# Software Engineering 2: PowerEnjoy Design Document

Andrea Pace, Lorenzo Petrangeli, Tommaso Paulon

December 13, 2016

Version 1.0

## 1 Introduction

### 1.1 Purpose

The purpose of this document is to show:

- the high level architecture
- the main design patterns
- the main components,
- their interfaces and the way they are connected

### 1.2 Scope

This project has two targets of people: the user and the operator
The user will be able to sign up and make reservations, while the system will check if the data provided by the users are correct and will compute the cost of the ride.
The operator will receive notifications about problems with cars

### 1.3 Definitions, acronyms, abbreviations

### 1.4 Reference documents

- Rasd
- Assignments AA 2016-2017

# 2 Architectural design

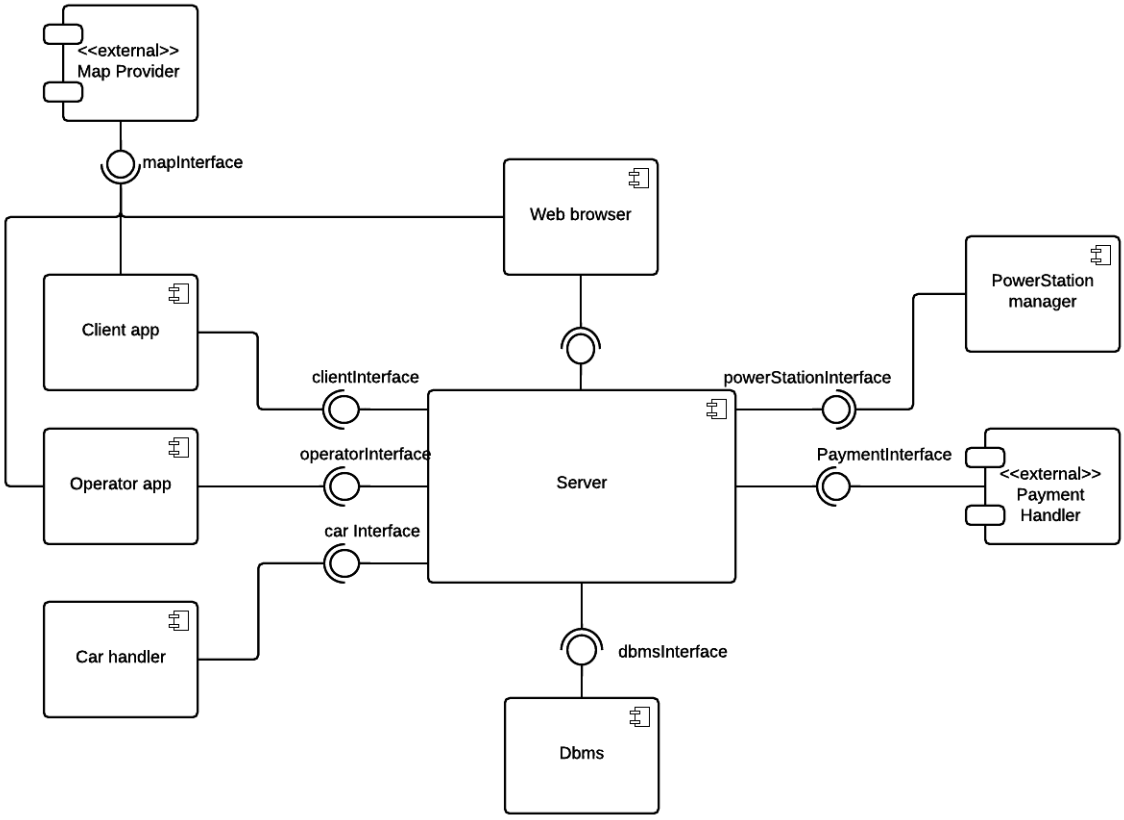
## 2.1 Overview

The system has a three tier architecture, we have:

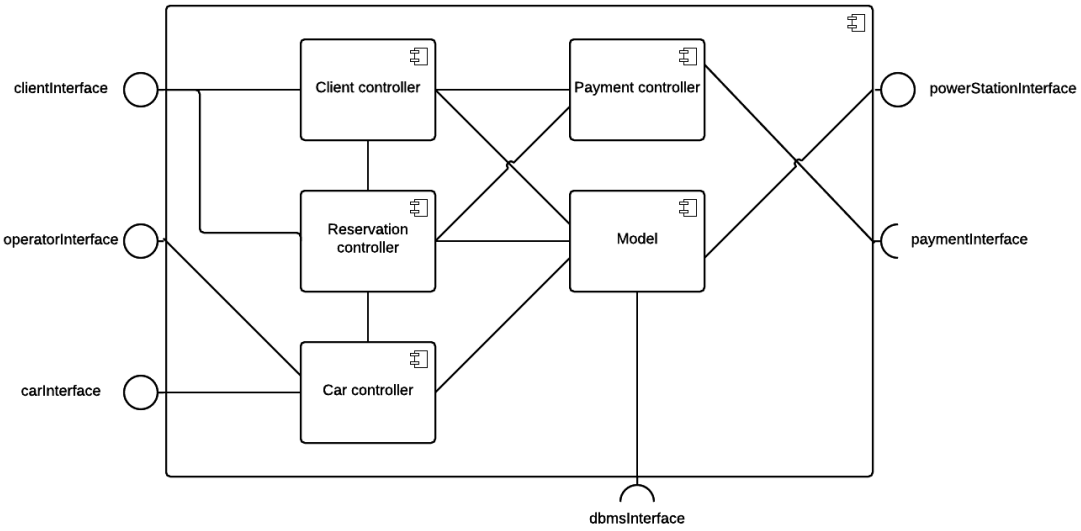
- The client and the operator devices, plus the web server
- an application server which handles the business logic
- a database server

## 2.2 High level components and their interaction

The central server handles the communication between the different components.
The client can send requests from the app or from the web server and receives notifications. The requests that can be sent from the web server are a subset of the requests available via app.
The car handler sends notifications about his status(battery level, current charge, position) and receives request for internal state change from the server (reservation and unlocking).
The power stations only send informations about the number of free plugs.
The payment handler receives payment data and returns a positive answer if the payment is accepted through an external interface.
The map provider simply receives positions and returns a map through an external interface to the client apps and to the browser


