POLITECNICO DI MILANO

COMPUTER SCIENCE AND ENGINEERING

MASTER OF SCIENCE

RASD

PowerEnjoy

Software engineering II project

Giovanni Agugini, Matteo Foglio, Tommaso Massari

Contents

1 Introduction **4**
1.1 Description of the given problem 4
1.2 Goals 4
1.2.1 Clients 4
1.2.2 Operator 5
1.3 Domain properties 5
1.4 Glossary 6
1.5 Text assumptions 7
1.6 Constrains 8
1.7 Proposed system 9
1.8 Identifying stakeholders 9
1.9 Reference documents 9

2 Actors identifying **10**

3 Requirements **10**
3.1 Functional requirements 10
3.1.1 Clients 10
3.1.2 Operators 12
3.2 Non functional requirements 14

4 Scenario identifying **26**
4.1 Scenario 1 26
4.2 Scenario 2 26
4.3 Scenario 3 26
4.4 Scenario 4 26
4.5 Scenario 5 26
4.6 Scenario 6 26
4.7 Scenario 7 27
4.8 Scenario 8 27
4.9 Scenario 9 27
4.10 Scenario 10 27

5 UML models **28**
5.1 Use Case Diagram 29
5.2 Use case description 31
5.2.1 Not registered client make registration 31
5.2.2 Client Log in 32
5.2.3 Client edits his personal data 32
5.2.4 Client Logout 33
5.2.5 Car selection 33
5.2.6 Car reservation 34
5.2.7 Cancel Reservation 34
5.2.8 Reservation Expiration 35

5.2.9 Rent car . . . 35
5.2.10 The client activate the “Money Saving Option” . . . 36
5.2.11 The client selects the destination . . . 36
5.2.12 The client calls the Call Centre . . . 37
5.2.13 End rental . . . 37
5.2.14 Payment . . . 38
5.2.15 Blocked client complete payment request . . . 39
5.2.16 Operator log in . . . 39
5.2.17 Operator Logout . . . 40
5.2.18 Move car . . . 40
5.2.19 Charge car . . . 40
5.2.20 Set car status . . . 41
5.2.21 Validate client . . . 42
5.3 Class Diagram . . . 43
5.4 Sequence Diagram . . . 44
5.5 BPMN . . . 45
5.6 Activity Diagram . . . 46
5.7 State Chart . . . 47
5.8 Other diagrams . . . 48
5.8.1 Traceability matrix . . . 48

6 Alloy Modeling **49**
6.1 Model . . . 49
6.2 Results . . . 55
6.3 World generated . . . 56
6.4 Metamodel . . . 57

7 Hours of work **58**

1 Introduction

1.1 Description of the given problem

We will project and implement PowerEnjoy, a service of car-sharing with electric cars located in a limited area. The service is mainly addressed to one target of people, i.e. the clients, but also an operator team has to deal with it for the maintenance of the cars and for the administation of the clients. The users will have the possibility to reserve an electric car via web app thanks to the GPS system that will let users to find the nearest car from a current position or a specified address. The main purpose of the system is to be efficient, usable and eco-friendly. To reach this purpose, the service we are about to develop will have:

- electric cars with energy derived from renewable sresource
- users incentivated to well behave, in sense of taking more people in a car and charging the car when necessary
- optimized ditribution of cars

1.2 Goals

The goals of our project can be divided by grouping them for each actor of the system:

1.2.1 Clients

- [G1] Allows clients to find the location of the cars and the distance from their current position or from a specified address
- [G2] Allows clients to reserve an available car
- [G3] Allows clients to access the vehicle when a reservation is made and they are nearby
- [G4] Allows clients to use the GPS service of the on-board computer in order to get traffic indications on both the destination and the power grid stations
- [G5] Encourages clients to have an eco-friendly behaviour applying discounts or fees associated to specific actions
- [G6] Allows clients to visualize the current fare during the drive
- [G7] Allows the users to enable the "money saving" option in order to guarantee an uniform distribution of vehicles in return for discount

1.2.2 Operator

- [G8] Allows operators to localize cars with empty battery in order to put them in charge
- [G9] Suggests the operators where to move cars in order to have a uniform distribuition of vehicles
- [G10] Allows operators to get access of all cars
- [G11] Allows operators to change the state of a car
- [G12] Allows operators to approve the request of registration of a client

1.3 Domain properties

We suppose that these properties hold in the analyzed world

- The system always knows the current position of each car
- The GPS of user's mobile phone will always give the right position
- The GPS of the on-board computer will always give the right position
- The on-board computer is always on
- Given any GPS coordinates they always belong to either a safe area or a non-safe area
- The set of safe areas for parking cars is pre-defined by the management system
- When a client applies for registration, an operator will check the validity of documents sent by the client in order to accept or reject the request. Until the operator's approval, the client will not be able to rent a car
- If one of ID or driving license expires, the client's account will be suspended until he uploads the new document and an operator validates it
- Any legal issue related to revocation of driving license or violation of the traffic rules is the responsibility of the user
- All cars in system, except the ones in the "Extraordinary maintance" state, are working
- All grid stations are in Safe areas

1.4 Glossary

- User: can be both a client and an operator
- Client: the end user of the system (see Actor Identifying)
- Operator: an employee of PowerEnjoy (see Actor Identifying)
- Inactive Client: it's a client that has not completed the registration yet or that has been banned. An inactive client cannot rent cars
- Active client: it's a client able to use the full service
- Safe Area: is a set of delimeted positions where users can end the rental and park cars
- Unsafe Area: every area that is not a Safe Area
- Grid station: it's the place owned by PowerEnjoy where users can put cars in charge
- Validation: it's the approval of an account of a client by an operator who is in charge of verifying the validity of the uploaded documents
- Rental: it's the entire process composed by:
  - Reservation: it's the process that begins when a client books a car and ends when the Ride begins or the reservation is canceled
  - Ride: it's the process that begins when the client, after a previous reservation, opens the car and ends when the client communicates that the rental is over
- State of a car: it represents the condition of the car in our system. A car must in one and only one of the following state:
  - Available car: the car is working and has not been reserved yet, meaning that a client is allowed to reserved it
  - Unavailable car: a car in this state is not available for reservations due to different possible circumstances specified by one of the following sub-state:
    - In-use: a car in this state has been already reserved by a client who is reaching it or driving it. This state imply that no client can interact with this car except the one who is currently using it
    - Empty-battery: a car in this state has a battery charge that is less then a fixed percentage. An intervention of an operator is required in order to take the vehicle to a grid station or to recharge it on-site.

    * Ordinary maintance: a car in this state requires the intervention of the external maintanance society for a routine cleaning or inspection
    * Extraordinary maintance: a car in this state requires the intervention of the external maintanance society in order to fix technical issues or replace the car with another one

- Uniform distribution: intended as best optimized distribution in function of the density of usage per zone

1.5 Text assumptions

- PowerEnjoy society relies on an external company for the management and the maintance of the cars. The deal among the companies can be summarized in the following points:
  - CleanCar society will provide to PowerEnjoy a fixed number of cars meaning that in case of both ordinary and extraordinary maintance, CleanCar society will be in charge of solving the issue as soon as possible by fixing the car or providing a new one with full transparency for PowerEnjoy system
  - Ordinary maintance will be carried out by CleanCar society following these steps:
    * The state of a car is set to "Ordinary maintanance" by an operator or an automatic routine running on our system or
    * CleanCar society will do the maintenance of the car
    * CleanCar society will set the state of the car to "Available"
  - In case of exceptional events that prevent the usability of a vehicle, like accidents or damages to cars, the procedure followed by the company will require the manual intervention of an operator in the following steps:
    * The operator will be notificated of the breakage of the vehicle via telephone
    * The operator will set the state of the car to "Extraordinary maintanance"
    * The operator will call the external maintance society
    * The external society will fix the car or replace it with another one
    * The external society will notify our operator that the problem is solved
    * The operator will set the state of the car to "Available"

- In the case that a client is not in a safe area and the car battery level reaches 2%, the car stops but the car system (on board screen, GPS,

etc) continues to work for several days. The client must call the Call centre and an operator proceed in setting the car status to "Extraordinary maintenance". Such an event is considered the same as an accident.

- PowerEnjoy society has a call center service in order to manage extraordinary events and provide help to clients

1.6 Constrains

Regulatory policies

The system must require to the client some personal data. Since these are sensible data, the system must manage them according to privacy laws.

Hardware limitations

- Web app
  - Client
    - 3G connection or above
    - One of the following:
      - Browser with AJAX support
      - Space for app package
    - Optional
      - NCF
      - Bluetooth
  - Operator
    - NFC
    - 3G connection or above
    - Space for app package
- On-board computer
  - 3G connection or above
  - GPS system
- On-board technologies
  - NFC
  - Bluetooth

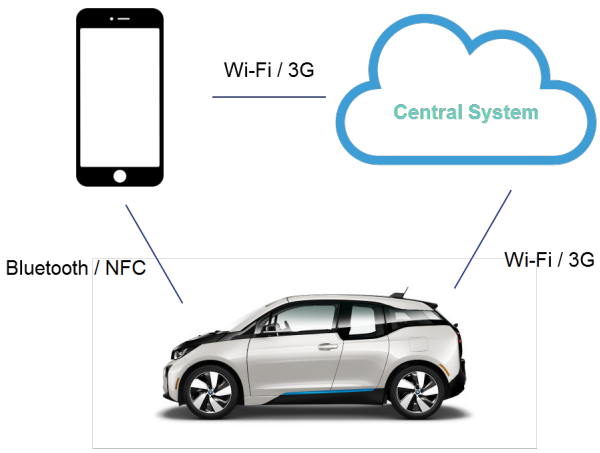
Parallel operation

The system supports parallel operation for different clients and operators and the same time.

1.7 Proposed system

Our architecture is a client-server type. It is based on MVC pattern in order to manage just one server application that can be accessed from both the web-application and the web-app for mobiles. These applications are developped with responsive web design techniques.

Figure 1: Architecture



1.8 Identifying stakeholders

Our only stakeholder is the mayor of the city. He wants to improve the road traffic by inducing people to fill cars and reducing the number of mono-passenger cars. He also wants to improve the overall air quality by reducing the number of gas powered vehicle and introducing more electric vehicles. He also wants to sensitize people on share-based transportation.

1.9 Reference documents

- RASD sample from Oct. 20 lecture.pdf
- IEEE Recommended Practice for Software Requirements http://ieeexplore.ieee.org/xpl/mostRecentIssue.

2 Actors identifying

The actors of our system are the following:

- Client: he is the end user of PowerEnjoy company. He interact with the system, even if he has not completed his registration yet
- Operator: he is an employee of PowerEnjoy designated to task that involves an interaction with the system

3 Requirements

3.1 Functional requirements

Assuming that the domain properties stipulated in the paragraph [1.3] hold, and, in order to fulfill the goals listed in paragraph [1.2] , we have derived the following functional requirements. Each requirement is referred to the satisfied goal. Given that the goals are divided by actors, we continue to use this order to list the functional requirements.

3.1.1 Clients

[G1] Allows clients to find the location of the cars and the distance from their current position or from a specified address:

- The system must show to the user a map of the defined zone with all the available cars
- The system must be able to detect the current position of the user and the position of the cars through GPS.
- The system must indicate through a correct use of GPS the distance between a car and the current position just by clicking on the car in the map
- The system must also give the possibility to enter a specified address and view the distance between a car and that address always by clicking on the wanted car on the map
- The system must be able to indicate the time (on foot) in order to reach the car from the current position or from the specified address

[G2] Allows clients to reserve an available car:

- The system must let the user reserve a car by clicking on a car in the available cars map
- The system must start a timer of one hour when the user decides to book a car

- The system must show the timer to the user
- The system must be able to dismiss the booking of a car when the one hour timer is over and to charge 1 euro to the user
- The system must indicate the best path to reach the booked car from the current position or a specified address

[G3] Allows clients to access the vehicle when a reservation is made and they are nearby:

- The system must let the user communicate that he/she is nearby the car
- The system must unlock the car if and only if the verification is successful, this happens if one of the following events take place:
  - the client brings his/her phone (with NFC activated) close to the NFC tag placed on the car
  - the client pairs the phone with the car through bluetooth
  - the client types the car's plate into the text box
- The system must ask the user to enter a PIN on the on-board computer

[G4] Allows clients to use the GPS service of the on-board computer in order to get traffic indications on both the destination and the power grid stations:

- The system must provide a GPS service in order to give the user traffic indications on his/her final destination
- If the user want the GPS service about the final destination, the system must ask the user to enter the destination
- The system must offer a GPS service in order to give the user the chance to leave the car in a power grid station near his final destination and to charge it

[G5] Encourages clients to have an eco-friendly behaviour applying discounts or fees associated to specific actions:

- The system must be able to control the battery status before and after each ride
- The system must be able to detect through apposite sensors the presence of other passengers
- The system must charge 30% more on the last ride if the user leaves the car with more than 80% of battery empty
- The system must charge 30% more on the last ride if the user leaves the car at more than 3 Km from the nearest power grid station

- The system must apply a 10% discount on the last ride if the user takes at least two other passengers
- The system must apply a 20% discount on the last ride if the user leaves the car with no more than 50% of battery empty
- The system must apply a 30% discount on the last ride if the user leaves the car in a power grid station and provides to charge it
- The system must apply a 35% discount on the last ride it the user leaves the car in a power grid station and provides to charge it also by guaranteeing an uniform distribution of the car

[G6] Allows clients to visualize the current fare during the drive:

