POLITECNICO DI MILANO

Department of Electronic, Information and Biomedical Engineering

Master Degree course in Computer Science Engineering

Project of Software Engineering 2

PowerEnJoy

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

(RASD)

Version 1.0 (13 November 2016)

Reference professor: Prof. Elisabetta Di Nitto

Authors:

Davide Anghileri   matr. 879194

Antonio Paladini   matr.

Academic Year 2016-2017

Contents

Revision History

Name	Date	Reason For Changes	Version
First complete version	13/09		1.0

Chapter 1

Introduction

In this chapter it will be displayed the main purpose of the project, a general and high-level description of the problem which will be detailed in subsequent steps of the documentation, goals and actors of the system and finally a glossary which explains how to interpret the terms that will be used from now on.

1.1. Purpose

The goal of this document is to represent the requirement analysis and specification document (RASD) with the support of most common analysis tools.

We are going to represent both functional and not-functional requirement via UML-diagrams, show the constrains and the limits of the system, simulate the typical use cases that will occur after the development and to give a high level specification of both application domain and the system to be developed.

In the last part of this document we will present a more accurate analysis using Alloy.

This document is intended to all developers and programmers who want to implement the requirements an could be used as a contractual basis between the customer and the developer.

1.2. Scope

Our main goal is to project and implement a digital management system for a car sharing service that exclusively employs electric cars.

Our system is called PowerEnJoy and is completely new, and developed with the aim of encouraging the use of electric cars instead of the normal ones.

The system we want to develop will have to allow to users to sign-up providing their credentials and payment information, to log-in, to reserve a car and finally to use it in respect of our rules.

1.3. Actual system

We suppose that until now nothing has been created and we have to create the entire application without using or modify a previous system.

1.4. Goals

PowerEnJoy has to provide these main features:

- G1: Allow users to enter in our cars and use them

- G2: The system gives information to the users about the position of available electric cars

- G3: Allow users to reserve an available car for up to one hour before they pick it up

- G4: Guests must be able to register and get a private access-password

- G5: allow users to be recognised by login procedure

- G6: The system allows users to delete a reservation they have already doneG8: keep trace of the past charges

- G7: The system keeps trace of the past charges

- G8: make users aware of position and conditions of parking areas (weather full or not)

- G9: System allows stops during a drive session

- G10: The system manages car accidents

- G11: The system ends travel sessions and charges the users

- G12: PowerEnJoy incentivizes the virtuous behaviours of the users

- G13: The maintainer takes care about physical assistance

1.5. Actors

- **Guest**: a person who is not registered to PowerEnJoy yet, he is only able to access to the login page, or to the registration page. The only action he can perform is the registration

- **User**: the one who use the application. He is able to sign up and then to log in whenever he wants. He is able to access the list of available cars and to reserve them. He accesses to a car by directly communication with the system, then he can use it, and finally park it in a safe area.

- **Maintainer**: a physical person who takes care about recover cars left in *not safe* places. He does the ordinary and not-ordinary maintenance of cars in case of malfunctions, and performs any other physical action needed. He uses a specific and private username and password to access to a dedicated section of our application and he interacts with system by communicating all the actions he performs.

- **System**: it has to manage the user's registration and log in system, to make available some data to the user, as the list

of accessible cars with their level of battery, and their geographical positions. Moreover, it takes care about the cars management, checking their level of battery, charging them, making them available to the users. It opens a car when there is a nearby user who has reserved it. It charges the user for the using time and check information about the travelling through a mobile application which is present in each car.

1.6. Definitions, Acronyms, Abbreviation

1.6.1. Definitions (STATE)

- PowerEnJoy: we use it to refer to the application.
    - o Synonymous: application, system
- Guest: he (or she) is a visitor of the application who is not registered yet. Synonymous: new user
- *User:* a person who use the application and is correctly registered.
    - o *Synonymous:* customer
- *Maintainer:* a person who takes care about each physical action which is needed for the system operation.
- *Logged User:* a user who is currently logged and is performing some action in the application.
- *Renter:* a *logged user* who has reserved.
- *Driver:* a *renter* who started using the car he has reserved and has not parked it yet.
- *Bad Action:* an action performed by a user which is not conform with the system rules

- *Out-law User:* a user who has committed more than 5 bad actions.

- *Account:* the personal area of a user. It contains all his (or her) personal data.

- *Busy Car:* a car which is not available for reserving

- *Bad Car :* a *busy car* which is not available because of some mechanical or technical problem.

- *Closest Car:* the closest car to the location chosen by a user

- *Sensor:* every object on the cars which gives information about a trip to the system

- *Parking area:* we use this term to indicate a safe area where is possible to park a car. Synonymous: normal parking area, safe parking area.

- *Special Parking area:* we use this term to indicate a parking area where is possible to charge cars. Synonymous: power grid station.

- *Drive session:* period of time in which a car is used by a user. It starts in the moment he picks up the car and ends when he lives the car in a safe parking area, or in one of the other not ordinary cases.

- *Charging station:* the physical charger of the cars. In a special parking area there are a lot of charging station.

- *To Repair:* with this world we intend all the physical action that the maintainer has to do in order to return a car available such as recharge the car, take it to a parking area or adjust a fault.

1.6.2. Acronyms

- RASD: Requirements Analysis and Specification Document.
- IEEE: Institute of Electrical and Electronic Engineers

1.6.3. Abbreviations

- min(a,b)= a if and only if a<=b and b if and only if b<=a
- Km= kilometres
- DB=database

1.7. Reference Documents

- Assignments AA 2016-2017
- International Standard ISO/IEC/IEEE 29148
  First edition 2011-12-01
- RASD_meteocal-example1.pdf
- RASD_meteocal-example2.pdf

1.8. Document Overview

This document is divided into five different sections so organised:

- *Introduction*: here is given a first general description of the system, with its main features, goals and actors.
- *Overall Description*: gives some more specific details over the users and the software specification, focusing on its constraints and our assumptions.
- *Specific Requirements*: contains the description of the scenarios, the use cases that describe them, and the models describing requirements and specification for the problem under consideration.
- *Alloy*: containing a consistency test of our model made using Alloy Analyzer.