<<external>>
Map Provider
mapInterface
Web browser
PowerStation
manager
Client app
clientInterface
powerStationInterface
Operator app
operatorInterface
Server
PaymentInterface
<<external>>
Payment
Handler
car Interface
Car handler
dbmsInterface
Dbms

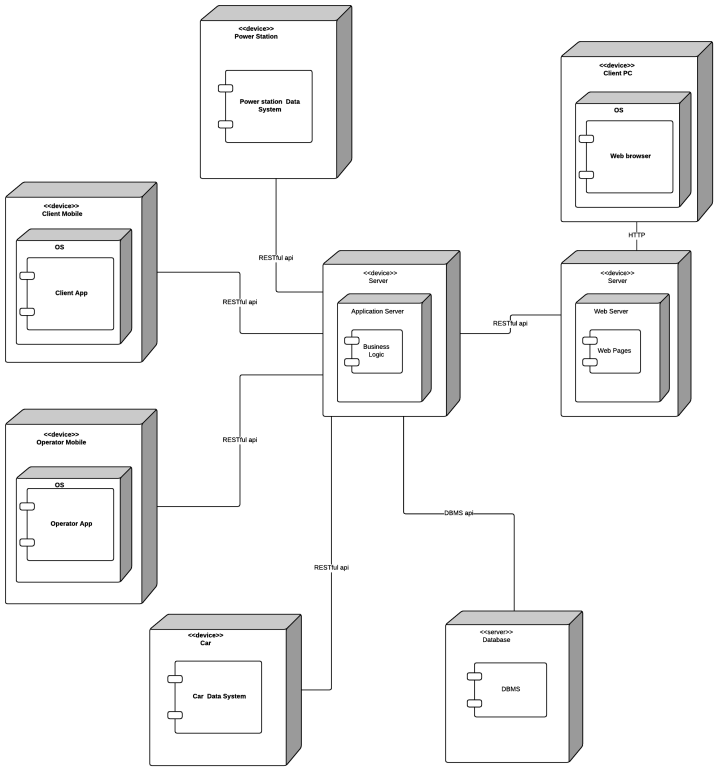
## 2.3 Component view




- Client controller: handles login, registration and in general all the client data
- Reservation controller: handles the reservation, the unlock requests and submits the final charge to the payment controller. It includes the car controller in order to receive data concerning position, number of passengers and every information needed to compute the final charge
- Car controller: manages all the data from the car sensors. It is also in charge of notifying the operator for all the problems stated in the RASD
- Payment controller: manages the real communication with the payment handler. It also interacts with the Client controller when the client decides to pay off his debts.

The power stations communicate directly with the model, since the only information they share is the number of free plugs.
Every component interacts with the model, which is the only link to the DBMS.