- The system must calculate the charging fare of the ride starting when the motor ignites
- The system must indicate through the on-board computer the charging fare of the ride
- The system must also calculate the total cost of the ride after having checked all the possible discounts and fees
- The system must send an email to the user with the receipt of the last ride

[G7] Allows the client to enable the "money saving" option in order to achieve a uniform distribution of vehicles in return for discount:

- The system must let the user choose on the on-board computer the "money saving" option
- The system must indicate through the GPS service the best path in order to arrive to the power grid station
- The system must indicate through the GPS service the best path in order to park in an area and so achieve a uniform distribution

3.1.2 Operators

[G8] Allows operators to localize cars with empty battery:

- The system must be able to know for each car the state of its battery
- The system must check at the end of each ride the state of the battery
- The system must change the state of a car to "empty battery" when a car is more than 80% of battery empty
- The system must use a GPS service with which it offers to the operator a map with all the empty battery cars

[G9] Suggests the operators where to move cars in order to have a uniform distribuition of vehicles:

- The system must indicate through the GPS service where the operator has to leave the empty battery car in order to guarantee an uniform distribution of the cars
- The system must calculate how many cars and how many kilometers the operator has performed

[G10] Allows operators to get access of available and empty-battery cars:

- The system must let the operator communicate that he/she is nearby the car
- The system must verify through NFC (or Bluetooth) that the operator is really nearby the car
- The system must unlock the available or empty battery car if and only if the verification is successful

[G11] Allows operators to change the state of a car:

- The system must let the operator change the state of a car

[G12] Allows operators to validate the request of registration of a client:

- The system must show to the operator the pending requests of inscriptions to the service of all the users
- The system must let the operator confirm the inscription of an user after that the operator has verified the suitability of the user

3.2 Non functional requirements

Client interface

Figure 2: registration - App start

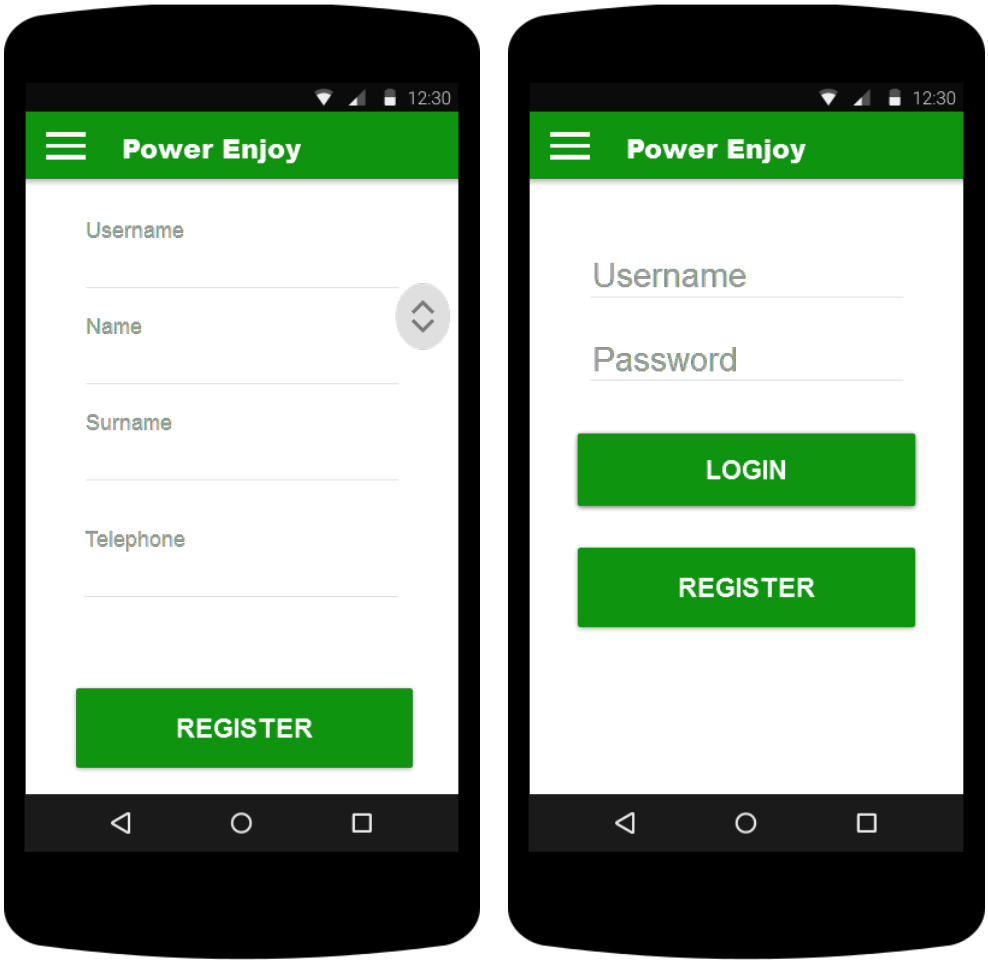


Figure 3: App Map - App Reserve

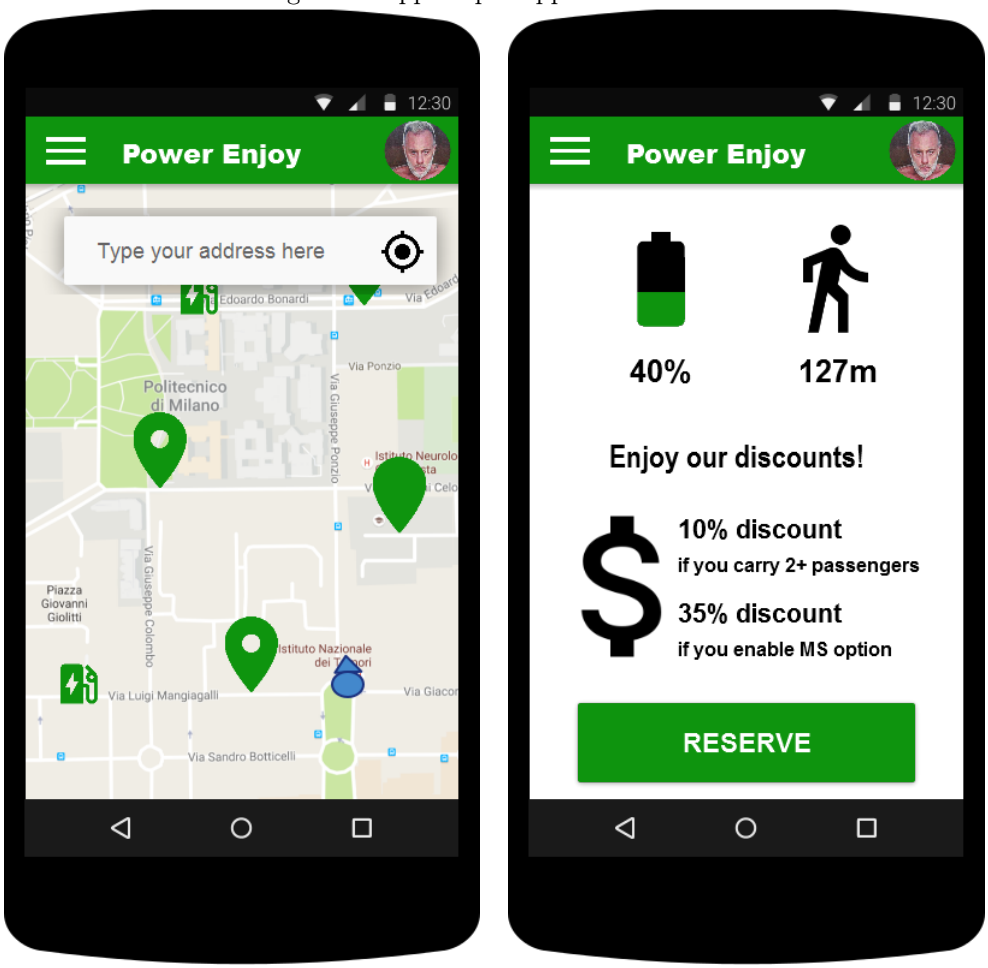


Figure 4: App Reservation - App Payment

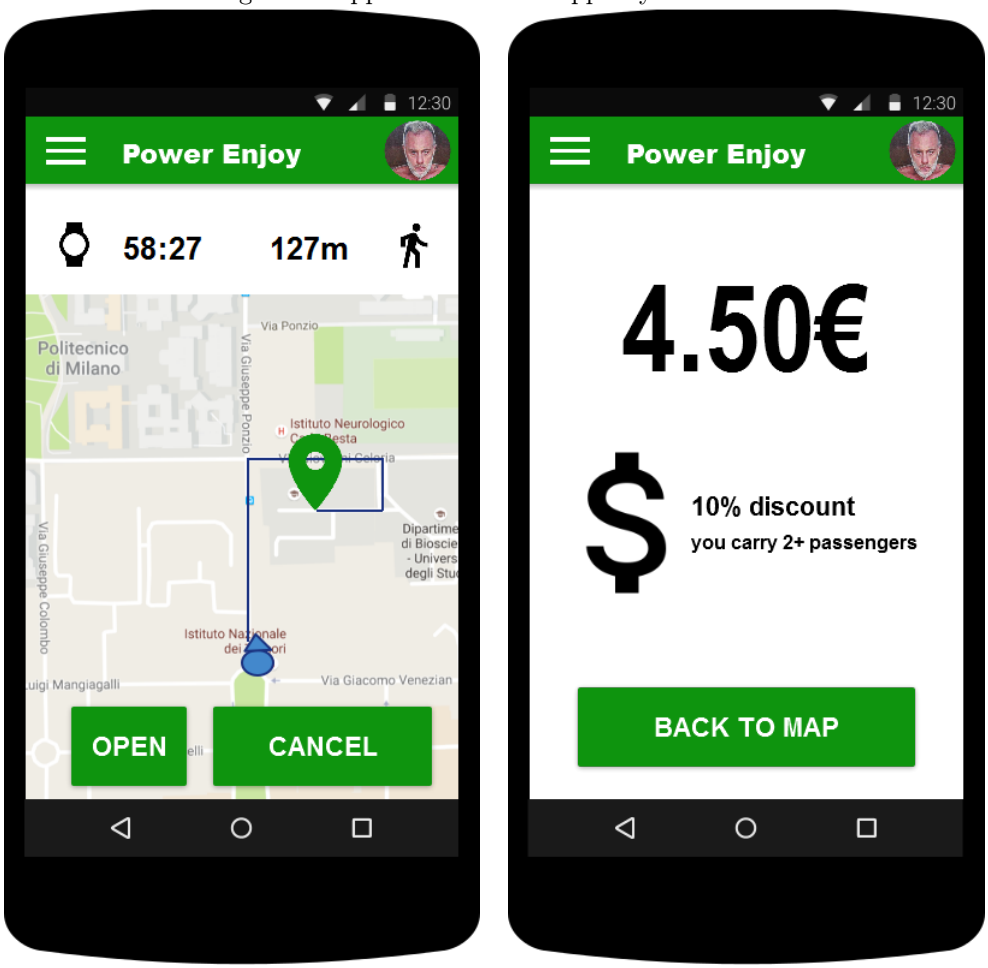


Figure 5: Edit personal data - Menu

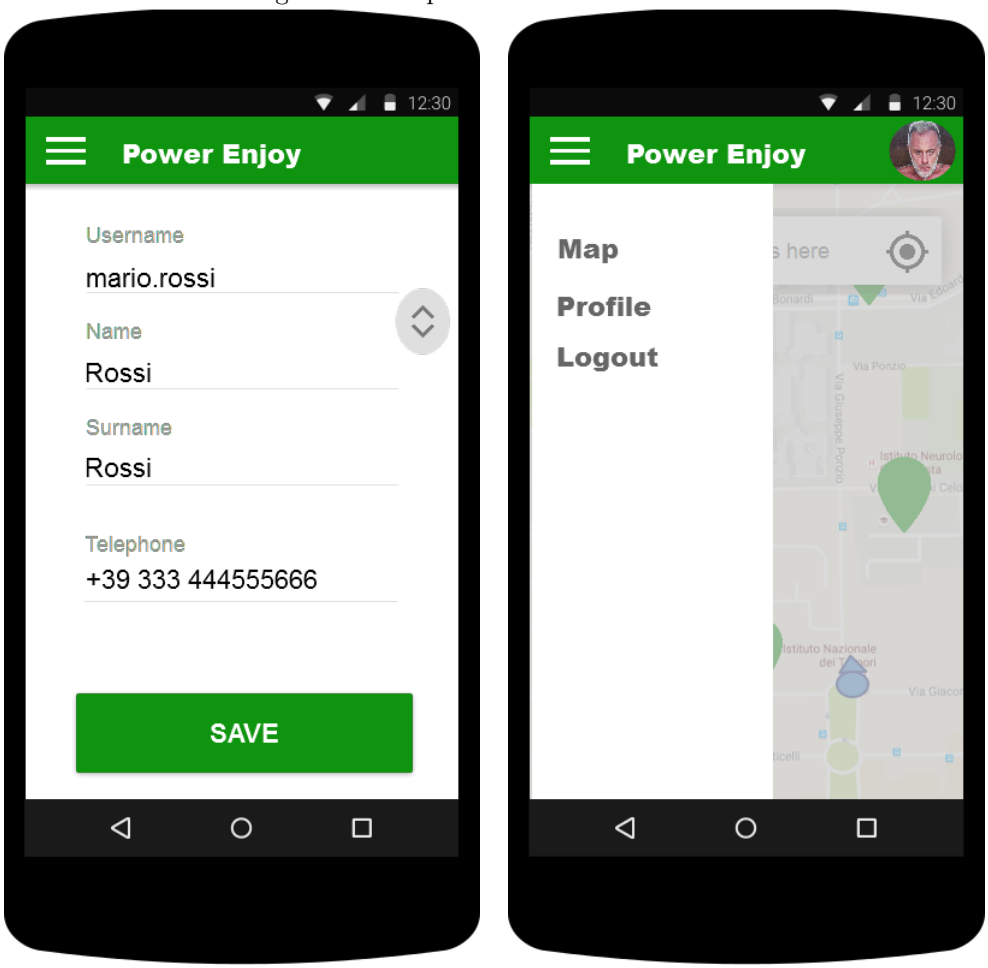
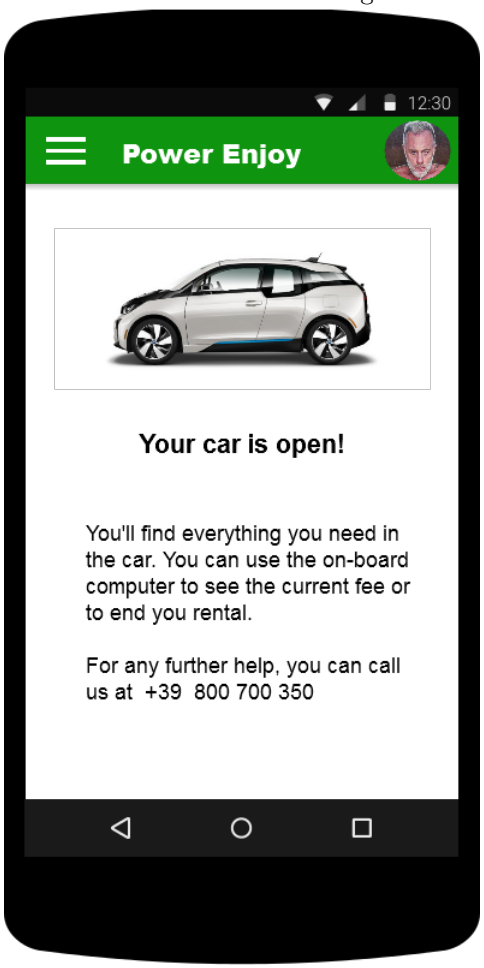