- *Other Information*: gives some imformations about the tools that we used for write this document, the working hours and a possible future development for our application.

Chapter 2

Overall Description

2.1. Product Perspective

The proposed solution is an independent mobile application which is not integrated with a previous system. The user will be able to access via internet using any portable device. The application will not have any internal interface for administration but it will be only user based.

2.2. User Characteristics

The application has two different kind of users:

- User: as described above, he (or she) is a user which interact with the application in order to use our car sharing service; the user has to register to provide basic information about him (or her) like personal data, driver license and payment methods. The user must be able to use a simple portable device and download our mobile application, and he (or she) needs internet access and GPS.

- Maintainer: is a specific person who interact with the system in order to carry out physical tasks.

2.3. Constraints

2.3.1. Regulatory policies

A user must be entitled to drive in compliance with the traffic rules of his country.

2.4.2. Hardware limitations

PowerEnJoy has to work on a device with a working internet connection and GPS. Moreover, the system relies on a database.

2.4.3. Interfaces to other applications

PowerEnJoy has to interface with the mobile devices placed on each car.
All payment system must be able to be dynamically invoked by other systems relying on it.

2.4.4. Parallel Operation

PowerEnJoy must support parallel operation from different users when working with database and for each operation done after the connection of users.

2.5. Assumptions and dependencies

There isn't any dependence between users.

Here is a list of the main domain assumptions, successively we will link each of these with the goal related.

1. The cars are integrated with a mobile application that allows to send and receive signals from the system such as the number of passengers on board, the position through a GPS device and the car battery status.

2. There is a particular figure called *maintainer* who performs all the physical actions needed from the system. The system keeps track about each action is performed by a personalized interface for this kind of users.

3. The position of a logged user is known with a precision of 5 meters by GPS.

4. The system is able to unlock a specific car by internet connection.

5. Cars can be turned on and off through a start&stop button on board.

6. The cars always show the battery level and the estimated autonomy.

7. There is a screen on each car.

8. Position and battery level of cars are known thanks to the application on board.

9. Only renter of a car can drive it.

10. The set of safe areas for parking cars is pre-defined and well known by the system.

11. If an accident takes place the driver contacts the support and advises about the fact.

12. The application on board of each car Is able to understand how many passengers are into the car.

13. Maintainers are able to move cars from a place to another.

14. Maintainers are able to repair the ordinary damages.

2.5.1 Assumptions over requirements

We have considered the following assumptions which specify or extend the ones required in the assignment:

1. A user can't register without providing a valid method of payment.

2. It is possible to register only one account per person. This is because we specify some penalties for those who use the application but doesn't care about its rules. In this way, if a person for example is banned, he is not able to create a new account with the same personal details.

3. A user can reserve only a car at a time.

4. Older users pay less than the younger ones, as results from the formula:

   $[0,3-\min(n,10)*0.01]€$

   where "n" is the number of year passed from when the user has registered.

5. If a renter gets into a car and doesn't turn it on for more than 5 minutes the system does it automatically (as we would like to avoid people who keep cars busy without paying for their use).

6. A user can temporary park a car in a place not labelled as "*safe parking area*" for up to 30 minutes at a time. In other words, he can do small stops, during which he continues to pay as he was driving.

2.6. Considerations about the system

We want to add some consideration about our application because they represent our vision of the application.

So we think PowerEnJoy has to be:

- Easy to use: we really want that almost anyone who own a mobile device is able to use it.

- Stable: system faults have to be as rare as possible.

- Robust: in case of faults the system shouldn't change too much its behaviour

Chapter 3

Specific Requirements

3.1. External Interfaces

3.1.1. User Interfaces

Here we represent the structure of our application as it compares in some different pages.
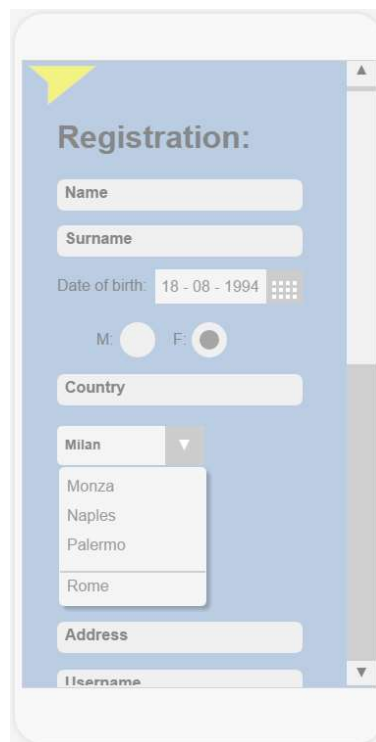
3.1.1.1. Login

This is the page shown when the application is launched. Here users can log in into the application and guests can access to the registration form.

3.1.1.2. Registration Form

This page allows to guests to register a new account by providing their personal details as shown in the picture below.