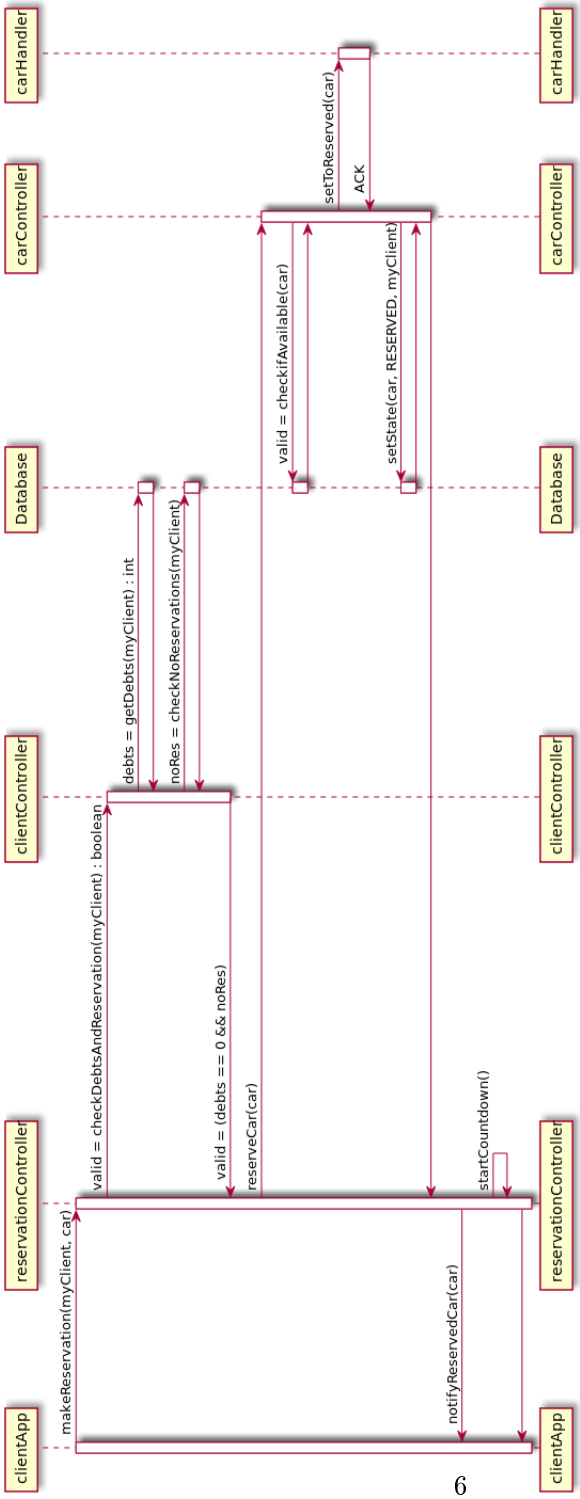
## 2.4 Deploying view




## 2.5 Runtime view

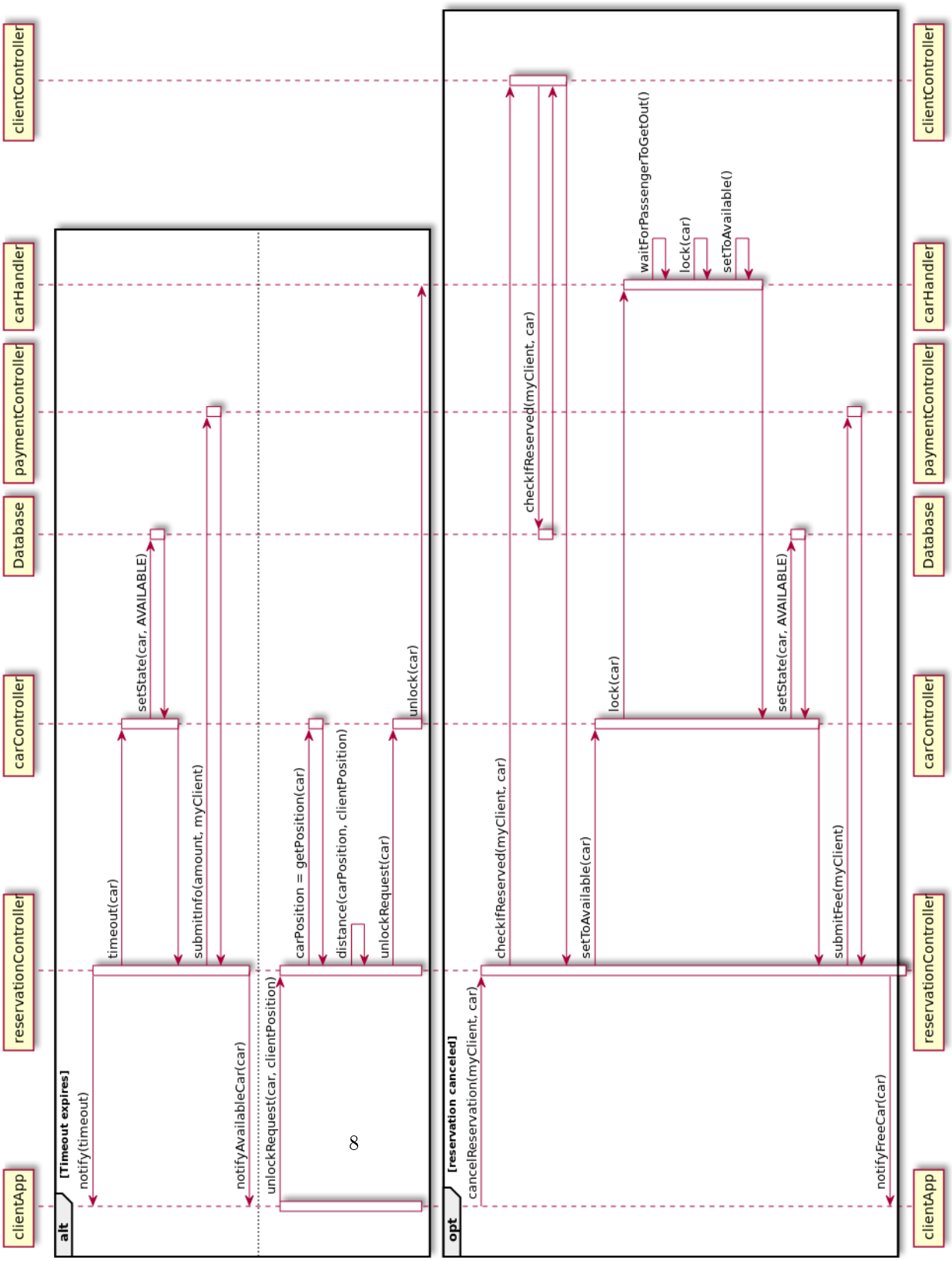
The following sequence diagrams show the interactions between components from the reservation of a car to the end of the ride. The first three diagrams are consecutive. The interaction with the external payment handler is shown in a separate diagram since it's always the same process. Every interation with the database is actually handled by the Model component, but we always skip this component in the diagrams.

### 2.5.1 Reservation



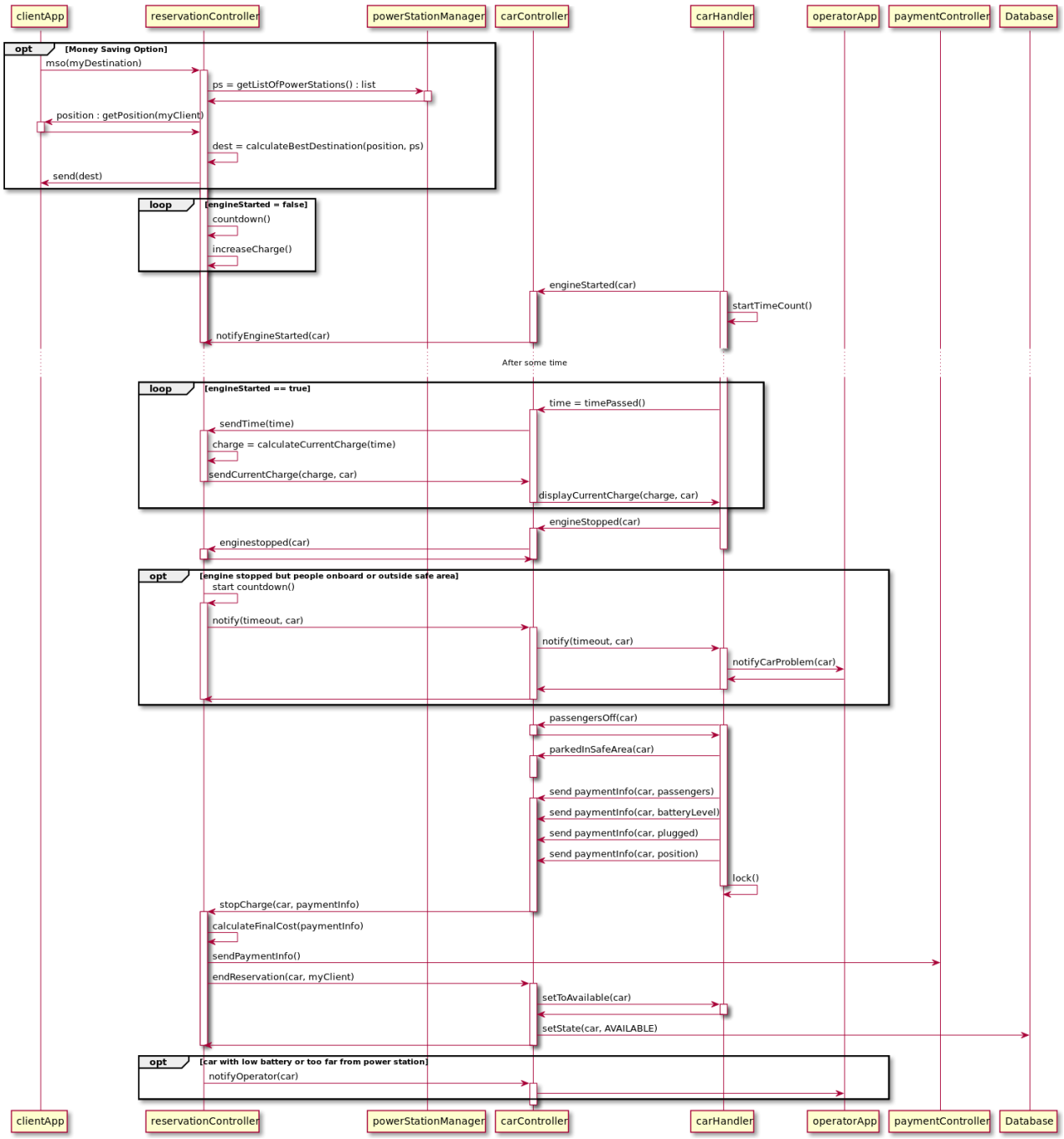
This diagram represents the reservation process. Every call is synchronous since no process can proceed without the previous operations.
This particolar view shows a situation where all the conditions to make a reservation are met. If any control fails the process simply stops. If the reservation is confirmed, the reservation controller broadcasts this information, so the user app is always up to date