Figure 6: Rental page



Onboard interface

Figure 7: On board pin

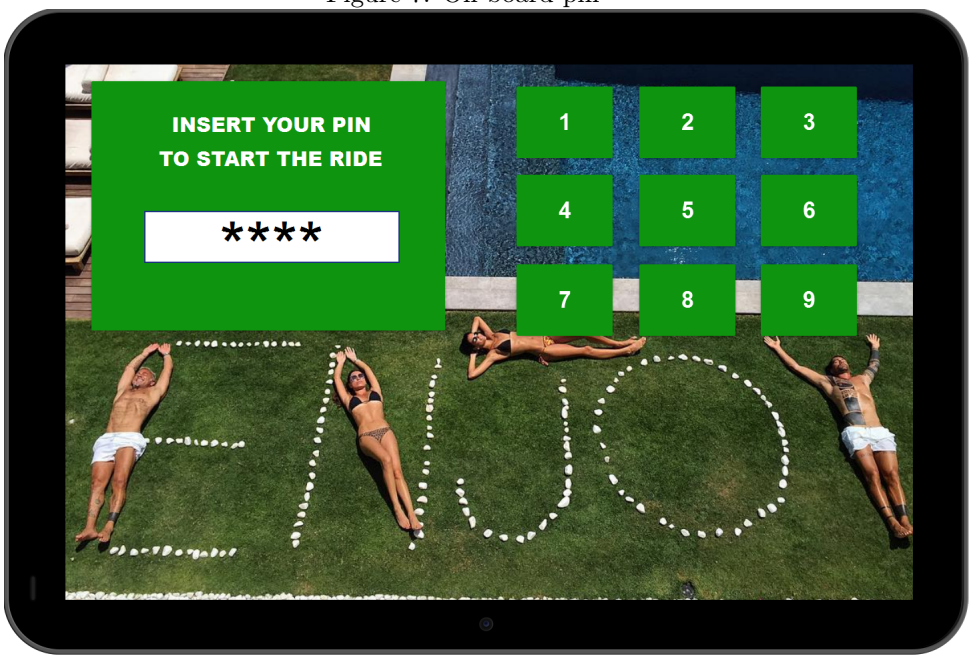


Figure 8: On board options

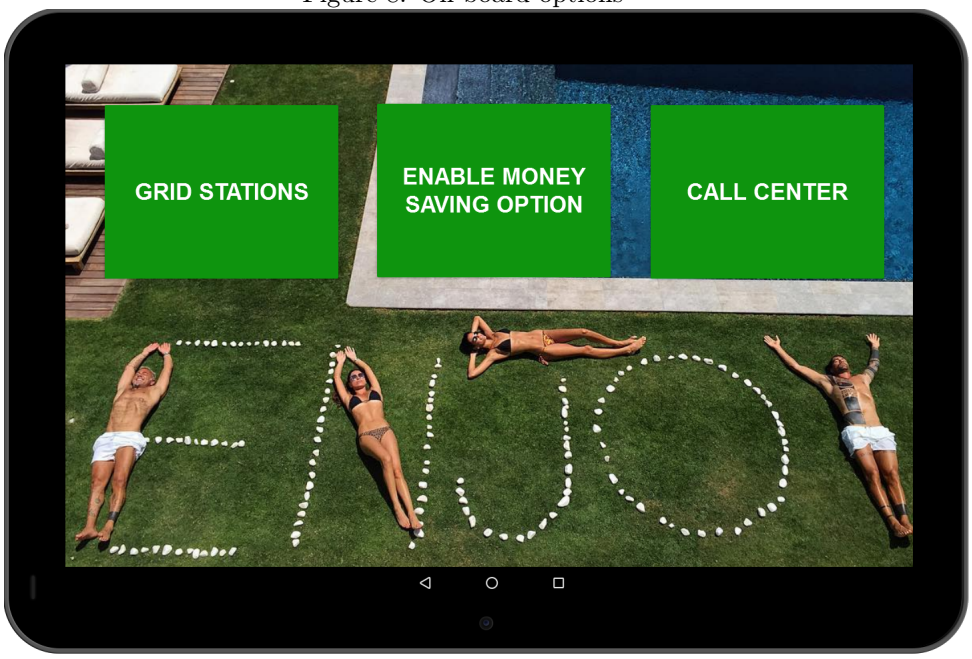


Figure 9: On board map

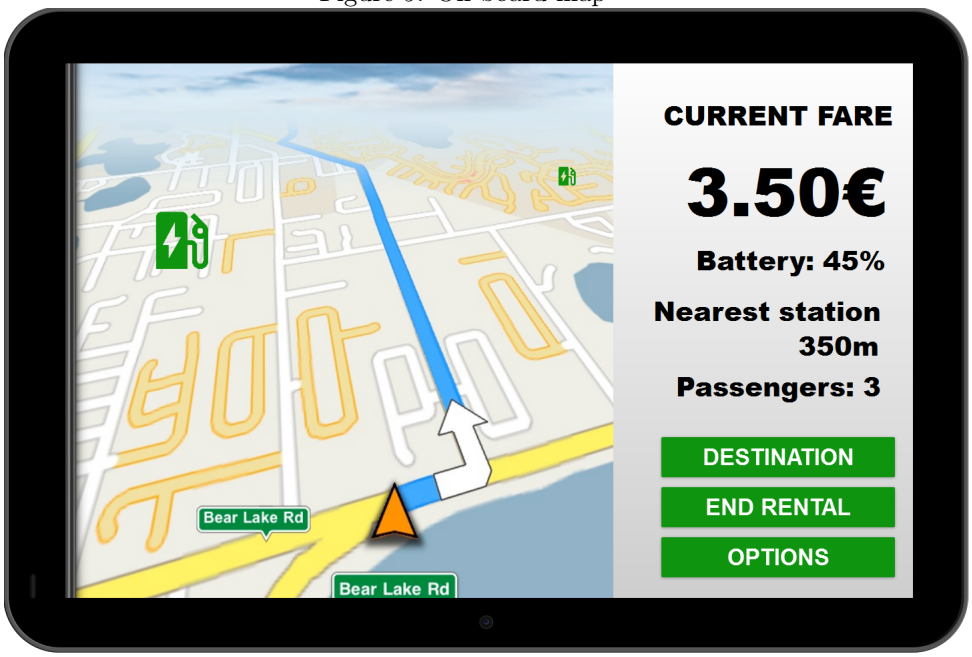


Figure 10: On board address

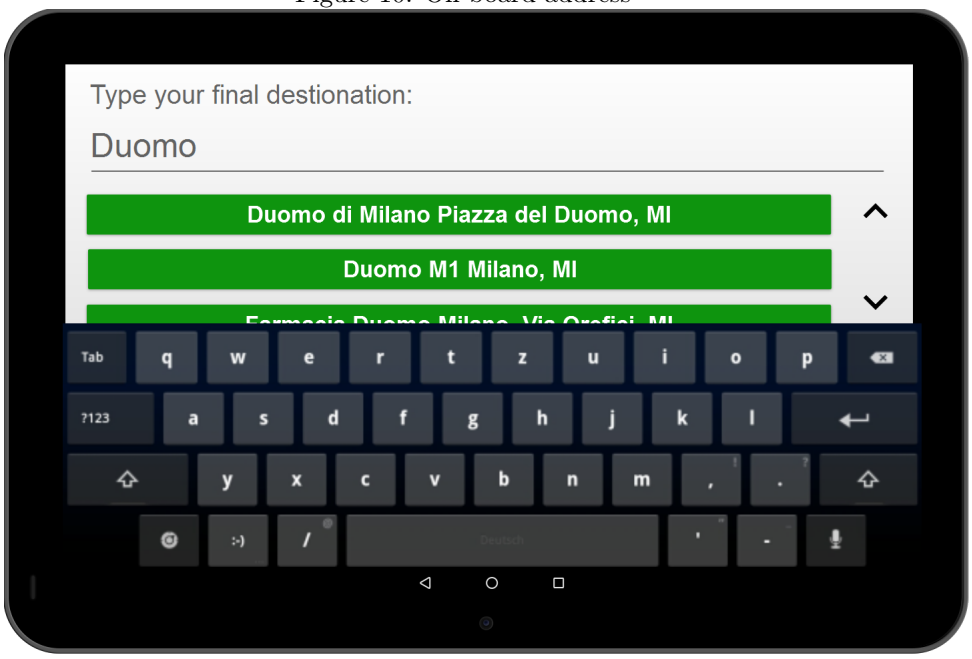


Figure 11: On board grid station

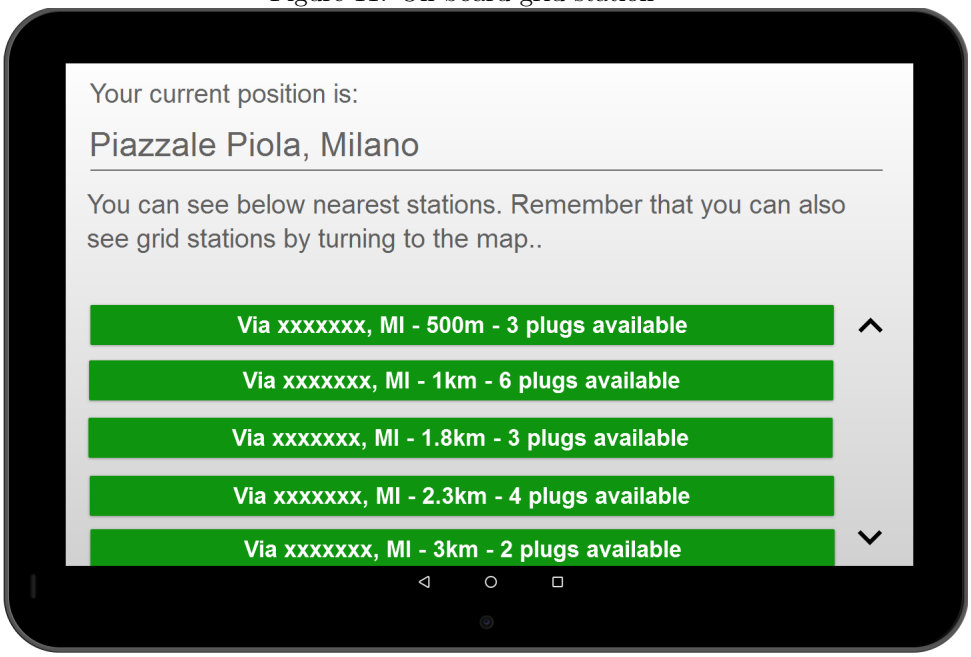
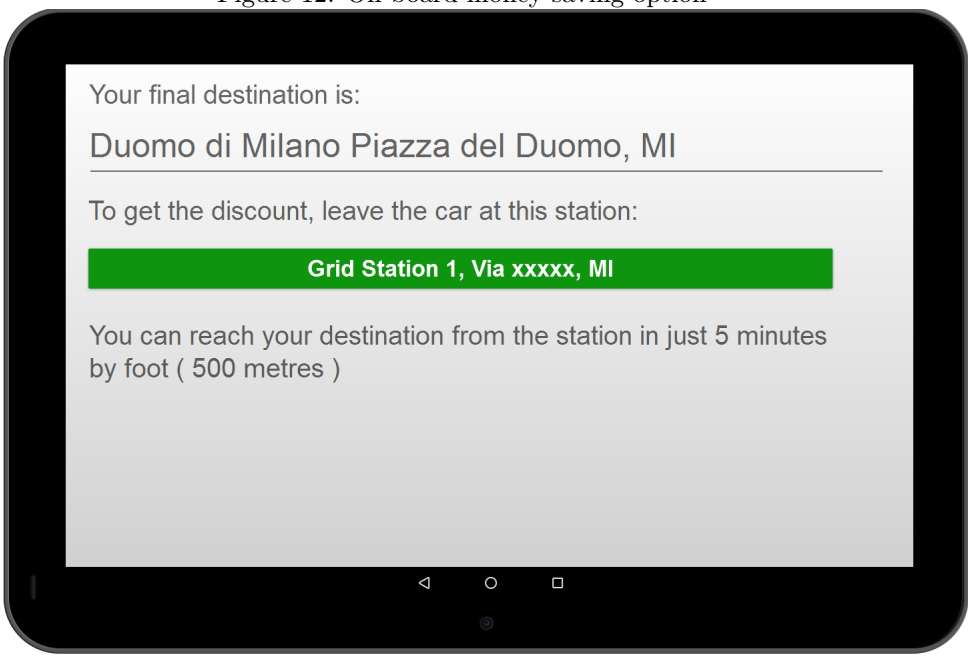


Figure 12: On board money saving option



Operator interface

As it will be shown in the Design Document some screens will be in common for the clients and operators.