3.1.1.3.   Home Page

This mock shows the home page of our application where a logged user can access to the following pages: "Reserve a Car", "Personal Area", "Cancel Reservation", "Past bookings" and "Support".
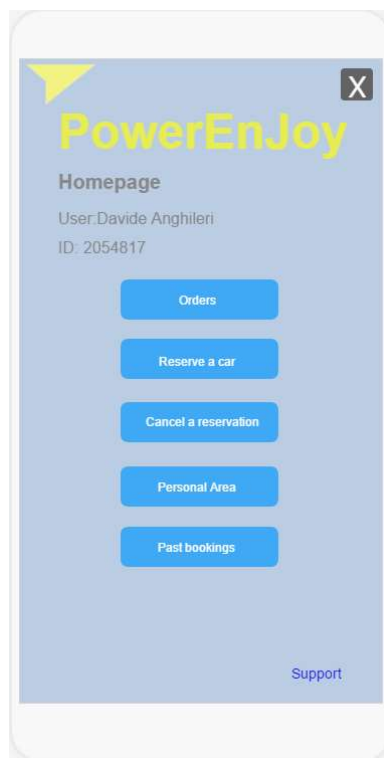
3.1.1.4.   Reserve a Car

In this page a logged user can specify a new position or select one from his chronology and then he can select a car from a map where are shown all the available cars around his (or her) position.

He can also go back to the Homepage or use the navigation bar to go through "Personal Area" and "Support" pages.
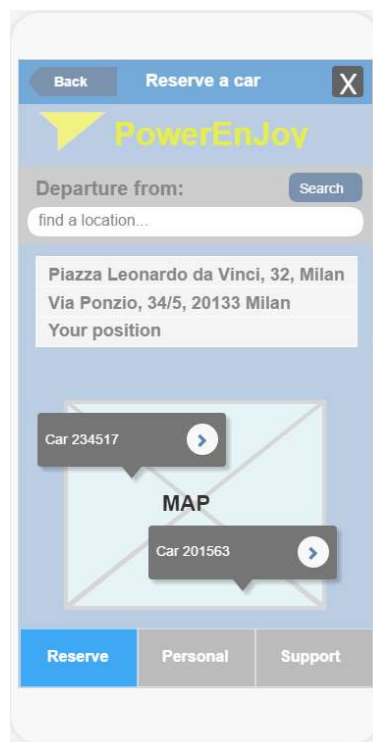
3.1.1.5. Confirm Reservation

This page is shown after the user has chosen a specific car and is used to confirm the reservation.

The user can also go back to "Reserve a car" page and select a different car.

3.1.1.6.   Cancel Reservation

Through this page a logged user can select a reservation that is still active and cancel it.
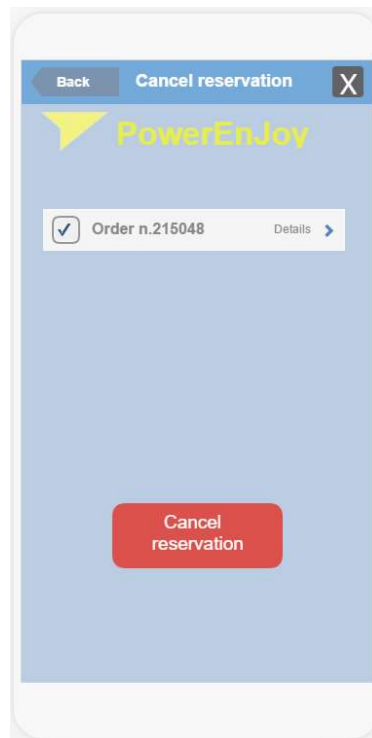
3.1.1.7.   Personal Area
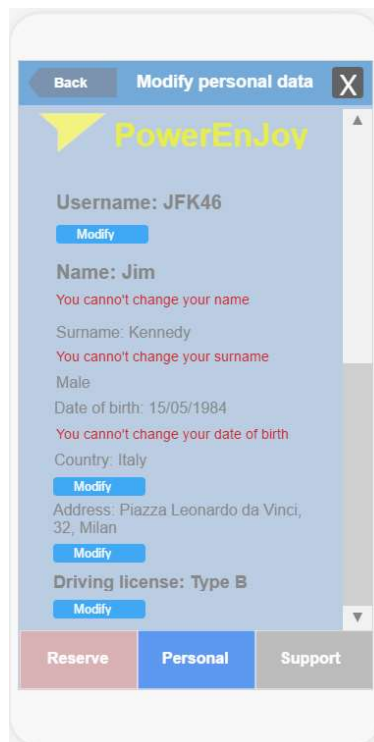
In this page a logged user can see all his personal data, and access to the modification page.

He can also go back to the Homepage or use the navigation bar to go through "Reserve a car" and "Support" pages.

3.1.1.8.   Modify Personal Data

Here a user can modify some of his (or hers) personal data. He can also go back to the "Personal data" page or use the navigation bar to go through "Reserve a car" and "Support" pages.

3.1.1.9. Past Bookings

This mock shows information and statistics about a user and his (or hers) past reservations.
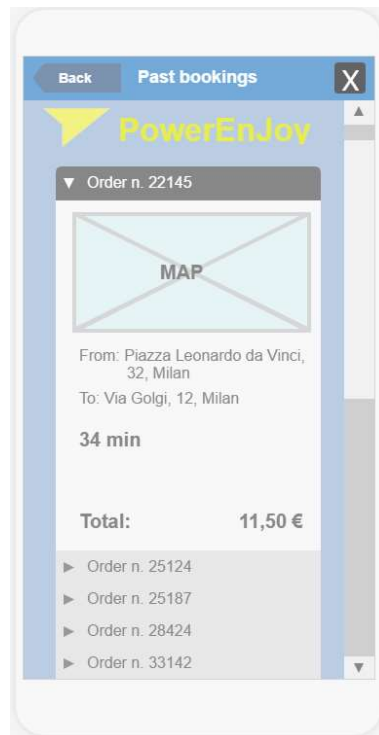
3.1.1.10. Support