### 2.5.2 Car unlocking



This diagram shows several situations. If the timeout expires before the unlock request the car becomes available again, and the client pays a fee and the availability of the car is broadcasted to all the clients
If the unlock request arrives before the countdown expires, the system checks if the client's position is close enough. In this view we show the system behaviour when this condition is met. A reservation cancel can occur in this part of the process. In this case the system checks if there is a real reservation and if this occurs the reservation is canceled. According to the requirements we should check if the engine is already ignited, but we simply don't accept a reservation cancel after that moment. As usual the availability of the car is broadcasted.

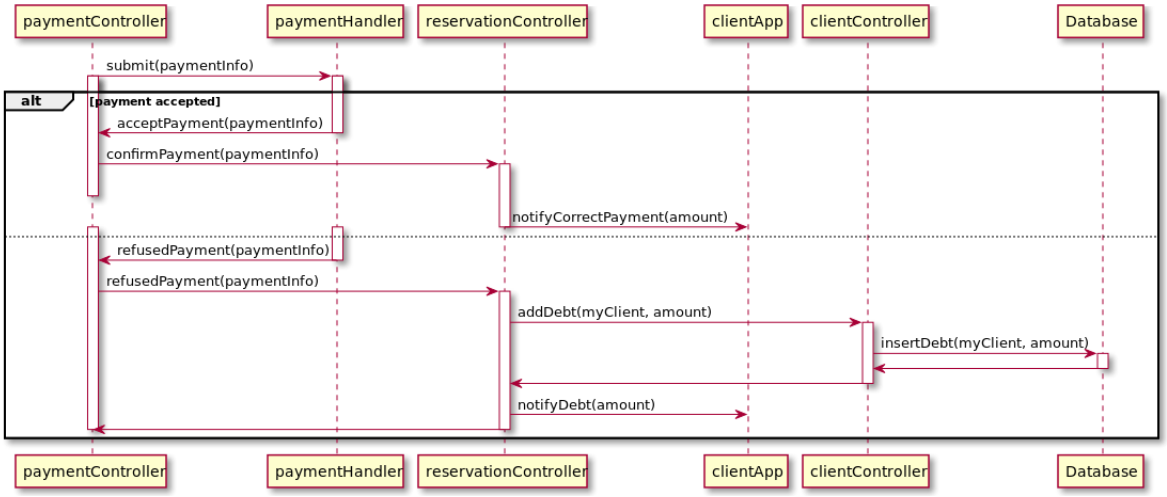
### 2.5.3 End of the Ride




The first optional part concerns the money saving option. The list returned

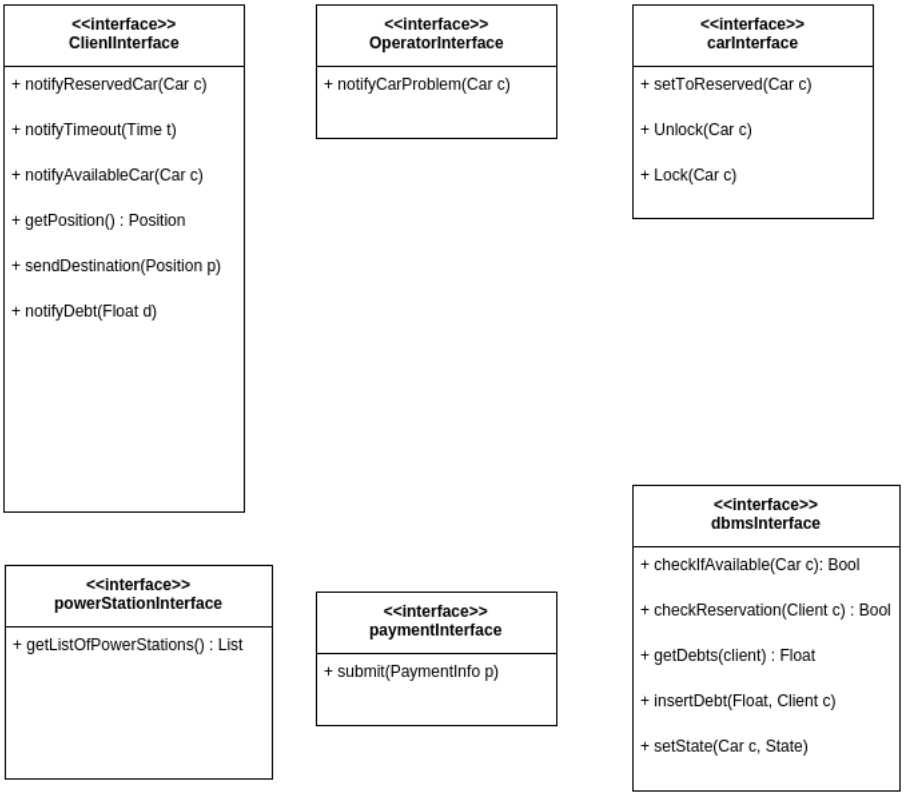
by the powerStationManager contains the number of free plugs of each power stations.
The system is notified of this option after the unlocking of the car.
Then the system periodically increases the client charge if he/she doesn't ignite the engine.
During the ride the system periodically sends the current charge to the car, in order to display the amount to the user.
Then we have different behaviours if the client parks the car in a safe area or not. If the client leaves the car outside the safe area or doesn't leave the car the current charge keeps increasing. In those cases an operator is notified.
In the final part an operator is notified of a car left too far from the nearest power station or left with low battery

### 2.5.4 Payment process




This diagram shows the two situation when a payment is submitted to the payment handler.