Figure 13: Operator Login

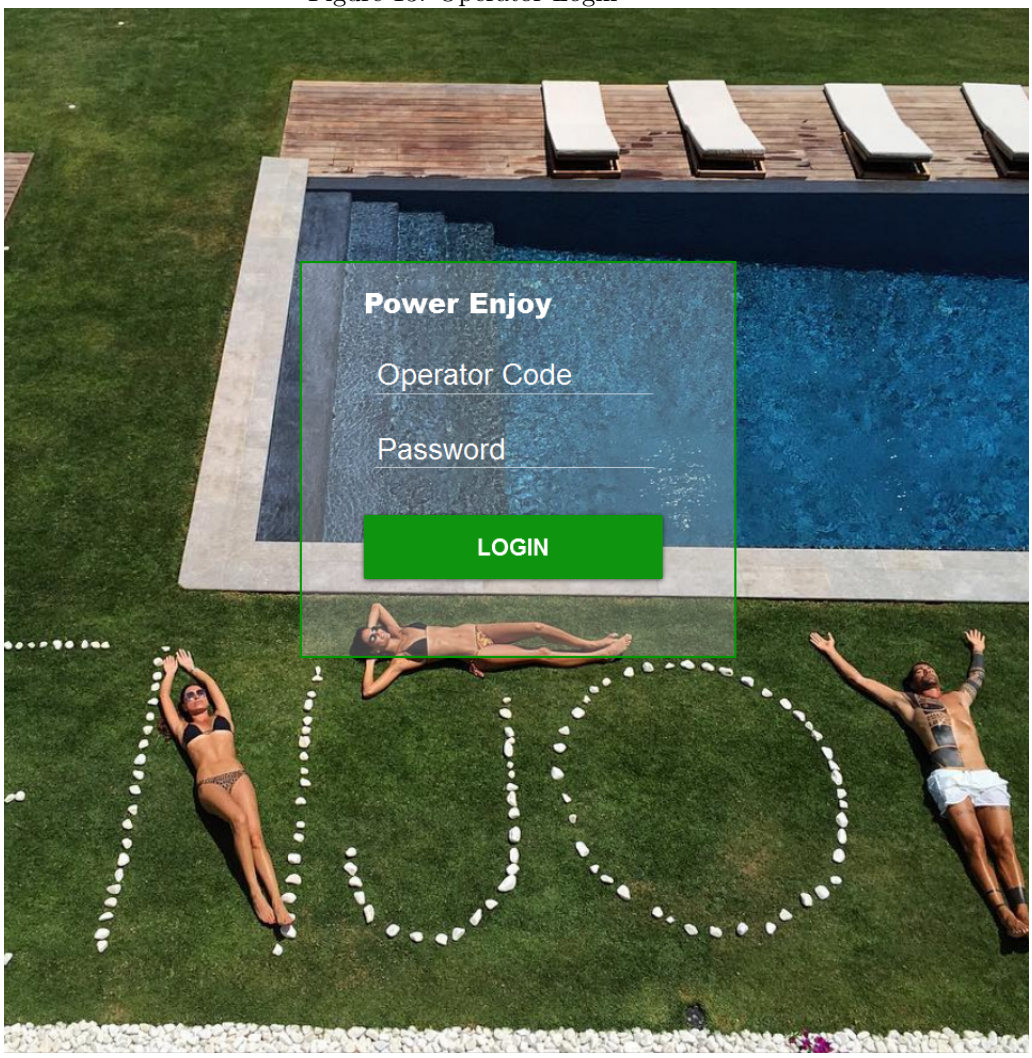


Figure 14: Operator Maps

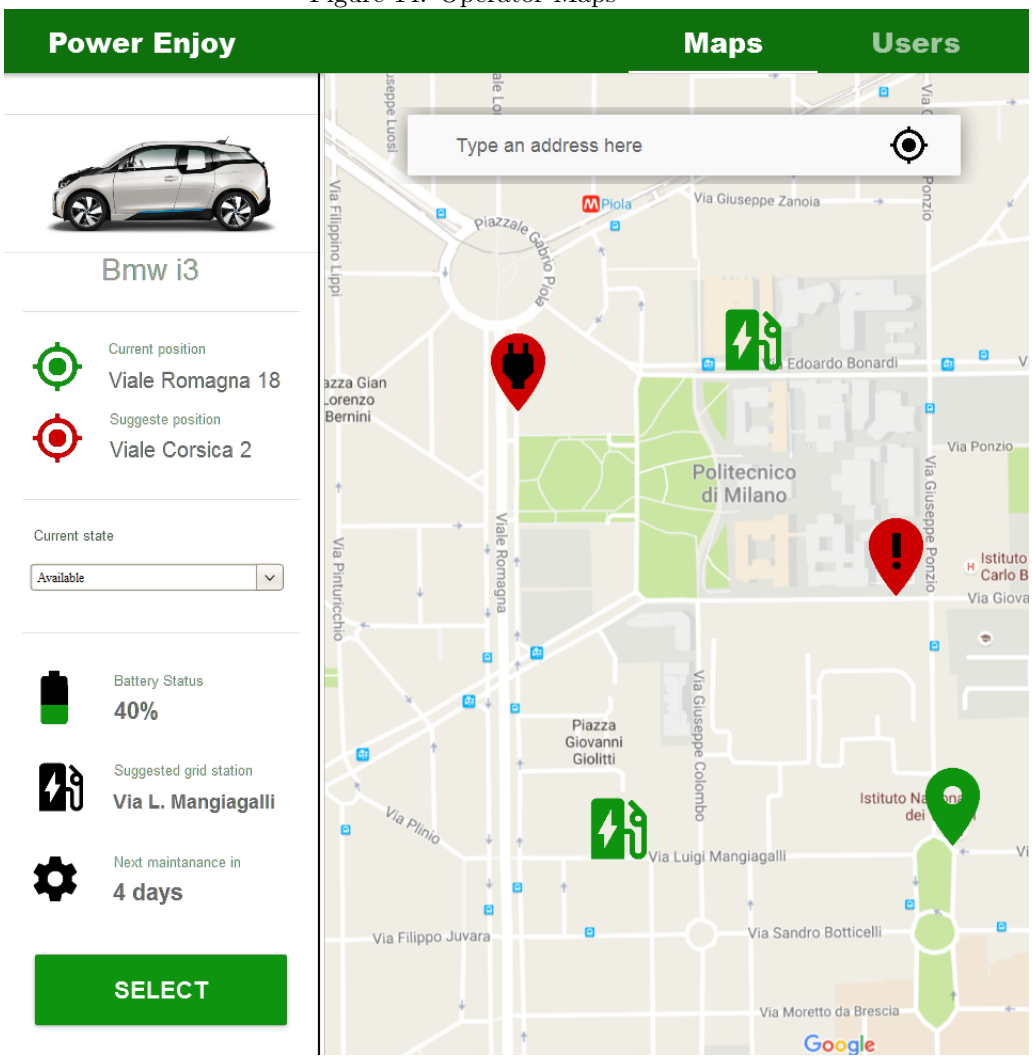
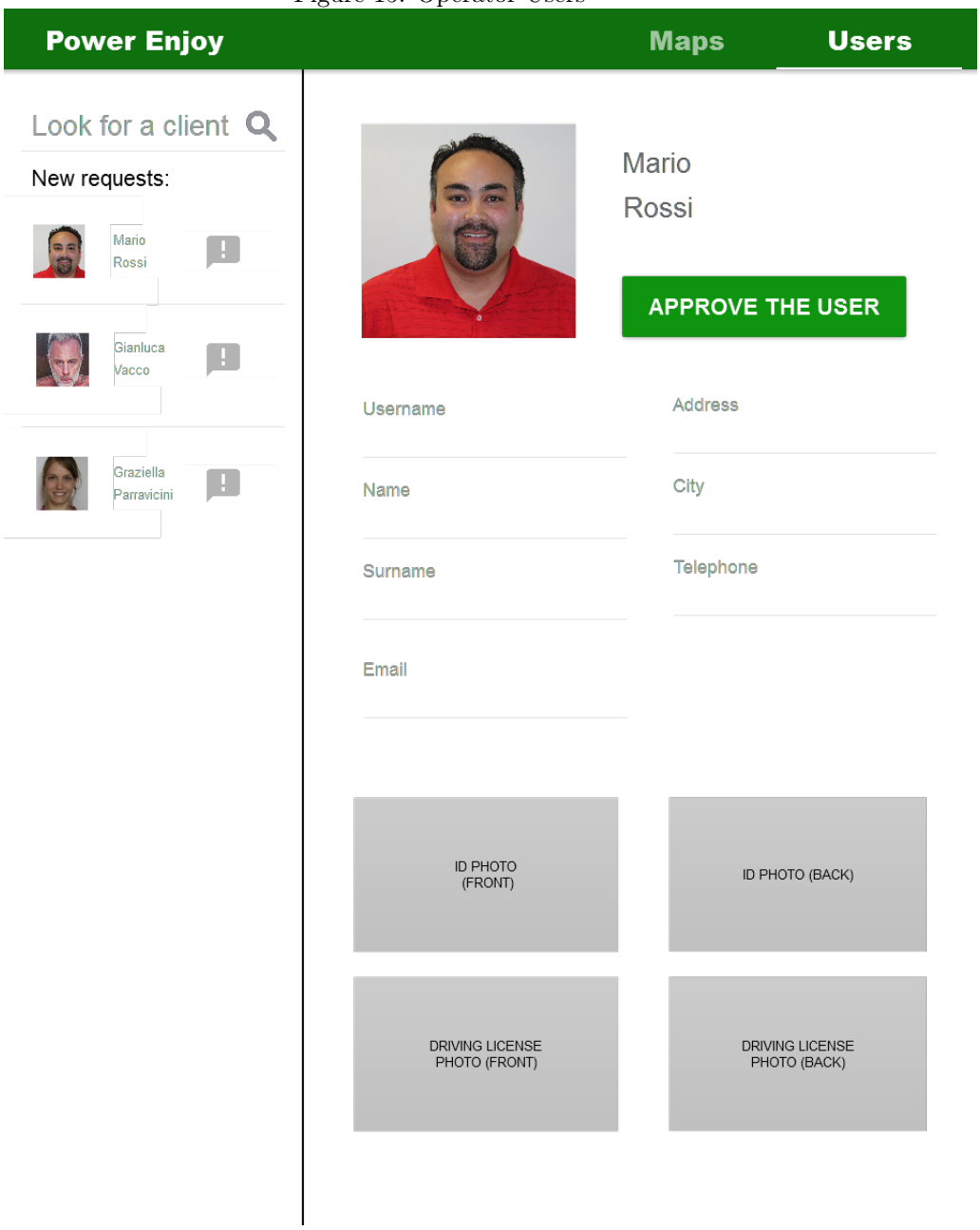


Figure 15: Operator Users



4 Scenario identifying

4.1 Scenario 1

Julia is in a work reunion and after she has a dinner with his father. Since she's in a hurry and she sees on the app that there is just one available car in her area, she reserves the car in advance without knowing when the reunion will end. After one hour she's still at work, so the service fines Julia and make again the car available. Lucklily, after the reunion Julia can reserve the same car, so she starts a ride with it.

4.2 Scenario 2

Today Mark has accomplished all of his job tasks in advance so he can go back home with calm. Mark is a frequent user at PowerEnjoy, and this time he decides to park the car in a power grid station and to charge it. After this action he receives a 30% discount.

4.3 Scenario 3

Peter leaves the car with more then 80% of battery empty. Peter receives a 20% addition on the total cost of service.

4.4 Scenario 4

Rudolf leaves the car at more the 3km from the nearest power grid station and he receives a 30% addition on the total cost of the service

4.5 Scenario 5

Donald is a young student who has just decided to suscribe to the PowerEnjoy service. On Monday he decides to pick a car but since he's not too much money he decides to enable th emoney saving option. Therefore, he parks in a power grid station by achieving an uniform distribution and he gets a 35% disount on the final price of the service

4.6 Scenario 6

Steve is a 20 years old student and he has only a prepaid card which is charged by his parents every month. Steve has subscribed his inscription to PowerEnjoy with his card. Today Steve has to reach his school mate for a project and he decides to grab a PowerEnjoy car. After the rental the system discovers that there are no money on the card of Steve. Steve is temporally banned from the service and he gets the news by mail. Steve contacts his parents and asks them to charge the card. After the charging he pays the rental and returns in the service.