In this page a user can see all the information that he need to contact us and ask for support including our email and telephone number.
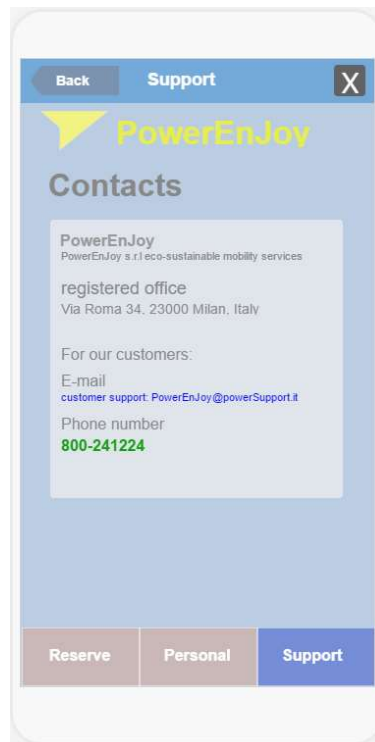
3.1.2. Maintainer Interfaces

3.1.2.1. Login

This page is the same as the user log-in page.

3.1.2.2. List of requested assistance

In this mock a maintainer can see a list of all the request of physical support and all the information that he need to solve the problem.
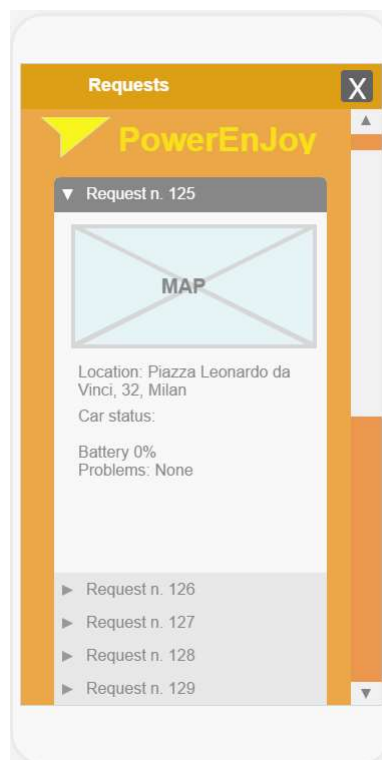
3.1.3. API Interfaces

For the maps in our application we use the Google Maps Android API and Google Maps Directions API in order to create and customize the map and calculate directions between locations; for more information, see the Google Website (https://developers.google.com/maps/documentation/android-api/)

3.1.4. Hardware Interfaces

Both for users and maintainers:

- Mobile device:
    - 3G connection (at least)
    - GPS

3.1.5. Software Interfaces

- Database Management System (DBMS):
    - Name: MySQL
    - Version: MySQL Standard Edition 5.7
    - Website:http://www.mysql.com/products/standard/
- Android Run Time (ART):
    - Name: ART
    - Website:
        https://source.android.com/devices/tech/dalvik/index.html
- Operating System (OS):
    - Name: Android
    - Version: 5.0 Lollipop or gather
    - Website: https://www.android.com/intl/it_it/

3.1.6. Memory

The minimum memory requirement is 50 MB.

3.2. Functions

3.2.1. G1: allow users to enter in our cars and use them

- [D1]: The position of a logged user is known with a precision of 5 meters by GPS.
- [D2]: The system is able to unlock a specific car by internet connection.
- [D3]: Cars can be turned on and off through a start&stop button on board.
- [D4]: The cars always show the battery level and an estimated autonomy.
- [D5]: There is a screen on each car.
- [R1]: If and only if a renter is closer than 10 meters to a car he (or she) has reserved, he (or she) can ask to the system to open it.
- [R2]: If a renter has asked to unlock a car, the system does.
- [R3]: When a car is unlocked by the system it saves the exact time.
- [R4]: As soon as the engine ignites, the system starts charging the driver for an amount of money per minute calculated with the following formula:

$$[0,3-min(n,10)*0.01]€$$

  where "n" is the number of year passed from when the user has registered.
- [R5]: If a renter does not turn on the engine within 10 minutes from when he (or she) entered into the car, the

system starts charging the renter as if he (or she) has done it

- [R6] The system notifies drivers of the current charges through the screens on the cars.

- [R7]: The system keeps track of the route of each trip.

3.2.2. *G2:* The system gives information to the users about the position of available electric cars

- [D1] Position and battery level of cars are known thanks to the application on board.

- [R1] If a car isn't located in a safe parking area it is not available.

- [R2] If a car has more than 40% of battery empty it is not available.

- [R3] Cars which are currently reserved are not available

- [R4] Cars which are currently in use are not available.

- [R5] The system allows logged users to choose a specific position around which search for available cars.

- [R6] The system makes available a map of the available cars to logged users.

- [R7] The system shows in the same map also cars with a current level of battery lower than 60% and that are on charge.

- [R8] For cars which are on charge but with a battery level lower than 60% system shows the remaining time needed to become available.

3.2.3. *G3:* Allow users to reserve an available car for up to one hour before they pick it up.

- [D1] Only renter of a car can drive it.
- [R1] Only a logged user can access to details of an available car.
- [R2] Only a logged user can reserve any available car.
- [R3] A logged user has to choose one of the payment methods he (or she) has linked to his (or her) account before to proceed with a reservation.
- [R4] The system keeps track of all the active reservation, saving from which user they were done and at what time they were made.
- [R5] A user can only reserve a car at a time.
- [R6] If a car is not picked-up within one hour from the reservation the reservation expires.
- [R7] If a car is not picked-up within one hour from the reservation the user pays a fee of 1€.