## 2.6 Component Interfaces




These interfaces provide access to the client apps and to external components.

## 2.7 Selected architectural styles and patterns

### 2.7.1 Protocols and architecture

The application is divided into three tiers: Database, application logic and thin client.
We have a relational database who communicates with the application logic tier through the dbms APIs.
The application logic server interacts with the web server and the clients with RESTful api. This approach has several advantages: being a stateless protocol means that no session of the client is saved on the server, each request is independent from previous requests and contains all the information to be interpreted correctly.
The application server is made of stateless components: this ensures that multiple instances of the components could be created and they could serve any request in the same way in order to guarantee scalability.

The thin client layer handles the GUI, receives notifications and can send requests.
We also use HTTP between the web server and the clients.
The client app comunicates with the map provider through its API.

#### 2.7.2 Design Patterns

The mvc pattern concerns the whole system: the model contains the internal representation of the world, the controllers are the handlers of the business logic and the views are represented by the clients applications. In this way any of the three conponents can be changed regardless of the others.
Then also a pub/sub pattern is used: the reservation controller notifies all the currently logged users about reserved and released cars. The user subscribes to this topic when he logs in.

# 3 Algorithm design

## 3.1 Calculate cost:

the algorithm will start considerating the amount of money calculated by the carHandler, which depends only on the duration of the ride. Then the algorithm will apply penalties, if any :

- if a car is left at more than 3 KM from the nearest power station or with more than 80% of the battery empty,the algorithm charges 30% more

And then benefits will be considered:

- if the car sensor detected more than two passenger during the travel the algorithm applies a 10% discount
- if the car is left with more than 50% of battery, the algorithm applies a 20% discount
- if the car is plugged into a power grid and it's left in a safe area, the algorithm applies a 30% discount

where a x% discount(charge) on a value y means: y = y - y*x/100 (y = y + y*x/100)

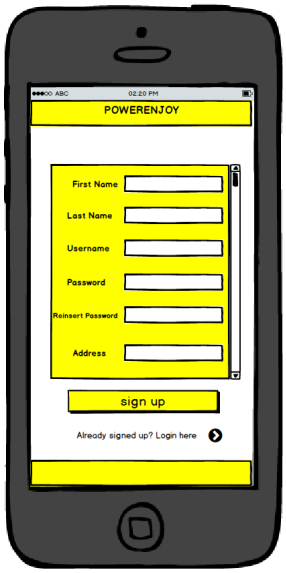
## 3.2 Power station for money saving option:

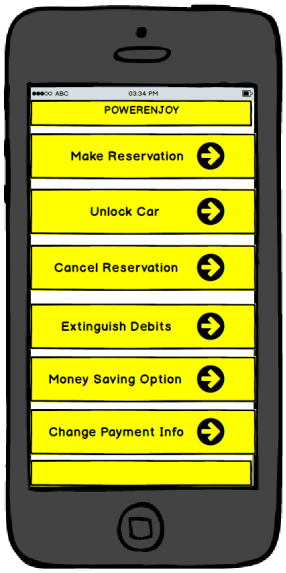
The chosen power station will be the one closest to the requested destination with a percentage of free plugs greater or equal to the average of the percentage of the free plugs of all the power stations.

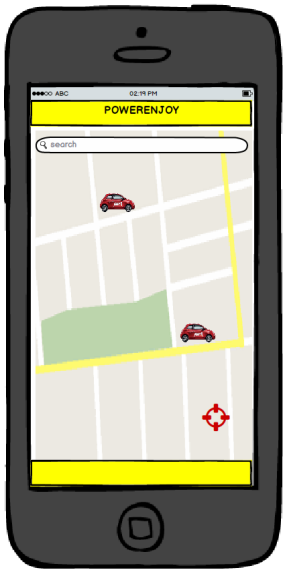
# 4 User Interface design

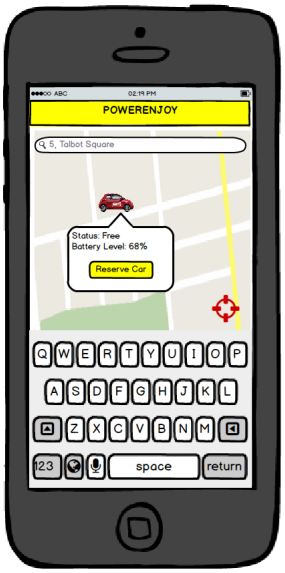
## 4.1 Mockups

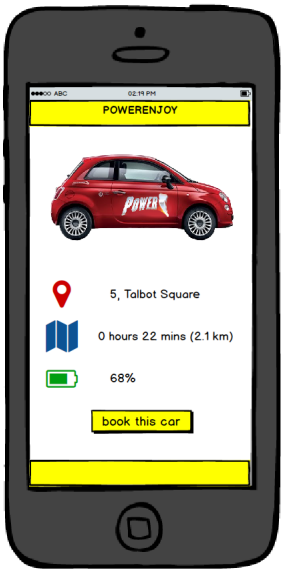
### 4.1.1 Client App



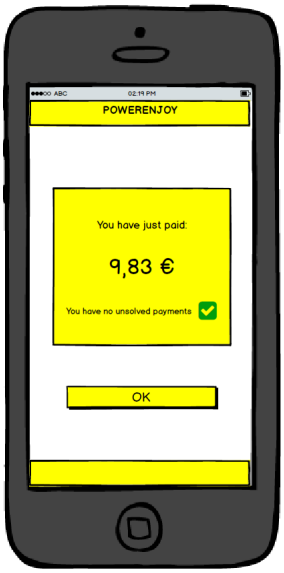


ABC
03:34 PM
POWERENJOY
Make Reservation
Unlock Car
Cancel Reservation
Extinguish Debits
Money Saving Option
Change Payment Info

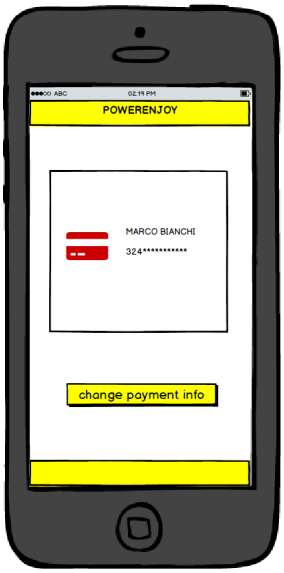

ABC
02:19 PM
POWERENJOY
search


POWERENJOY
5, Talbot Square
Status: Free
Battery Level: 68%
Reserve Car
space
return