4.7 Scenario 7

Bob wants to go shopping just outside the city because there is a big mall doing discounts on the weekend. Since his sister has taken the family car, he decides to use the PowerEnjoy service. The only problem is that the mall is outside the permitted boundaries, but he decides anyway to reserve a car. When he arrives at the mall, he wants to leave the car. The system through the on-board computer doesn't let Bob end the rent because he's outside the defined zone of the service. He can only put the car in parking mode and continue to pay a reduced fee while he's in the mall. Bob does it and afdter the shopping he retakes the car, goes inside the boundaries and leaves definitively the car.

4.8 Scenario 8

Daniel, Eva and Rob are three friends who decided to go to the cinema on Saturday. Since they live nearby, they decide to use PowerEnjoy. Daniel has got the subscription of the service so he reserves a car and then tells Eva and Rob where the car is. When all the friends arrive to the car, Daniel unlocks the car from the smartphone and then tells the sytem he's with two friends. At the end of the ride, Daniel and the others exit the car, then Daniel locks the car and after two minutes he receives the final prize of the ride with the 10% discount: the system has verified that Daniel was actually with other two friends.

4.9 Scenario 9

Frankie is an operator of the PowerEnjoy service. On Monday, Frank wakes up and goes to the office: he opens his laptop and sees that there are a lot of empty battery cars! He cannot leave those cars out of charge otherwise the service is going to be unefficient. So he opens his smartphone and sees on the map of empty battery cars where is the nearest one. When he arrives to the car he tells the system he's nearby so he can open the car. Frankie charges the car with the generator that he has brought and then uses the function on the on-board computer to park the car in a zone to guarantee an uniform distribution. At the end of the ride, Frankie uses his smartphone to change the status of the car from "empty battery" to "available". Frankie exits the car.

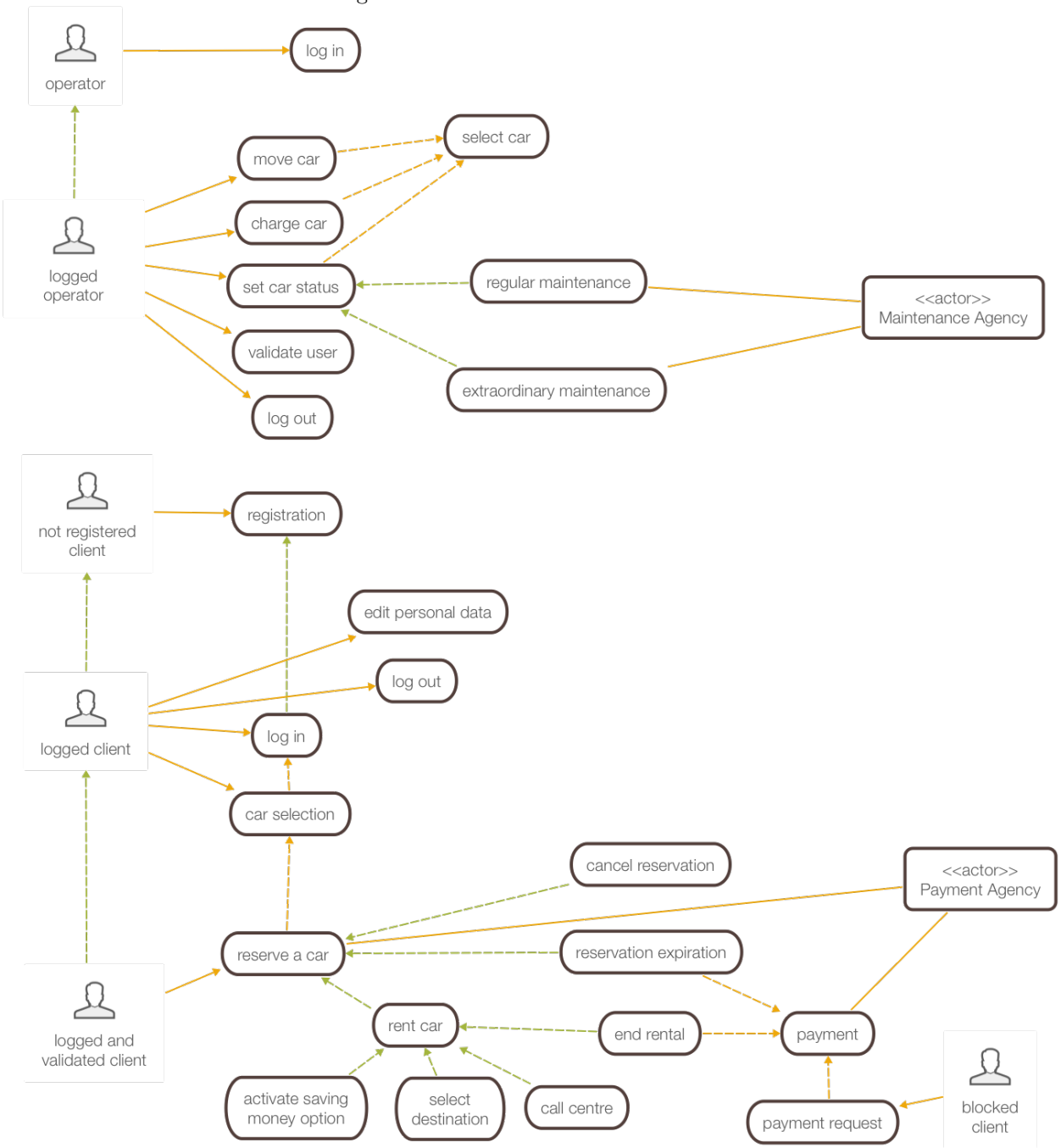
4.10 Scenario 10

Telma is an operator of the PowerEnjoy service. Telma usually works at the bureau in order to administrate and organize all the documentation. She has also the work to approve requests of the users. This day she opens her laptop and sees that there are two possible users requesting the inscription. Telma checks that the documents received are correct and appropriate. After this work she goes on the system and she approves the requests. Immediatly the system send an email to the users with the result of their requests. The users receive also a PIN to open the car.

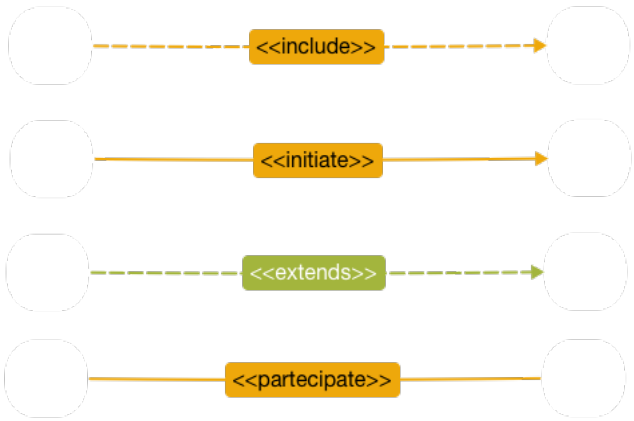
5 UML models

5.1 Use Case Diagram

Figure 16:



Legend



5.2 Use case description

5.2.1 Not registered client make registration

Name: Registration

Actors: Not regisered client

Entry conditions: No entry conditions

Flow of events:

- the client arrives in the website "StartPage" and clicks on the "Register" button
- he/she fills in the registration form. This includes:
  - name
  - surname
  - clientname
  - e-mail address
  - telephone number
  - address
  - city
- the client must fill in 2 more forms, one for each document (driver licence and ID) and upload a scan or a hi-res photo of back and front of each document. This includes:
  - release date
  - expiration date
  - document id number or code
  - 2 images (back and front)
- the client uploads a recent photo of himself
- the client insert payments data
- the client clicks on "Register"
- the system checks the correctness of the input data and sends a confirmation e-mail tho the client (in this e-mail is also provided the password)

Exit conditions: The documents are now in system and ready for the manual validation provided by an operator ("validate client" use case). The user is in "Map Page".

Exceptions: The system check find one or more errors. For example: not vaild e-mail address or phone, not vaild date, one ore more field is left blank, one or more pictures are missing. The system alert the client about the error and provides the possibility to re-fill the form and re-upload the photos.

5.2.2 Client Log in

Name: Log in

Actors: Client

Entry conditions: The client must be registered in the system

Flow of events:

- the client arrives in the website or the app "'StartPage" and clicks on the "Log in" button
- he/she fills in log in form, i.e. e-mail and password.
- the client clicks on "login" button
- the system redirects the client to his personal page i.e. "MapPage"

Exit conditions: The client is logged in

Exceptions: The e-mail or password typed are incorrect. The system advises the client of the error and allows him/her to re-type.

5.2.3 Client edits his personal data

Name: Edit personal data

Actors: Logged client

Entry conditions: The client must be logged in

Flow of events:

- the client clicks on "Profile" in the app menu
- the system redirect him/her on the "Edti personal data page" where all the information about him are shown
- the client edit his/her data by modifying them in a form or by uploading new photos of his/her documents
- the client click on "Save" button

- the system verifies if the client has update some documents due to their expirations and in this case

Exit conditions: The documents are now in system and ready for the manual validation provided by an operator ("validate client" use case). The user is in "Map Page".

Exceptions: The system check find one or more errors. For example: not vaild e-mail address or phone, not vaild date, one ore more field is left blank, one or more pictures are missing. The system alerts the client about the error and provides the possibility to re-fill the form and re-upload the photos.

5.2.4 Client Logout

Name: Logout

Actors: Logged client

Entry conditions: The client must be logged in

Flow of events:

- the client clicks on "Logout" in the app menu

Exit conditions: The client is in "Start Page".

Exceptions: No exceptions

5.2.5 Car selection

Name: Car selection

Actors: Logged client

Entry conditions: The client must be logged in and in the personal "Map-Page" where it's displayed a map with all nearby car's position

Flow of events:

- the client clicks on a car among the displayed ones
- the system redirect him/her on the "Car page" where all the information about the car are provided

Exit conditions: The client is in the "Car Page"

Exceptions: No exceptions

5.2.6 Car reservation

Name: car reservation

Actors: Logged and validated client

Entry conditions: The client must be logged in and his/her account must be validated. A car has to be already chosen. The client is in the selected "Car page".

Flow of events:

- the client clicks on the "reserve" button
- the system check through the Payment Agency the availability for the minimum requirements
- the system change the car status from "available" to "in use"
- the system redirect the client to the "reservation page"

Exit conditions: The car has been reserved and its status is now "in use". The client is in the "Reservation Page"

Exceptions: The system check fails because the client doesn't have enough money on his/her credit card. The system does not reserve the car nor change its status, and advises the client about the error. The client remains on the "Car page"

5.2.7 Cancel Reservation

Name: Cancel reservation

Actors: Logged and validated client

Entry conditions: A reservation has been made less than an hour before. The client is in the "Reservation page"

Flow of events:

- the client clicks on the "cancel reservation" button
- the system change the car status from "in use" to "available"
- the system redirect the client to his/her personal "Home page"

Exit conditions: The state of the car is returned to "available" and it's now visible to other clients. The client is on his/her personal "Home page"

Exceptions: No exceptions

5.2.8 Reservation Expiration

Name: Reservation expiration

Actors: Logged and validated client

Entry conditions: A reservation has been made and the maximum time (60 min) is reached without the "rent car" use case has happened

Flow of events:

- the system change the car status from "in use" to "available"
- the system redirect the client to the "MapPage"
- the system register a new payment for the client then the use case "payment" happens

Exit conditions: The state of the car is returned to "available" and it's now visible to other clients. The new payment is registered.

Exceptions: No exceptions

5.2.9 Rent car

Name: Rent car

Actors: Logged and validated client

Entry conditions: A reservation has been made less than an hour before. The client is in the "Reservation page"

Flow of events:

- the client reaches the car's proximity and one of the following events happen:
  - the client open the car door through NFC technology bringing his/her phone close to teh NFC tag placed on the car
  - the client clicks on the "open door" button and through bluetooth the phone pairs with the car

  - the client clicks on the “open door” button and types the car’s plate into the text box

- the system unlocks the door of the car

Exit conditions: The door’s car is unlocked. The client is in “Rental Page”.

Exceptions: No exceptions

5.2.10 The client activate the “Money Saving Option”

Name: Activate money saving option

Actors: Logged and validated client

Entry conditions: the client is in the “Rental page” and inside the car. The on board screen shows the “Map Page”

Flow of events:

- the client clicks on “options” button on the on-board screen
- the client clicks on “money saving option” button on the on-board screen
- the on board computer shows the available grid station where to park the car.
- the client clicks on the grid station name to confirm
- the system activate the money saving option for the ride
- the on board computer returns to the “Map Page”

Exit conditions: The money saving option is activated and the on board computer is in the “Map Page”

Exceptions: No exceptions

5.2.11 The client selects the destination

Name: select destination

Actors: Logged and validated client

Entry conditions: the client is in the “Rental page” and inside the car. The on board screen shows the “Map Page”

Flow of events:

- the client clicks on "destination" button on the on-board screen
- the client types the desired address on the on-board screen
- the on board computer shows the suggested destinations
- the client clicks on the destination to confirm
- the on board computer returns to the "Map Page" and shows the path to the destination

Exit conditions: The on board computer is in the "Map Page" and shows the path to the destination

Exceptions: No exceptions

5.2.12 The client calls the Call Centre

Name: call centre

Actors: Logged and validated client

Entry conditions: the client is in the "Rental page" and inside the car. The on board screen shows the "Map Page"

Flow of events:

- the client clicks on "options" button on the on-board screen
- the client clicks on "call centre" button on the on-board screen
- the on board computer calls the call centre
- when the call is concluded the on board computer returns to the "Map Page"

Exit conditions: The on board computer is in the "Map Page"

Exceptions: No exceptions

5.2.13 End rental

Name: End rental

Actors: Logged and validated client

Entry conditions: The client is in the "Reservation page"

Flow of events:

- the client clicks on "end rental" button on the on-board screen
- the client turns off the car, exit from the vehicle and closes the door
- the system locks the door of the car
- the system change the car status from "in use" to another state
- the system register a new payment for the client and calculate the price, then the use case "payment" happens

Exit conditions: The car's state is not "in use" any more. The new payment is registered. The client app is in the "Map Page".

