging Station

Name

Insert station name

Price (€/kWh)

Insert price

Connectors

Longitude

Insert longitude

Latitude

Insert latitude

Last heartbeat

--

☐ Dismissed

Note

Insert note

Description

Insert description

SAVE

CANCEL

Figure 4: Add Station

SCSMS
HOME
ADD STATION
USER
REPORT

Edit Charging Station 2

Name
Station 2

Price (€/kWh)
2.28

Connectors
Type 2

Longitude
8.93768364

Latitude
44.40973067

Last heartbeat
28/2/2024, 15:36:45

☐ Dismissed

Note
This is a note

Description
description 2

SAVE
CANCEL

Figure 5: Edit Station

5.4 Current year's report

Through the dashboard, it is possible to get a report on the current year's earnings and power consumed.

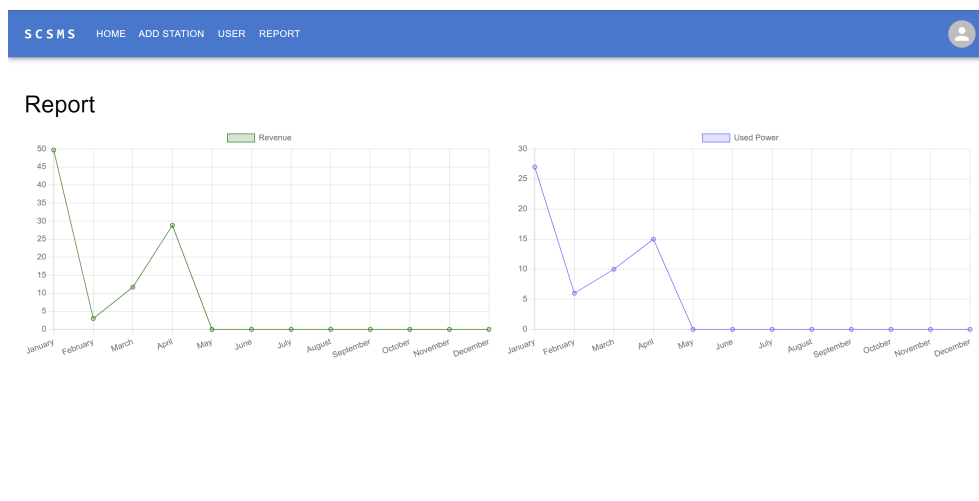


Figure 6: Report

5.5 Admin user management

It is possible to give the admin role through the dedicated page in the dashboard. An admin can appoint another user admin by going to the User section and changing the flag in the corresponding table row.

SCSMS				
HOME ADD STATION USER REPORT				
ID	User Name	user Surname	User Email	Is Admin?
1	matteo	I	mi@g.com	False
2	Matteo	La Licata	mi2@g.com	True
5	User	Test	user@gmail.com	False
1-3 of 3 < >				

Figure 7: User table

5.6 Access Policy

Only admins can access the dashboard, in case any user tries to enter the login would be rejected.

SCSMS

Login

Email *

Password *

LOGIN

Figure 8: Login

localhost:3000/login

Google Idee blog centurini Indirizzo Email Tem... AulaWeb 2022

localhost:3000 dice
You are not authorized to access this page!
OK

SCSMS

Login

Email *

mi2@g.com

Password *

.....

LOGIN

org - Scacchi... Homepage - Chess... Training di scacchi o...

Figure 9: Login Failed

6 Mobile Application

6.1 Description

The application was implemented using React Native in order to have a single application available right away for both iOS and Android.

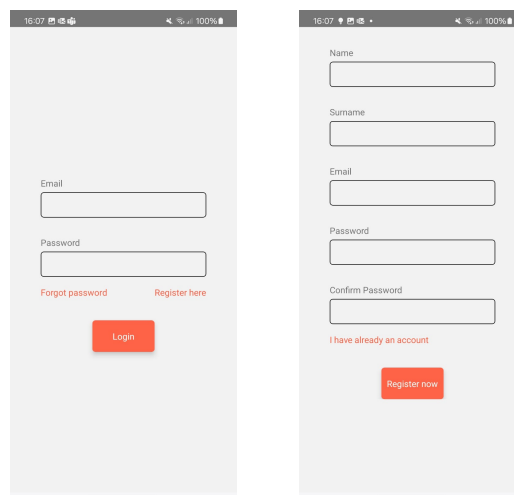
The application sends HTTP requests to the backend seen above.

The application allows all users to:

- Register and authenticate
- See a map of nearby charging stations (with status)
- Reserve a charging station and initiate a recharge
 - Charging can be initiated by scanning QR codes or by searching for the charging station within the app
- See the status of the last reservation or recharge
- See available credit and do profile update

6.2 Register and authenticate

A user must have an account in order to use the app. You can register directly in the app, in case you already have an account you can log in.



The image displays two side-by-side mobile application screens. The left screen is the login page, featuring input fields for 'Email' and 'Password', a red 'Login' button at the bottom, and links for 'Forgot password' and 'Register here' above the button. The right screen is the registration page, featuring input fields for 'Name', 'Surname', 'Email', 'Password', and 'Confirm Password', a red 'Register now' button at the bottom, and a link 'I have already an account' above the button. Both screens have a light gray background and a white status bar at the top showing the time as 16:07 and 100% battery.

Figure 10: Login and Registration

6.3 Map of charging stations

The home of the application shows a map of the area surrounding the user's location showing through markers all the charging stations present, regardless of whether their status is: free, occupied or booked.