3.2.4. *G4:* Guests must be able to register and get a private access-password

- [R1] The system has an email address which is used for automatic communications.
- [R2] Guests are only allowed to access to log-in or registration page.
- [R3] It is possible to register only one account per person.
- [R4] Guests must choose a username not already used by another user.
- [R5] Guests have to provide a correct e-mail address in order to register.

- [R6] Guests must provide all their personal data, that is: name, surname, date, gender, date of birth, residential address.

- [R7] Guests must provide both a copy of ID Card and Driving License.

- [R8] The system checks whether information on the documents match the ones the guest has provided in the registration form.

- [R9] The system checks whether the documents are valid.

- [R10] Guests must provide at least one method of payment at the moment of the registration.

- [R11] If all the fields of the registration form are full the guest can choose to submit them.

- [R12] If the system detects something wrong in the documents and payment methods provided by guests it communicates it by the e-mail address provided and cancel the registration procedure.

- [R13] Once the guest has submitted the registration details the system provides an e-mail which contains a private access-password and a button for activation.

- [R14] Once a guests clicks for the activation he (or she) becomes a user and is able to login with his (or her) credentials.

3.2.5. G5: allow users to be recognised by login procedure

- [R1] The login can be performed only by users who provide the correct credentials.
- [R2] A user has to provide both username and password he (or she) registered with in order to login.

- [R3] The system allows users to request for a new password if they don't remember theirs. The new password is provided by email.

- [R4] The system doesn't ask to do the login procedure if the user closed the application less than 30 minutes before.

3.2.6. G6: The system allows users to delete a reservation they have already done

- [R1] A logged user can check whether there is already an active reservation on his (or her) behalf.

- [R2] The system allows to delete a reservation if and only if it is more recent than 30 minutes.

- [R3] If a user cancels a reservation he (or she) doesn't have to pay any fee.

3.2.7. G7: The system keeps trace of the past charges

- [R1] For each user are saved in the database all the trip he (or she) has done with the associated details (route, duration, price).

- [R2] A logged user can access to the list of past charges at any time.

3.2.8. G8: make users aware of position and conditions of parking areas (weather full or not)

- [D1] The set of safe areas for parking cars is pre-defined and well known by the system.

- [R1] System shows through the screens on board a map where are visible all the nearest parking areas.

- [R2] For each parking area the system shows how many places are free.

- [R3] For each parking area the system specifies whether it is a normal parking area or a special parking area.

3.2.9. G9: System allows stops during a drive session

- [R1] A driver can lock the car by asking it to the system.

- [R2] If a driver gets out of the car and doesn't come back for more than 2 minutes, the system automatically locks the car.

- [R3] If a driver gets out of a car and doesn't come back within 30 minutes the reservation expires.

- [R4] If a driver gets out of a car and doesn't come back within 30 minutes the system requires the intervention of a maintainer to move the car in a parking area.

- [R5] If a driver leaves a car in a place not signed as safe parking area and doesn't come back within 30 minutes he (or she) commits a bad action.

- [R6] If a user commits the first bad action his (or her) account is banned for a month.

- [R7] If a user commits two bad actions his (or her) account is banned without terms.

- [R8] A user can unlock his (or her) account by paying a fee of 100€.

3.2.10. G10: The system manages car accidents

- [D1] If an accident takes place the driver contacts the support and advises about the fact.

- [R1] When an accident takes place the drive session ends.

- [R2] Users must respond of damages they do by paying the repair.

- [R3] If an accident is notified by a user, the system requires the intervention of a maintainer in order to get it repaired

- [R4] After a maintainer has repaired a car he (or she) takes careof movingit in a safe parking area.

3.2.11. G11: The system ends travel sessions and charges the users

- [R1] When a user stops a car in one of the safe parking areas and get out of the car the drive session ends.

- [R2] When a drive session ends the system locks the car automatically.

- [R3] When a drive session ends the application shows to the user the route taken and the total charge.

- [R4] The total charge is automatically taken from the payment method chosen by the user at the moment of the reservation.

3.2.12. G12: PowerEnJoy incentivizes the virtuous behaviours of the users.

- [D1] The application on board of each car Is able to understand how many passengers are onto the car.

- [R1] If the system detects the user took at least two other passengers into the car, the system applies a discount of 10% on the ride.

- [R2] If a car is left with no more than 50% of battery empty the system applies a discount of 20% on the last ride.

- [R3] If a car is left at special parking areas where they can be recharged and the user takes care of plugging the car into the power grid, the system applies a discount of 30% on the last ride.

- [R4] If a car Is left at more than 3KM from the nearest power grid station or with more than 80% of the battery empty, the system charges 30% more on the last ride.

3.2.13. G13: The maintainer takes care about physical assistance

- [D1] Maintainers are able to move cars from a place to another.
- [D2] Maintainers are able to repair the ordinary damages.
- [R1] A maintainer can login with personal credentials provided by the system.
- [R2] All maintainers can see all the requests of assistance in chronological order.
- [R3] If a maintainer is not able to perform a repair he may decide to replace a car.

3.3. Matching table

Goals, requirements, domain assumptions, use cases.