Exceptions: The system detects somone still in the car or some door opened or the car still not off. The system alert the client of the error and provides information on how to solve it. Then it gives the client the possibility to re-click on "end rental" button.

5.2.14 Payment

Name: Payment

Actors: Logged and validated client

Entry conditions: The client is in the "Reservation page"

Flow of events:

- the system redirects the client to "the payment page"
- the client is on the "payment page"
- the system forward the payment to the Payment Agency
- the system sends a confimation e-mail to the client
- the client is redirected to his/her personal "home page"

Exit conditions: The payment confirmation e-mail has been sent and the client is in his/her personal "home page"

Exceptions: If the payment agency cannot process the payment the system sends the client a payment request via e-mail. The user is now blocked until payment.

5.2.15 Blocked client complete payment request

Name: Payment request

Actors: Blocked client

Entry conditions: The client has not completed the last payment and has been blocked

Flow of events:

- the client clicks on the payment link provided in the e-mail
- the client completes the payment request
- the system is informed by the Payment Agency and the payment is updated
- the system unlocks the client

Exit conditions: The pament has been registered and the client is active again

Exceptions: The payment fails, a new e-mail is sent to the client

5.2.16 Operator log in

Name: Log in

Actors: Operator

Entry conditions: The not logged operator must be on the "StartPage" in the website or the web app

Flow of events:

- the operator opens the operator website "StartPage"
- he/she fills in log in form, i.e. operator's code and password
- the operator clicks on "login" button
- The system redirects the operator to his/her personal page "MapPage"

Exit conditions: The operator is logged in

Exceptions: The e-mail or password typed are incorrect. The system advises the operator of the error and allows him/her to re-type.

5.2.17 Operator Logout

Name: Logout

Actors: Logged client

Entry conditions: The operator must be logged in

Flow of events:

- the operator clicks on "Logout" in the app menu

Exit conditions: The operator is in "Start Page".

Exceptions: No exceptions

5.2.18 Move car

Name: Move car

Actors: Logged operator

Entry conditions: Operator is logged and in the "MapPage"

Flow of events:

- in the "MapPage" list of movable car is shown on the map
- the operator select one car
- the system shows the operator the start position and the area where to bring the car
- the operator clicks on the "select" button
- the system sets the car status to "in use" until the operator has moved the car to the target Area
- the system sets the car status to "available" 'when the operator is done

Exit conditions: The car is moved to the target area

Exceptions: The car distribution is omogeneus and there are no car to be moved from an area to an other

5.2.19 Charge car

Name: Charge car

Actors: Logged operator

Entry conditions: Operator is logged and on the "MapPage"

Flow of events:

- in the "MapPage" list of movable car is shown on the map
- the operator select one car
- the system shows the operator the start position and the target station where to charge the car. There is no need to move the car if it is already in a charge station
- the operator clicks on the "select" button
- the system sets the car status to "in use"

Exit conditions: The car is moved to the target station and put in charge

Exceptions: There is no car to be charged, in this case the list of car is empty

5.2.20 Set car status

Name: Set car status

Actors: Logged operator

Entry conditions: Operator is logged and on the "MapPage"

Flow of events:

- the operator clikcs on "Change Car Status" button
- the list of all cars is shown on the map and the cars that need maintenance are hilghted so the operator can recognize them easily
- the operator select one car and clicks on "charge status" button
- the operator clicks on the new status
- the system sets the new status
- in case the new status is "regular maintenance" the maintenance request is automacally forwarded to the Maintenance Agency
- in case the new status is or "extraordinary maintenance" the operator has to contact manually the Maintenance Agency

Exit conditions: The new status has been set

Exceptions: No exception

5.2.21 Validate client

Name: Validate client

Actors: Logged operator

Entry conditions: Operator is logged and on the "MapPage"

Flow of events:

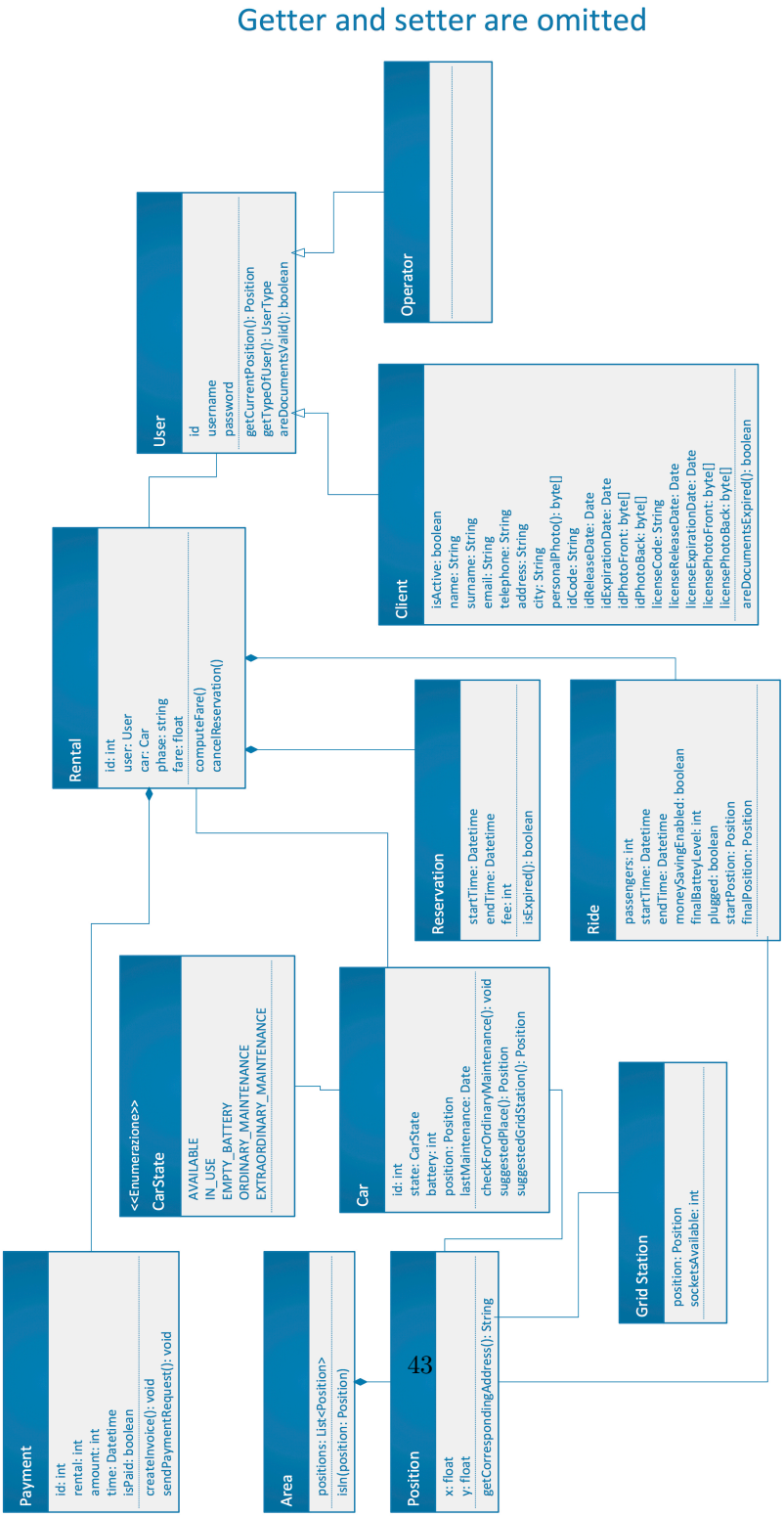
- the operator clikcs on "User" button
- the operator is redirected in the user section where the list of all awaiting clients is shown
- the operator select a client and visualizes all his data
- if the data are correct the operator cliks on "Validate" button
- in this case the system sends automatically an e-mail of confimation to the client
- if the data are not correct the operator cliks on "Don't validate" button
- in this case the system sends an e-mail to the client to alert him/her about the error. The email is completed by the operator with all the details about the error

Exit conditions: The client is validated/not validated and an e-mail of confimation/detail about the error has been sent