POWERENJOY
5, Talbot Square
0 hours 22 mins (2.1 km)
68%
book this car

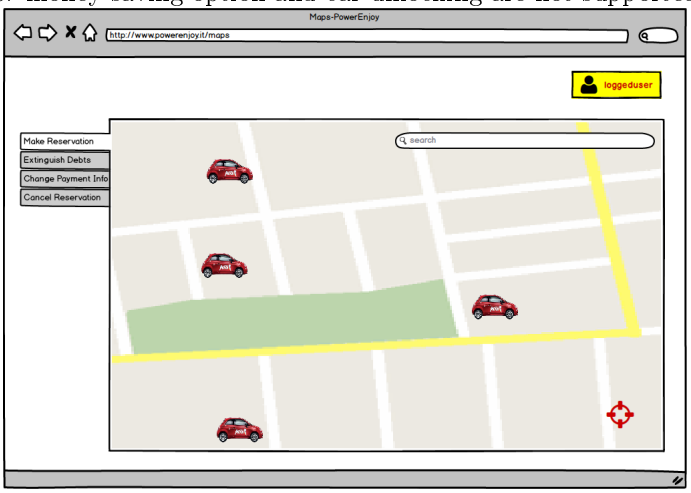

POWERENJOY
5, Talbot Square
43:33
unlock car

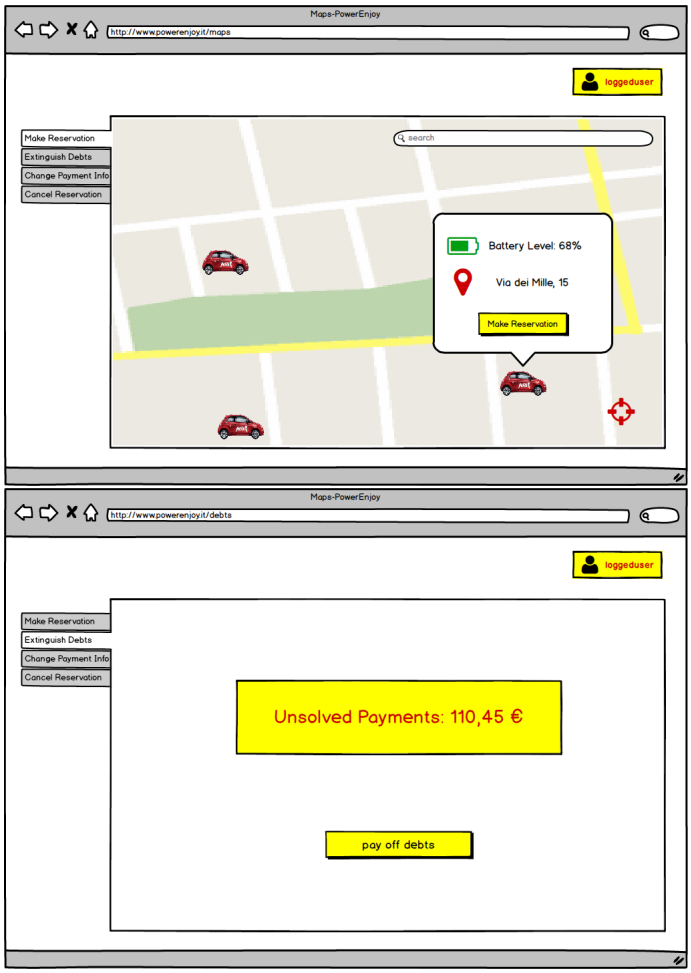

POWERENJOY
You have just paid:
9,83 €
You have no unsolved payments
OK



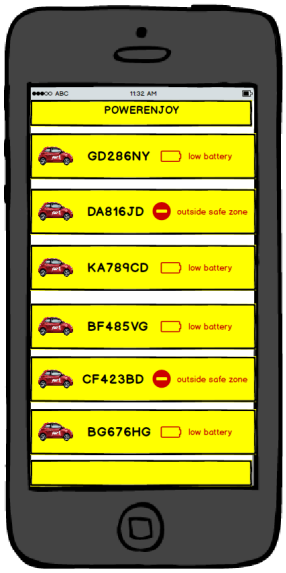

### 4.1.2 Client Web

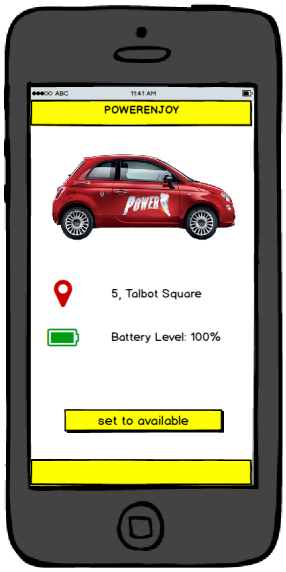
The web interface offers a subset of the functionality available with the mobile app: money saving option and car unlocking are not supported via web app.

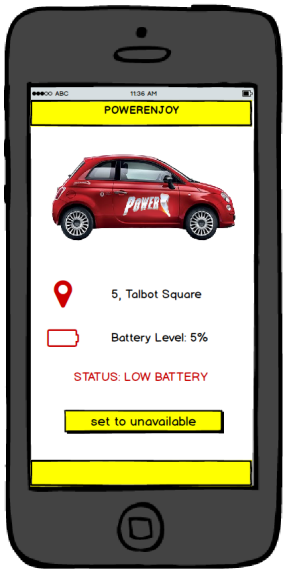



Maps-PowerEnjoy
http://www.powerenjoy.it/maps
loggeduser
Make Reservation
Extinguish Debts
Change Payment Info
Cancel Reservation
search
Battery Level: 68%
Via dei Mille, 15
Make Reservation
Maps-PowerEnjoy
http://www.powerenjoy.it/debts
loggeduser
Make Reservation
Extinguish Debts
Change Payment Info
Cancel Reservation
Unsolved Payments: 110,45 €
pay off debts