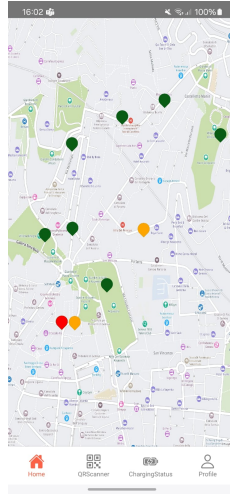


Figure 11: Home mobile

6.4 Reservation of a charging station and initiation of charging

Clicking on a marker representing a CS shows a bottom dialog with the CS info and the possibility to start/book a recharge. In case the CS was activated by a user previously this user can re-click on the marker and stop the recharge. The same applies to booking and canceling booking a CS.

If a CS is booked, clicking on the marker you can see when the booking will expire.

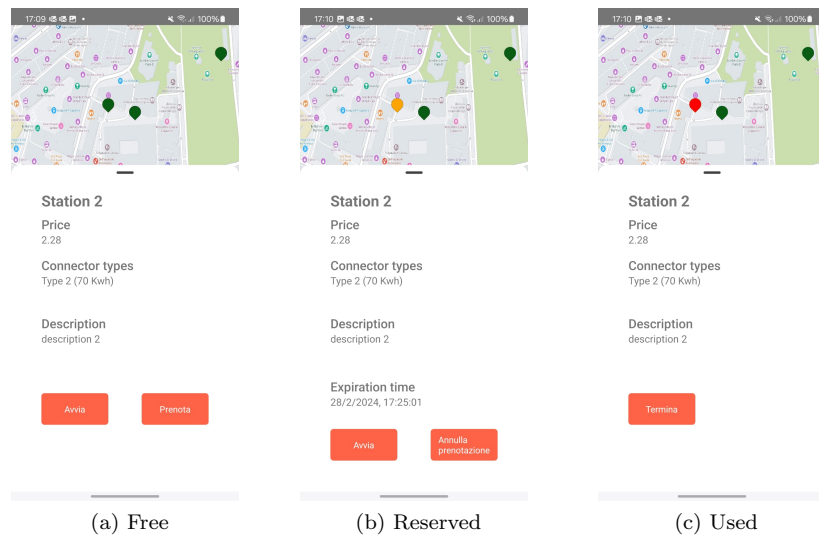


Figure 12: Info e Stop charging modal

6.5 Start and Stop charging scanning QRCode

There is a QR code reader in the app. When a user wants to start or stop a recharge he/she will simply scan the QR Code located on the CS concerned. If the CS has been reserved by another user or is already occupied scanning the QRCode will have no effect. QRCode reading has been implemented using the library react-native-qrcode-scanner.

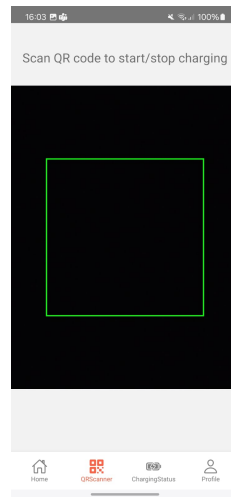
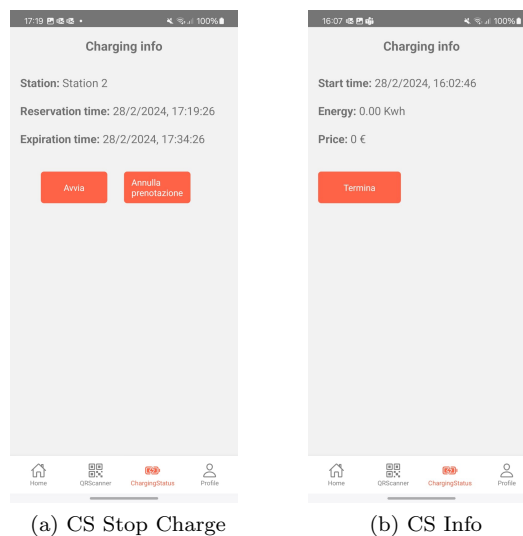


Figure 13: QRCode scanner

6.6 Status of last reservation or recharge

A user can monitor his or her last reservation or current charge in the "Charging Status" section, in which several info are shown. If the last operation is a charging it shows when it was started, the current cost, and the KWh consumed. While if the last operation is a reservation it shows the name of the reserved CS, when it was made and when it will expire.



(a) CS Stop Charge

(b) CS Info

Figure 14: Info e Stop charging modal

6.7 Available credit and profile update

Through the app, in the "User" section, a user is able to change their profile info such as: first name, last name, email, and password. By clicking on the Save button, the changes will be made. In this section, a user can also check his or her available credit.

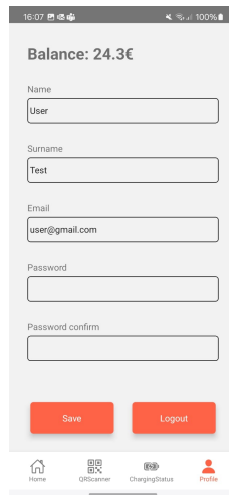


Figure 15: Caption

7 Further Developments

Possible future developments for the project could be:

- Implement an internal payment system within the app in order to increase available credit
 - Currently when an account is created it starts with 100 and you cannot reload the account
- Implement the initiation of a charge directly from the charging station via NFC or Credit Card using the "StartTransaction" call provided by the OCPP protocol
 - Currently, it is only possible to initiate a recharge using the mobile app