Exceptions: There are no client to validate, in this case the list is empty

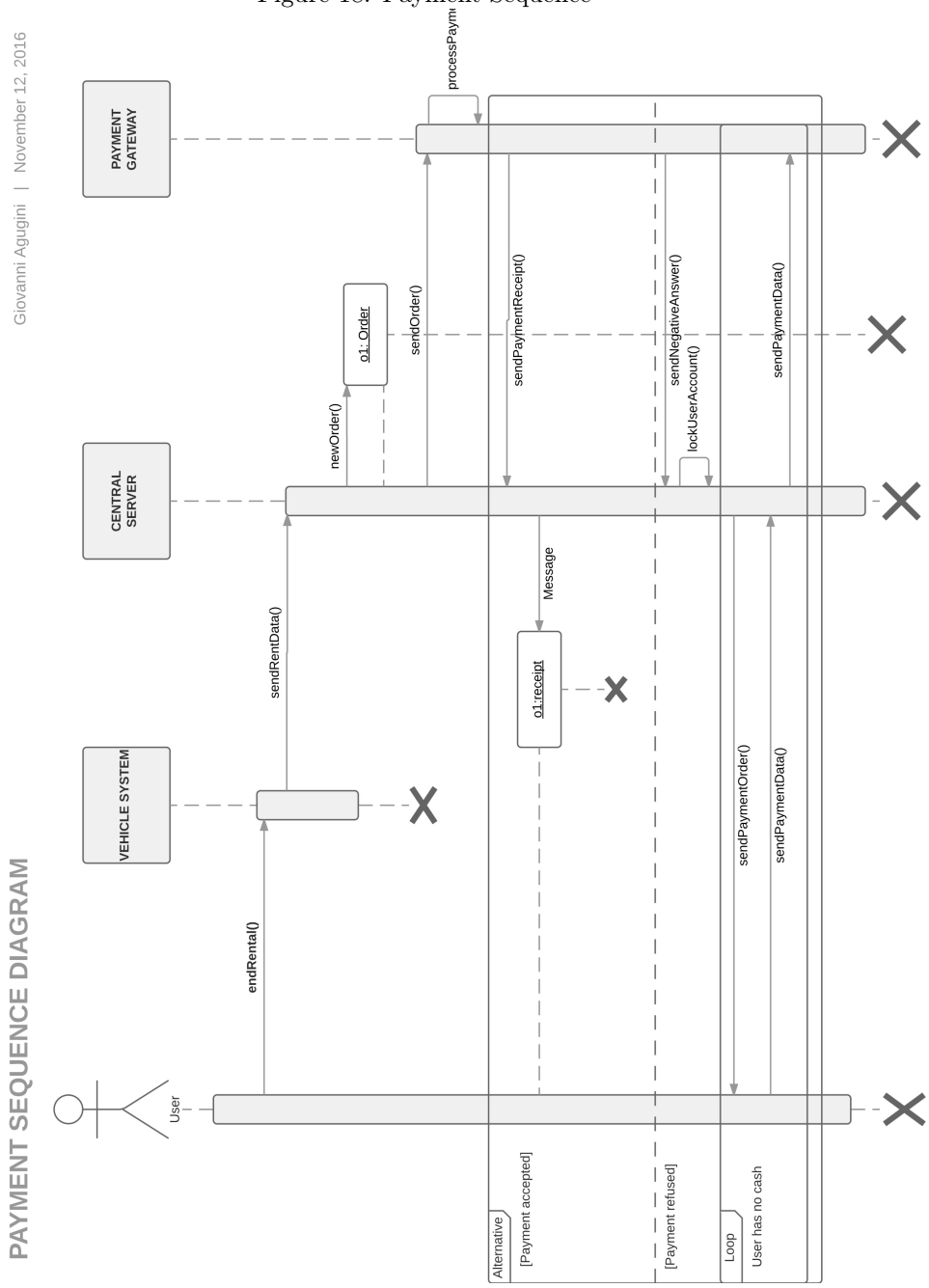
5.3 Class Diagram

Figure 17: Class Diagram



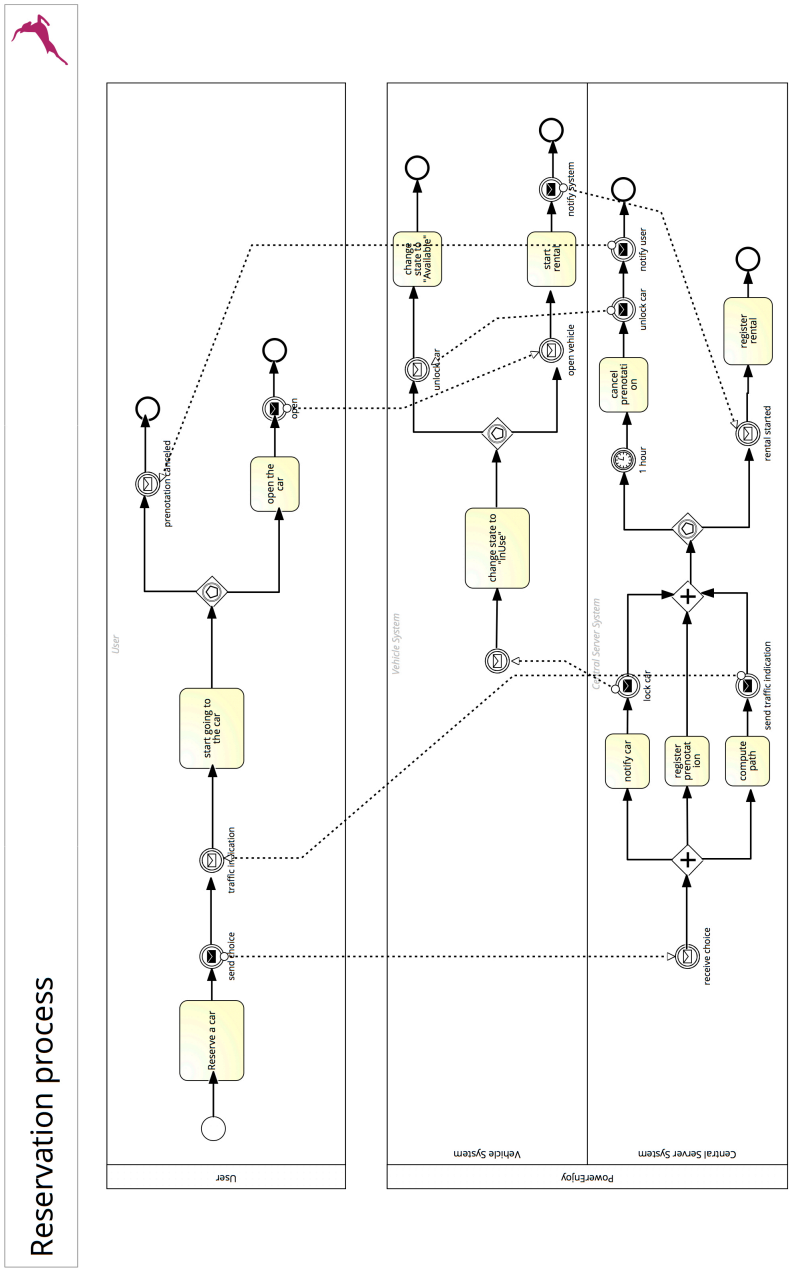
5.4 Sequence Diagram

Figure 18: Payment Sequence



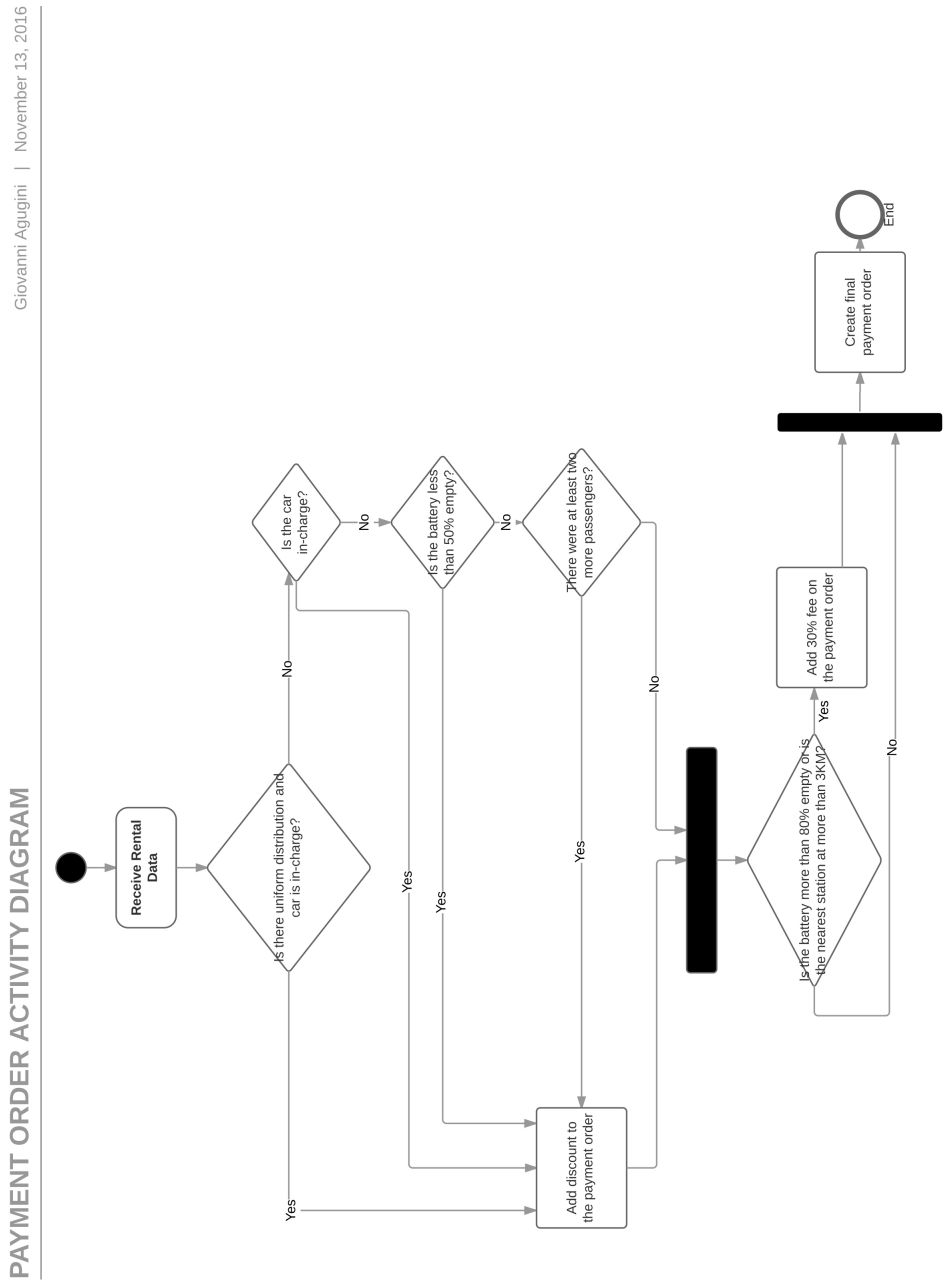
5.5 BPMN

Figure 19: Reservation Process



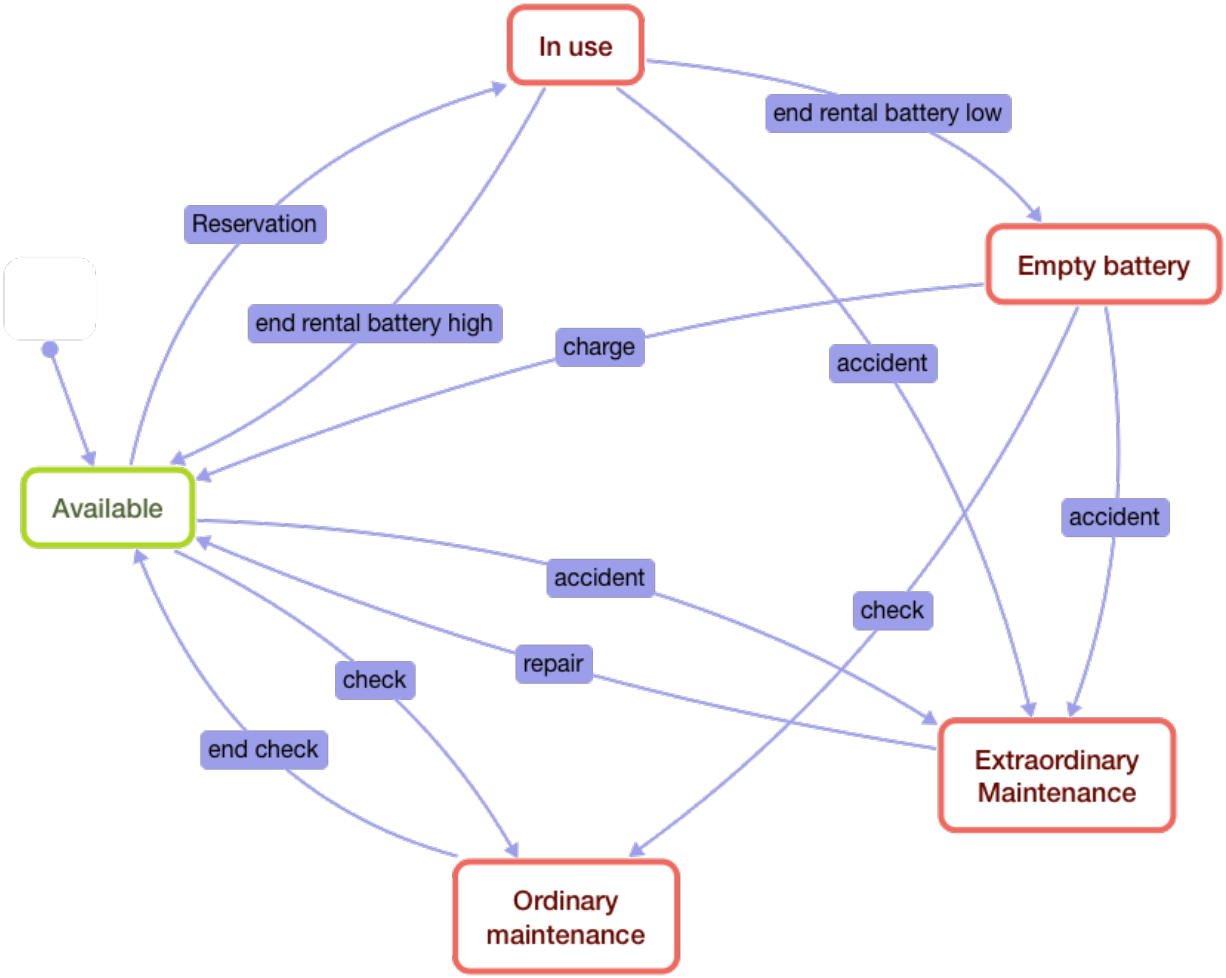
5.6 Activity Diagram

Figure 20: Payment Order



5.7 State Chart

Figure 21: State chart: car



5.8 Other diagrams

5.8.1 Traceability matrix

Table 1: Traceability matrix

Goal ID	Req ID	Use case ID	Comments
G1	1	5.2.3	
G2	2	5.2.4	
G2	2	5.2.5	
G3	3	5.2.7	
G7	5	5.2.8	
G8	8	5.2.14	
G9	9	5.2.13	
G10	10	5.2.14	
G11	11	5.2.15	
G12	12	5.2.16	