Goal	Requirement	Domain Assumptions	Use Cases
G1	R1, R2, R3, R4, R5, R6, R7	D1, D2, D3, D4, D5	UC-3
G2	R1, R2, R3, R4, R5, R6, R7, R8	D6	UC-5
G3	R1, R2, R3, R4, R5, R6, R7	D7	UC-4
G4	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14	no assumptions	UC-1
G5	R1, R2, R3, R4	no assumptions	UC-2, UC-10
G6	R1, R2, R3	no assumptions	UC-9
G7	R1, R2	no assumptions	UC-6
G8	R1, R2, R3	D8	
G9	R1, R2, R3, R4, R5, R6, R7, R8	no assumptions	
G10	R1, R2, R3, R4	D9	
G11	R1, R2, R3, R4	no assumptions	UC-12
G12	R1, R2, R3, R4	D10	
G13	R1, R2, R3	D11, D12	UC-14, UC-13

3.4. The world and the machine

For a better understanding about the phenomena and requirements we decide to use "The world & the machine" model by M. Jackson & P. Zave. This approach let us to identify the portion of the system to be developed (the machine), the portion of the real-world affected by the machine (the world) and all the shared phenomena that can be controlled by the world and observed by the machine or vice versa controlled by the machine and observed by the world (e.g. when a user is near the car and taps to unlock it on the application the system unlock the car).
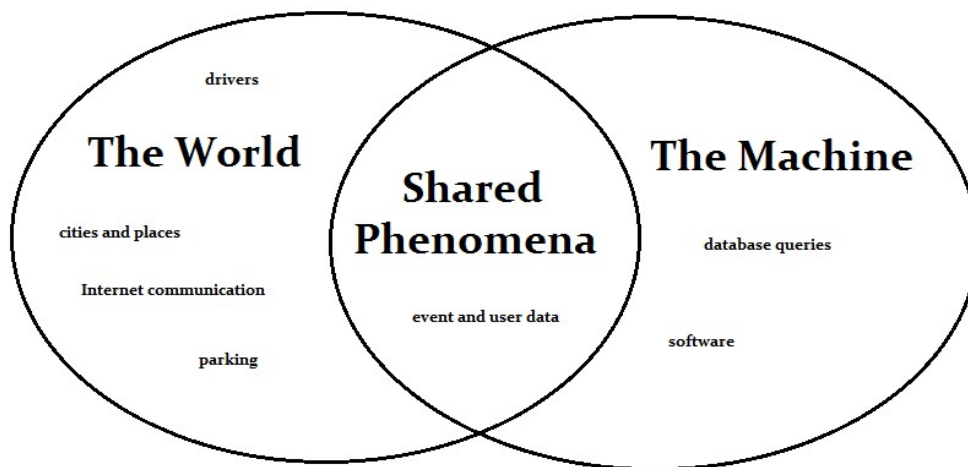
3.5. Scenarios

Scenario 1

Tom has started working in the city one week ago, and is looking for a good car-sharing service. He has heard about PowerEnJoy and as he is a convinced environmentalist decide to download the application and install it on his device. The first page he sees is the login page. He taps on *"register now"* button in order to register. Now the registration page is shown. Tom fills all the field required and submit the request of registration. Katia, who is a PowerEnJoy employee checks his data and finds something strange.

After ten minutes Tom receives an email from PowerEnJoy which says that his ID Card has expired and invites him to repeat the procedure. Tom have uploaded an old scanning of his card, so he does a scanning of his new ID Card and fill all the registration form for a second time. After a while an email is sent to Tom, it contains both a link for the activation of the account and a private password. Tom taps on the activation link, the system activates his account and he becomes a user.

Scenario 2

Tom has reserved his first car. He is getting near the car he has reserved, he takes his mobile phone and starts the application which shows automatically the homepage. From here Tom taps on the *"reserve a car"* option, which shows him a button *"unlock"*. The button is not active yet, Tom has to go really close to the car. After a few steps Tom realizes the button became active, then he taps it, and the car is unlocked.

Now Tom enters into the car, and pushes the "*start&stop*" button (which is on board), the car is set in motion. Tom drives for 20 minutes till his office, on the screen on board he sees a map, the battery level and the total amount charged for the trip updated in real time.

Scenario 3

Once arrived to his office Tom checks on the map for the nearest parking area. Tapping on it he realizes it is not a special parking area, so he tries with the second nearest which is a special parking area. Tom decide to go there in order to obtain the discount on the final charge. Once he arrives he parks the car and get out. The system detects the car is in a safe parking area and locks it.

Tom plugs-in the car in the charging station, then he checks on the application for the total amount to be paid; he has obtained the discount for his good behaviour.

Scenario 4

Today Tom has reserved a car for a long trip; he has to go to his parent's house which is across town. He will bring with him his wife and his two children. Before to go there he needs to stop at the supermarket to buy some stuff. Once got into the car Tom drives for a while till he gets to the supermarket, when he stops the charge counter is arrived to 9€, as he drove for 30 minutes. When the whole family come out of the car it is locked automatically by the system. They spend 20 minutes in doing shopping and then come back, require the system to unlock the car and get into it.

Once Tom turn on the engine the charge counter shows an amount of 15€.

Scenario 5

Once they arrive at destination they search for the nearest parking area. The nearest is really closed to Tom's parent's house but 4 Km far from the nearest power grid station. They decide to park there because it's very late. Once they get down from the car, the system detects they are in a safe parking area, so locks the car and cancel the reservation. Tom checks on his smartphone the total amount to be paid. He has used the car for one hour and a half, and battery is 70% empty.

The total is of 27€, he has obtained a discount of 10% because the system has detected more than one other passenger, but he gets also to pay 30% because the car has been left at more than 3 KM from the nearest power grid station, so he will pay a total charge of 32.4€.