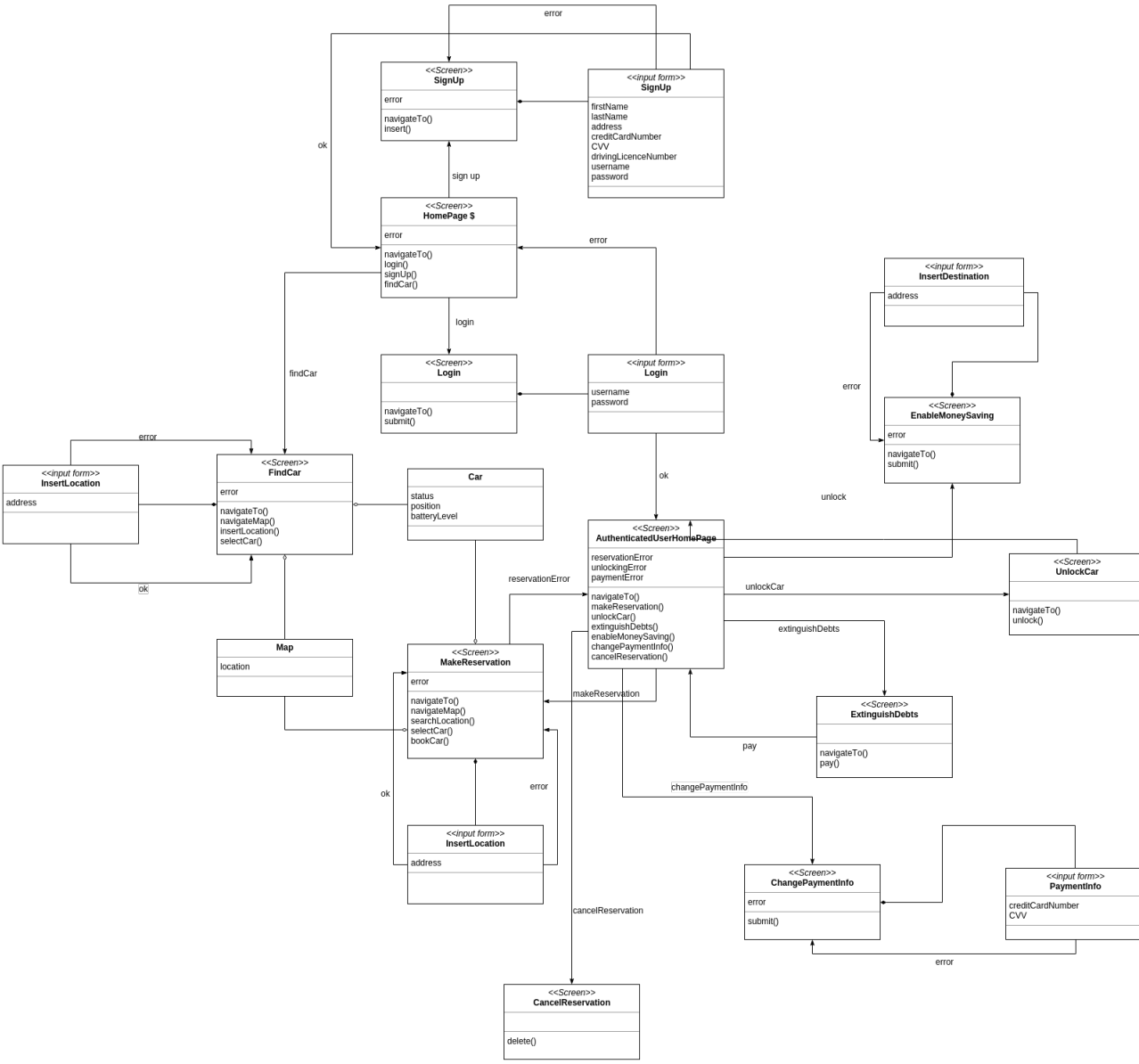
### 4.1.3 Operator App

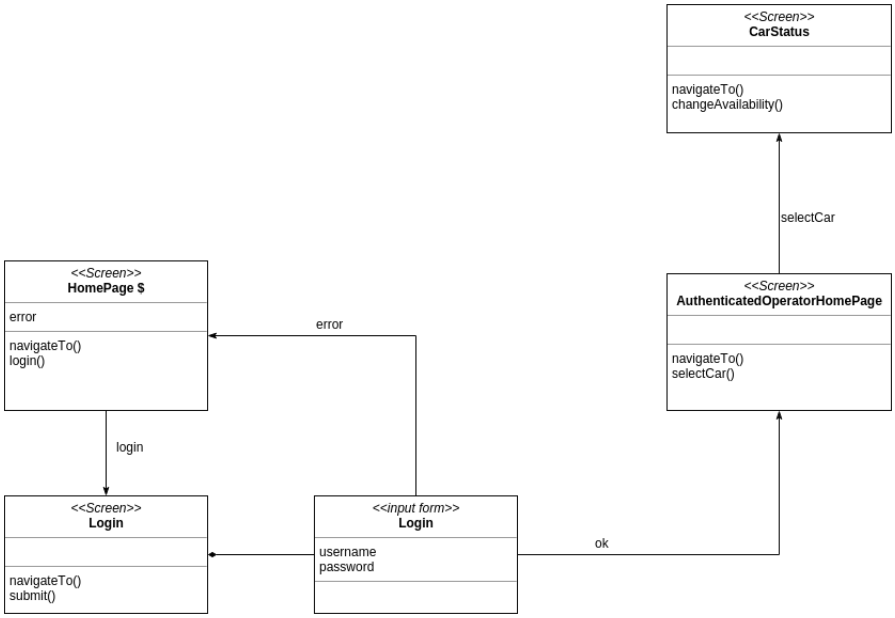


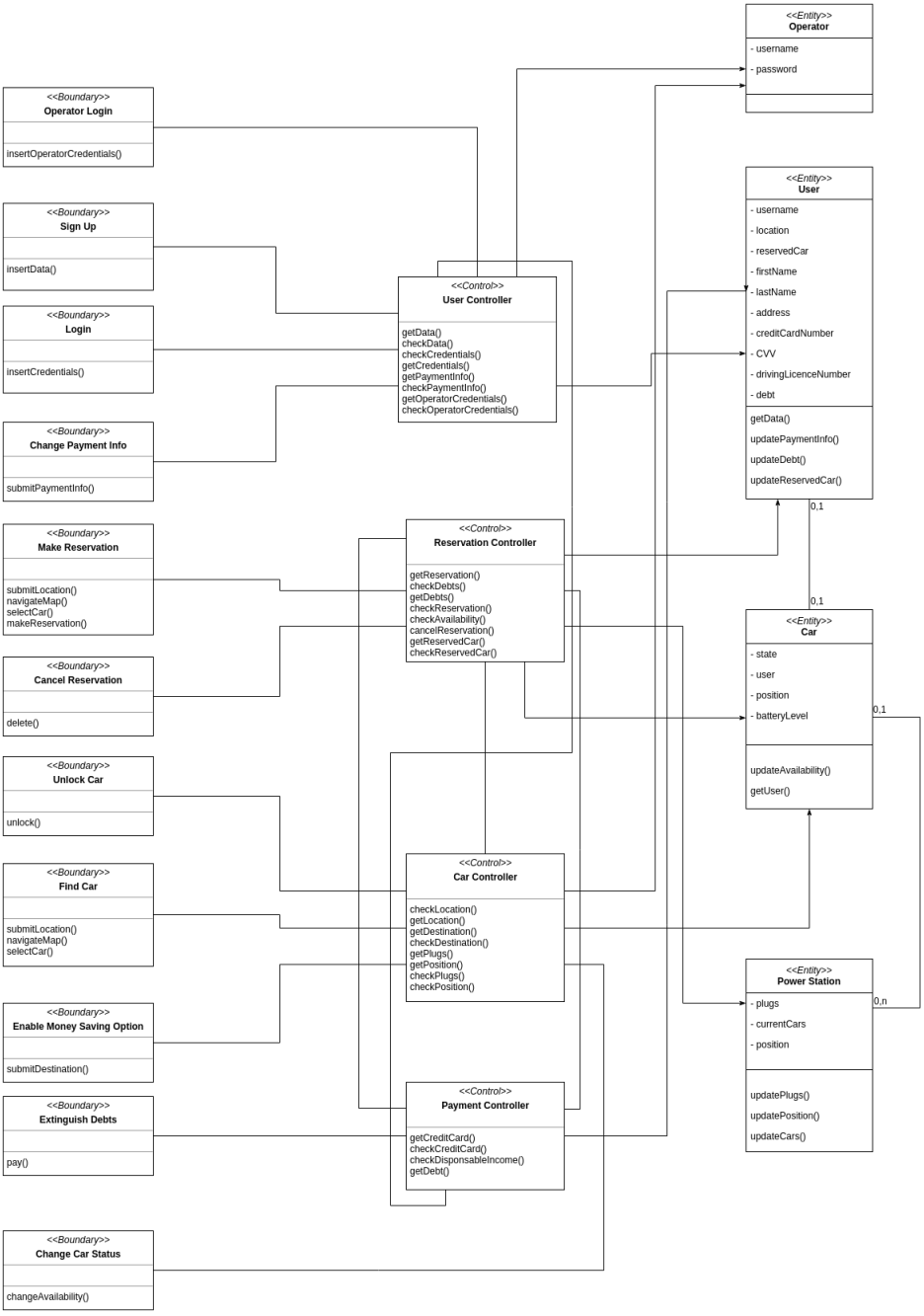

ABC
11:36 AM
POWERENJOY
POWER
5, Talbot Square
Battery Level: 5%
STATUS: LOW BATTERY
set to unavailable

## 4.2 Ux diagram




<<Screen>>
CarStatus
navigateTo()
changeAvailability()
selectCar
<<Screen>>
AuthenticatedOperatorHomePage
navigateTo()
selectCar()
<<Screen>>
HomePage $
error
navigateTo()
login()
error
login
<<Screen>>
Login
navigateTo()
submit()
<<input form>>
Login
username
password
ok

## 4.3 Bce diagram



# 5 Requirement traceability

[G1] Allow users to sign up, providing their driver license number, payment information and a username

- Client App
- Client Controller
- Database

[G2] Allow users to sign in

- ClientApp
- Client Controller
- Database

[G3a] Show available cars within a certain distance from current location
[G3b] Show available cars within a certain distance from a specified address

- ClientApp
- Map provider
- Client Controller
- Database

[G4] Let users reserve a single car

- ClientApp
- Client Controller
- Reservation Controller
- Database
- Car controller
- Car Handler

[G5] If a car is not picked up within an hour from the reservation, the system tags the car as available again, and the reservation expires; the user pays a fee of 1 EUR

- Client App
- Reservation Controller
- Car Controller
- Car Handler

- Payment Controller
- Database

[G6] A user that reaches a reserved car must be able to tell the system he's nearby, so the system unlocks the car and the user may enter. From that moment on, the user periodically pays an additional amount of money after a predefined time if he/she doesn't start the engine

- Client App
- Reservation Controller
- Car Controller
- Car Handler
- Database

[G7] As soon as the engine ignites, the system starts charging the user for a given amount of money per minute; the user is notified of the current charges through a screen on the car

- CarHandler
- ReservationManager
- Car Controller