6 Alloy Modeling

6.1 Model

```
open util/integer
    //POWER ENJOY
    one sig PowerEnjoy {
    cars: set Car,
    rentals: set Rental,
    users: set User
    }
    {
    #cars >= 0
    #users >= 0
    #rentals >= 0
    }
    fact allInPowerEnjoy{
    all car: Car | car in PowerEnjoy.cars
    all user: User | user in PowerEnjoy.users
    all rental: Rental |rental in PowerEnjoy.rentals
    }
    abstract sig User {
    position: one Position,
    id: one Int
    }
    {
    id >= 0
    }
    fact differentUserId {
    no disjoint u1, u2: User | u1.id = u2.id
    }
    //OPERATORS
    sig Operator extends User {}
    //POSITIONS
    sig Position {
    positionX: one Int,
    positionY: one Int
    }
    abstract sig Area {
    positions: set Position
    }
    {
    #positions > 0
    }
    one sig SafeArea extends Area {}
    one sig UnsafeArea extends Area {}
```

```
fact disjointAreas {
all p: Position | {
p in SafeArea.positions or
p in UnsafeArea.positions
}
no p: Position | {
p in SafeArea.positions and
p in UnsafeArea.positions
}
}
fact noDuplicatedPositions {
no p1, p2: Position | {
p1 != p2 and
p1.positionX = p2.positionX and
p1.positionY = p2.positionY
}
}
//GRID STATIONS
sig GridStation {
position: one Position
}
fact allStationInSafeArea {
all g: GridStation | g.position in SafeArea.positions
}
//CARS
sig Car {
id: one Int,
state: one CarState,
position: one Position,
battery: one BatteryState
}
{
id >= 0
}
fact differentCarId{
no disjoint c1, c2: Car | c1.id = c2.id
}
fact differentPositionForCars {
no disjoint c1, c2: Car | c1.position = c2.position
}
abstract sig BatteryState {}
one sig BatteryLow extends BatteryState {}
one sig BatteryHigh extends BatteryState {}
abstract sig CarState {}
one sig Available extends CarState {}
one sig InUse extends CarState {}
```

```
one sig EmptyBattery extends CarState {}
one sig OrdinaryMaintenance extends CarState {}
one sig ExtraordinaryMaintenance extends CarState {}
fact batteryAndCarStateCoherence{
all c: Car | (c.state = EmptyBattery) => (c.battery = BatteryLow)
}
fact availableCarInSafeArea {
all c: Car | (c.state = Available) => (c.position in SafeArea.positions)
}
fact allCarsNotInSafeArea {
all c: Car | (c.position in UnsafeArea.positions) => (c.state != Available)
}
//CLIENTS
sig Client extends User {
state: one ClientState,
}
abstract sig ClientState {}
one sig ActiveClient extends ClientState {}
one sig InactiveClient extends ClientState {}
//RENTAL
abstract sig RentalPhase {
time: one Int
}
{
time >= 0
}
sig Reservation extends RentalPhase {
}
{
time <= 60
}
sig Ride extends RentalPhase {
passengers: one Passengers
}
sig Ended extends RentalPhase {
}
abstract sig Passengers {}
one sig NoPassengers extends Passengers {}
one sig EnoughForDiscount extends Passengers {}
one sig NotEnoughForDiscount extends Passengers {}
abstract sig Boolean {}
one sig True extends Boolean {}
one sig False extends Boolean {}
abstract sig Rental {
id: one Int,
user: one User,
```

```
car: one Car,
phase: one RentalPhase,
fare: one Boolean
}
{
id >= 0
}
fact differentIdForeachRental {
all pe: PowerEnjoy {
all disjoint r1, r2: Rental | ((r1 in pe.rentals) and (r2 in pe.rentals)) =>
(r1.id != r2.id)
}
}
fact differentCarUserForeachActiveRental {
all r1, r2: Rental |
(
(r1 != r2)
and
(r1.phase != Ended)
and
(r2.phase != Ended)
)
<=>
(r1.car != r2.car and r1.user != r2.user)
}
fact onlyClientsPay {
all r: Rental | (r.user in Operator) => (r.fare = False)
all r: Rental | (r.user in Client) => (r.fare = True)
}
fact allRentCarAreInUse {
all r: Rental | (r.phase != Ended) => r.car.state = InUse
}
fact inUseCarAreAllRent {
no c: Car | {
c.state = InUse
all r: Rental {
c != r.car
}
}
}
fact inactiveClientCannotRent {
all c: Client, r: Rental | (r.user = c and r.phase != Ended) => (c.state =
ActiveClient)
}
fact ifCarIsNotInSafeAreaRentalIsInRidePhase {
all r: Rental, c: Car |
```

```
(r.car = c and (r.car.position in UnsafeArea.positions) )
=>
(r.phase = Ride)
all c: Car |
(c.position in UnsafeArea.positions )
=>
(c.state = InUse or c.state = ExtraordinaryMaintenance )
}
fact userCarSamePositionOnRide {
all rental: Rental | (rental.phase = Ride and (rental.car.position in SafeArea.positions)
) => (rental.user.position = rental.car.position)
}
//ASSERT
assert noCarWithTwoActiveRental {
all c: Car, r: Rental |
(r.car = c and r.phase != Ended) => {
no r2: Rental | r != r2 and r2.phase != Ended and r2.car = c
}
}
check noCarWithTwoActiveRental for 10
assert noUserWithTwoActiveRental {
all c: Client, r: Rental |
(r.user= c and r.phase != Ended) => {
no r2: Rental | r != r2 and r2.phase != Ended and r2.user = c
}
}
check noUserWithTwoActiveRental for 10
pred show{
some Operator
some GridStation
some r: Rental | r.phase = Reservation
some r: Rental | r.phase = Ride
some r: Rental | r.phase = Ended
some c: Client | c.state = ActiveClient
some c: Client | c.state = InactiveClient
some c: Car | c.state = Available
some c: Car | c.state = InUse
some c: Car | c.state = EmptyBattery
some c: Car | c.state = OrdinaryMaintenance
some c: Car | c.state = ExtraordinaryMaintenance
}
run show for 25 but 8 Int
pred smallShow{
#GridStation = 1
#Operator = 1
#ActiveClient = 1
```

```
#InactiveClient = 1
#Car = 3
one r: Rental | r.phase = Reservation
one r: Rental | r.phase = Ride
one r: Rental | r.phase = Ended
#Position = 5
some Operator
}
run smallShow for 10 but 8 Int
```

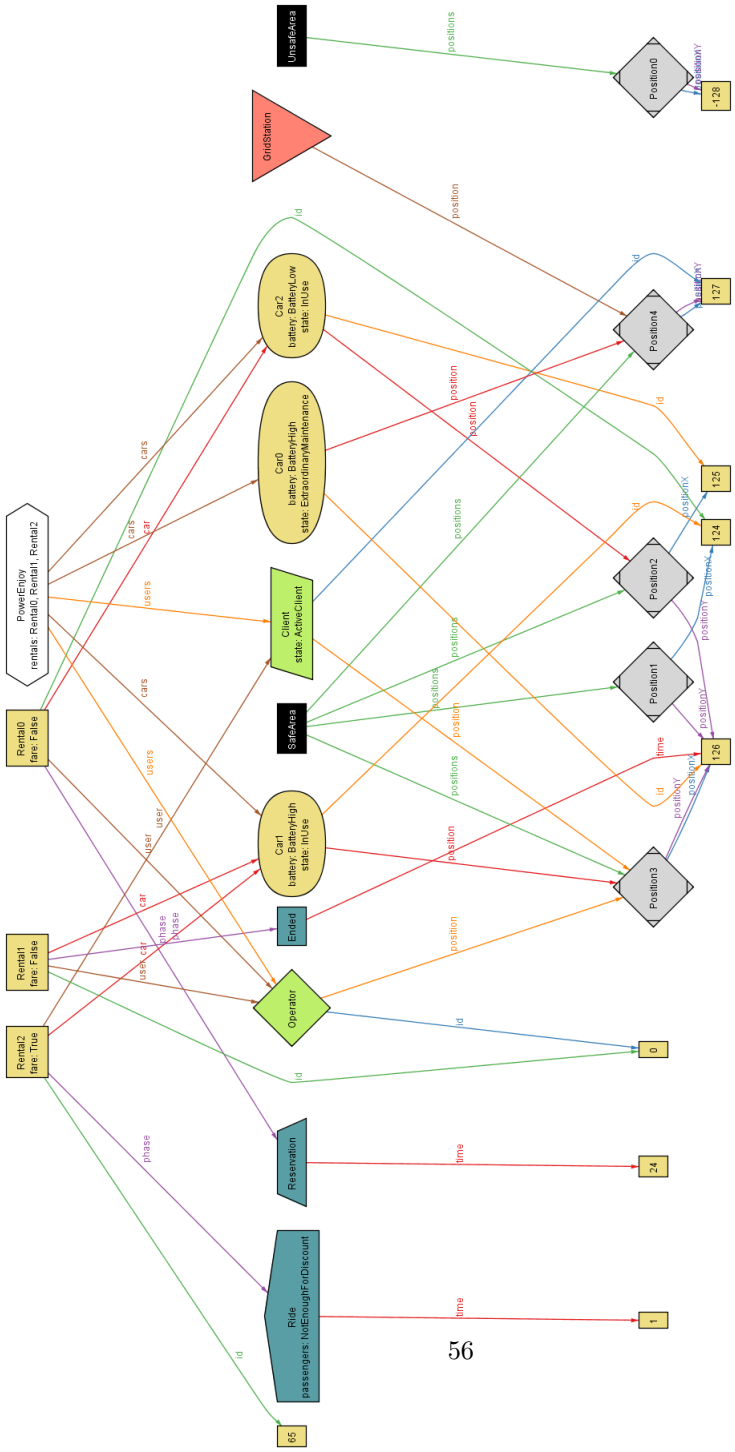
6.2 Results

Figure 22: alloy results

```
Alloy Analyzer 4.2 (build date: 2012-09-25 15:54 EDT)

Warning: JNI-based SAT solver does not work on this platform.
This is okay, since you can still use SAT4J as the solver.
For more information, please visit http://alloy.mit.edu/alloy4/

Executing "Check noCarWithTwoActiveRental for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  35273 vars. 1870 primary vars. 82462 clauses. 634ms.
  No counterexample found. Assertion may be valid. 400ms.

Executing "Check noUserWithTwoActiveRental for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  35273 vars. 1870 primary vars. 82462 clauses. 247ms.
  No counterexample found. Assertion may be valid. 273ms.

Executing "Run show for 25 but 8 int"
  Solver=sat4j Bitwidth=8 MaxSeq=25 SkolemDepth=1 Symmetry=20
  1481603 vars. 43100 primary vars. 3844329 clauses. 28707ms.
  Instance found. Predicate is consistent. 121031ms.

Executing "Run smallShow for 10 but 8 int"
  Solver=sat4j Bitwidth=8 MaxSeq=10 SkolemDepth=1 Symmetry=20
  381994 vars. 16240 primary vars. 1074273 clauses. 5471ms.
  Instance found. Predicate is consistent. 8704ms.

4 commands were executed. The results are:
  #1: No counterexample found. noCarWithTwoActiveRental may be valid.
  #2: No counterexample found. noUserWithTwoActiveRental may be valid.
  #3: Instance found. show is consistent.
  #4: Instance found. smallShow is consistent.
```

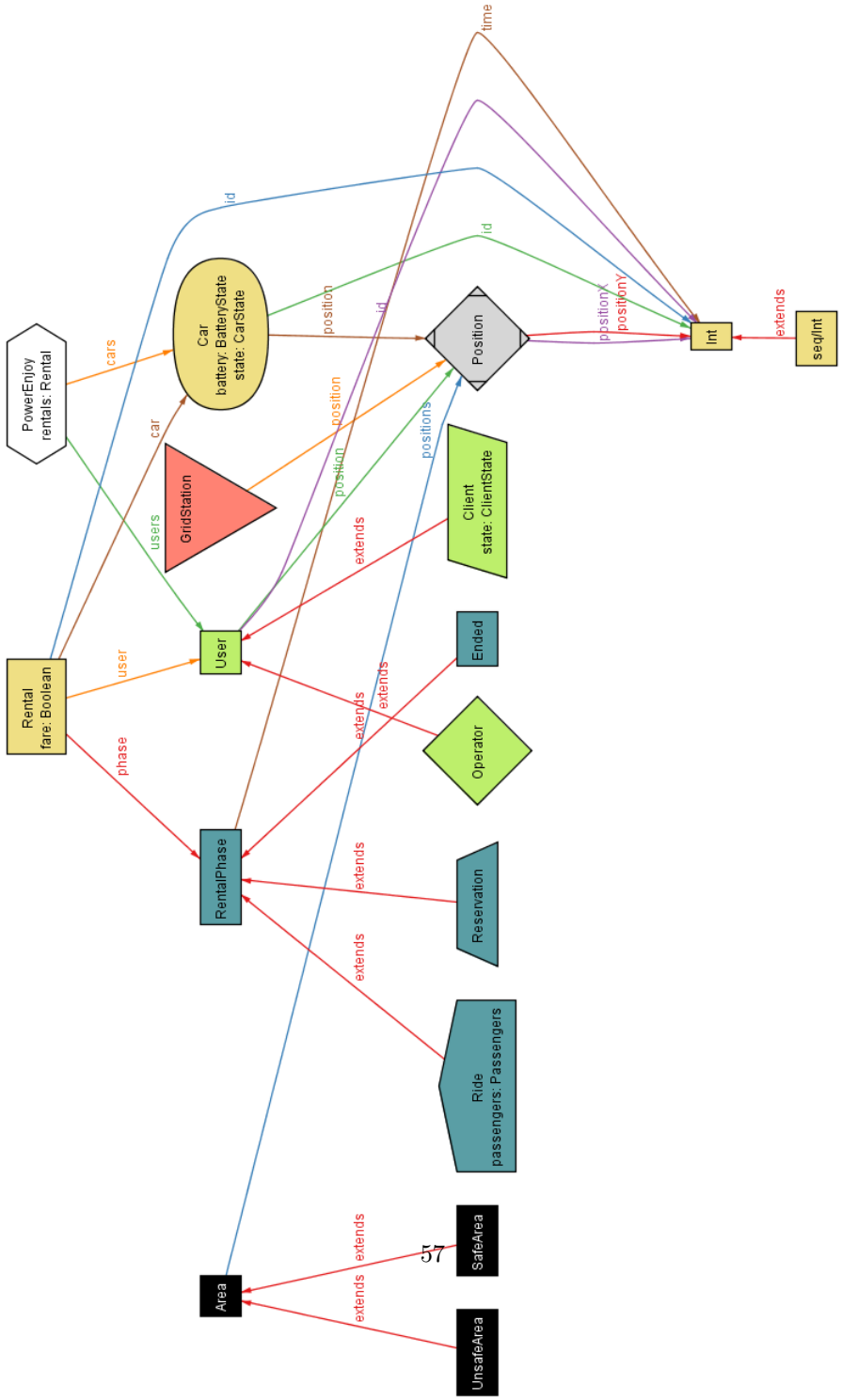
6.3 World generated

Figure 23: alloy results



6.4 Metamodel

Figure 24: alloy results



7 Hours of work

Giovanni Agugini

- 25/10: 2h
- 26/10: 2h
- 30/10: 5h
- 04/11: 5h
- 08/11: 3h
- 09/11: 4h
- 10/11: 4h
- 11/11: 4h
- 12/11: 4h
- 13/11: 7h

Matteo Foglio

- 25/10: 2h
- 26/10: 2h
- 30/10: 1:30h
- 03/11: 2h
- 04/11: 5h
- 5/11: 4h
- 07/11: 1:30h
- 08/11: 3h
- 09/11: 6h
- 10/11: 3h
- 11/11: 3h
- 12/11: 6h
- 13/11: 7h

Tommaso Massari

- 25/10: 2h
- 26/10: 2h
- 03/11: 3h
- 04/11: 2h
- 05/11: 3h
- 07/11: 1:30h
- 08/11: 1h
- 09/11: 5h
- 10/11: 3h
- 11/11: 5h
- 12/11: 4h
- 13/11: 7h