3.6.   UML Models

3.6.1. Use case
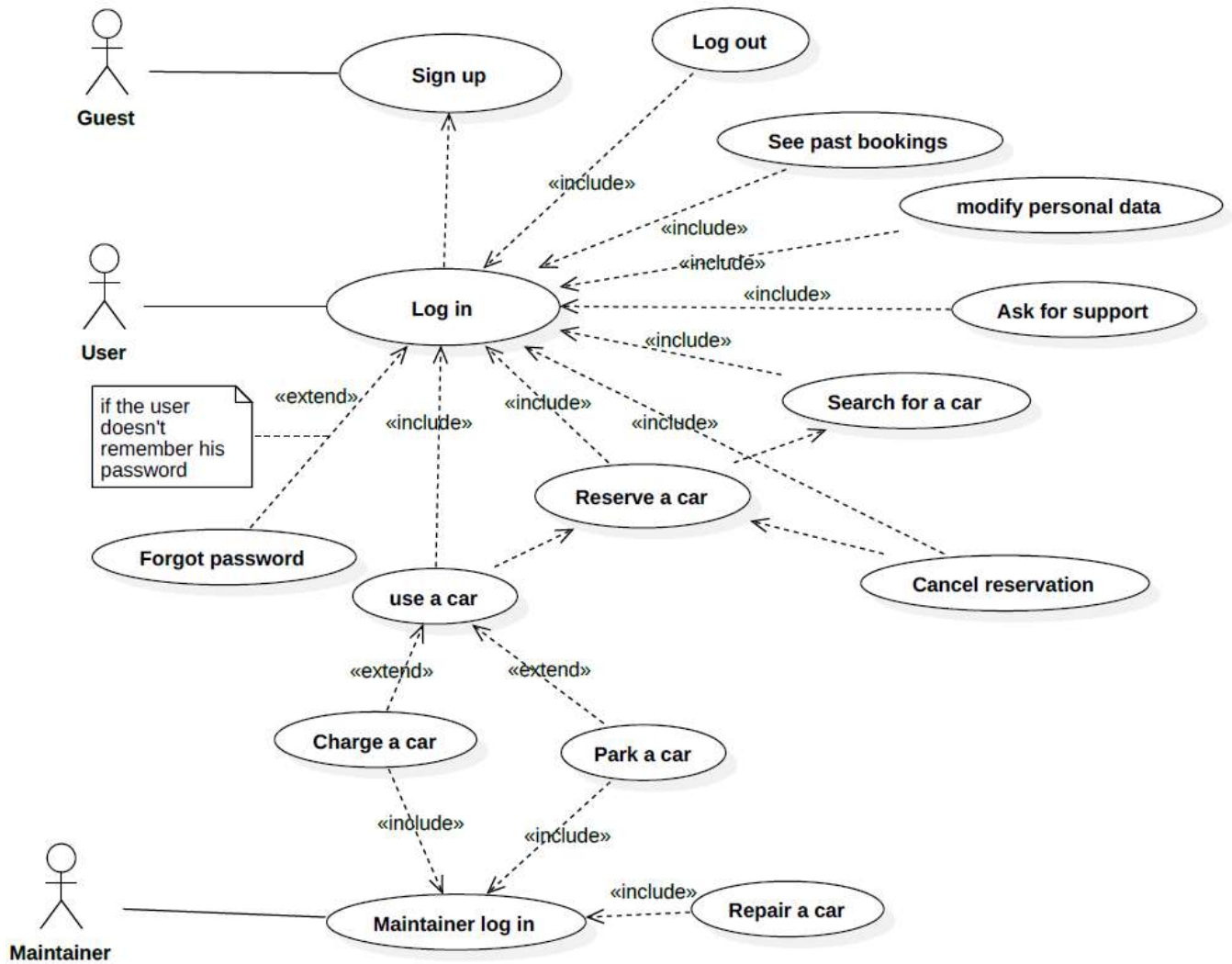
Now we describe in a detailed way the use cases that we derived from the scenarios and the use case diagram.

It is important to understand that all the references to "pages", "buttons" or "input forms" are only hypothesis to make the situation as clear as possible and to help the reader to draw a visual picture in his mind of what we plan to do, but the real structures will be well defined in the Design Document.

The use cases are the following:

1. Sign up
2. Log in
3. Use a car
4. Reserve a car
5. Search for a car
6. See past bookings
7. Modify personal data
8. Ask for support
9. Cancel reservation
10. Forgot password
11. Charge a car
12. Park a car
13. Repair a car
14. Maintainer log in
15. Log out

ID:	UC-1
Title:	Sign up
Description:	A guest (new user) of our application registered into it
Primary Actor:	Guest
Preconditions:	The guest has never completed the PowerEnJoy registration procedure before.
Postconditions:	The guest is registered and became a user
Main Success Scenario:	1. The guest starts the application 2. The guest taps the "register now" button 3. The system requires to insert username, email, name, surname, gender, date of birth, country, city, address, postal code, type of driving license, an image of the Identity card and one of the driving license 4. The system verifies the unicity of the email address and username 5. The system verifies the validity of the fields 6. The system confirms the registration 7. The system sends a confirmation email with the password to the guest 8. The guest taps on the confirmation link 9. The system notifies the registration by email
Extensions:	4. If the email address or the username are already in the DB the system asks to the guest to re-insert them. 5. If the fields are not coherent the system sends

	an email to the guest and asks to redo the registration procedure. 8. If the guest doesn't tap on the confirmation link within 24 hours the registration is cancelled 9. If the guest doesn't receive the confirmation email can tap to the "resend" button and receive a new one confirmation email
Frequency of Use:	Few times in a day
Priority:	P2 - high

ID:	UC-2
Title:	Log-in
Description:	A user log-in into the application
Primary Actor:	User
Preconditions:	The user is not yet logged in the application
Postconditions:	The user is logged in the application and entered into the Homepage
Main Success Scenario:	1. The user starts the application 2. The user inserts a valid username and password 3. The system verifies the username and password 4. The user is redirect to the Homepage
Extensions:	1. If the last log-in of the user is still active (within 30 minutes) he's directly redirect to the Homepage

	2. If the user doesn't remember his password can ask to the system to send a new one by email
	3. If the username and password are not correct the system asks to the user to reinsert them
Frequency of Use:	Several times in an hour
Priority:	P1 - very high