[G8] The system stops charging the user as soon as the car is parked in a safe area and the user exits the car; at this point the system locks the car automatically. Then the system submits to the payment handler the total amount of money that the user has to pay

- Reservation Controller
- Car Controller
- Car Handler
- Payment Controller
- Database

[G9] The final amount of money can be increased or decreased depending on the user behaviour. Parking too far from a power grid station or leaving a car with low battery will result in a penalty. Leaving the car with a certain amount of battery charge, taking other passengers onto the car or plugging the car into the power grid inside a safe zone will result in a benefit. An operator must be notified when a car is left with low battery charge and when a car is left ouside the safe area form more than a predefined time

- Reservation Controller

- Car Handler
- Car Controller
- Operator Controller
- Operator App

[G10] If the user enables the money saving option, he/she can input his/her final destination and the system provides information about the station where to leave the car to get a discount. This station is determined to ensure a uniform distribution of cars in the city and depends both on the destination of the user and on the availability of power plugs at the selected station

- Client App
- Reservation Controller
- Power station Manager

G[11] If the payment handler notifies the system about incorrect payment informations or about an insufficient balance, the user is prevented from reserving other cars. He can change his payment information and communicate to the system that he wants to pay off his debt. If the payment doesn't go wrong, the user can reserve cars again

- Payment Handler
- Payment Controller
- Database
- Reservation Controller
- Client Controller
- Client App

G[12] The user must be able to cancel his reservation until he starts the engine. He still has to pay a fee of 1 EUR

- Client App
- Reservation Controller
- Car Controller
- Car Handler
- Database
- Payment Controller

[G13] Every operator must be able to change any car status to unavailable. Then they must be able to revert the status back to available

- Operator App
- Car Controller
- Car Handler