ID:	UC-3
Title:	Use a car
Description:	A user drives a car
Primary Actor:	User
Preconditions:	The user has reserved the car and he's near the reserved car
Postconditions:	The car returns available
Main Success Scenario:	1. The user starts the application
	2. The user taps the button "unlock"
	3. The system unlocks the car
	4. The user enters into the car
	5. The user pushes the button "starts&stop"
	6. The system starts the charge counter
	7. The user uses the car
	8. The user parks the car in a parking area
Extensions:	4. if the user doesn't enter into the car within 2

minutes the system locks the car

5. if the user doesn't start the car within 10 minutes the system starts to charge him

8. if the user parks a car in a different area and goes out of the car for more than 2 minutes the system locks the car and if the user doesn't return into the car within 30 minutes a request of assistance is sent to the maintainers

Frequency of Use:	Several times in an hour
Priority:	P1 - very high

ID:	UC-4
Title:	Reserve a car
Description:	A user reserves a car
Primary Actor:	User
Preconditions:	The user has searched the available cars near a specified position
Postconditions:	The car is reserved for the user for up to one hour before he picks it up
Main Success Scenario:	1. The user select one of the available cars showed on the map 2. The application shows at the user all the details about the car like position, type of car, battery status. 3. The user taps the button "confirm" to reserve the selected car

Extensions:	3. The user can tap "decline" to reject the reservation and he's redirected to the page with the map and all the available cars near the position specified before.
Frequency of Use:	Several times in an hour
Priority:	P1 - very high

ID:	UC-5
Title:	Search for a car
Description:	A user searches the available cars
Primary Actor:	User
Preconditions:	The user has correctly logged in the application and he's in the Homepage. The user hasn't any other active reservation at the moment.
Postconditions:	The system shows to the user all the available cars
Main Success Scenario:	1. The user taps the "*Reserve a car*" button 2. The user inserts a specific place or select one from the history 3. The systems search all the available cars near the specified place and shows them on the map
Extensions:	There're no extensions for this use case
Frequency of Use:	Several times in an hour
Priority:	P1 - very high

ID:	UC-6
Title:	See past bookings
Description:	A user can see all his past reservations
Primary Actor:	User
Preconditions:	The user has logged in the application and he's on the Homepage
Postconditions:	There're no post-conditions for this use case
Main Success Scenario:	1. The user taps on the "*past bookings*" button 2. The system shows to the user all his past reservations 3. The user taps on one of them 4. The system shows all the details of the selected reservation like the date, the time, the route and the charge
Extensions:	2. If the user has never made a reservation the system doesn't show anything
Frequency of Use:	A few times a day
Priority:	P5 - very low

ID:	UC-7
Title:	Modify personal data
Description:	A user modifies his personal data

Primary Actor:	User
Preconditions:	The user has logged in the application and he's in the Homepage
Postconditions:	The data of the user are modified
Main Success Scenario:	1. The user taps the "*modify personal data*" button 2. The applications shows at the user all his personal data 3. The user taps the button "modify" to change a specific data 4. The user inserts a new value for this data 5. The user taps the "confirm" button 6. The system checks and modify the data
Extensions:	5. If the user taps the "decline" button then the modification is aborted and he's redirect to the "modify personal data" page 6. If the data are invalid or not coherent the system notify to the user the error and the modification is aborted, the user is redirect to the "modify personal data" page
Frequency of Use:	A few times per week
Priority:	P4 - low

ID:	UC-8
Title:	Ask for support
Description:	A user asks for support to us

Primary Actor:	User
Preconditions:	The user has correctly logged in the application and he's in the Homepage of our application or in the "Reserve a car" or "Personal data" pages
Postconditions:	There're no post-conditions for this use case
Main Success Scenario:	1. The user taps the "Support" button 2. The application shows to the user all the information that he need for asking support 3. The user can send an e-mail to us or directly call our assistance service
Extensions:	There're no extensions for this use case
Frequency of Use:	A few times a day
Priority:	P3 - medium

ID:	UC-9
Title:	Cancel reservation
Description:	A user can cancel his reservation
Primary Actor:	User
Preconditions:	The user has logged in the application and he's on the Homepage
Postconditions:	The reservation of the user is deleted
Main Success Scenario:	1. The user taps on the "*cancel a reservation*" button 2. The system shows to the user his reservation

	3. The user taps "*cancel reservation*" button
	4. The system delete this reservation
	5. The user is redirect to the Homepage
Extensions:	3. The user can go back to the Homepage without deleting his reservation
Frequency of Use:	A few times a week
Priority:	P3 - medium

ID:	UC-10
Title:	Forgot password
Description:	A user can obtain a new password
Primary Actor:	User
Preconditions:	The user has forgotten his password
Postconditions:	The user obtains a new password
Main Success Scenario:	1. The user starts the application
	2. The user taps on the link "forgot password?"
	3. The user taps the button "send new password"
	4. The system sends to the user a new password via e-mail
Extensions:	3. If the user remembers his password he can go back to the log-in page without changing it
Frequency of Use:	A few times a week
Priority:	P4 - low

ID:	UC-11
Title:	Charge a car
Description:	A user puts in charge a car
Primary Actor:	User
Preconditions:	The user has used the car and parked it in a parking area The parking area has the recharging stations
Postconditions:	The battery of the car is recharged
Main Success Scenario:	1. The user plug-in the car to the charging station 2. The system applies a discount to his fee
Extensions:	There're no extensions to this use case
Frequency of Use:	Several times a day
Priority:	P1 - very high

ID:	UC-12
Title:	Park a car
Description:	A user parks a car
Primary Actor:	User
Preconditions:	The user has used the car
Postconditions:	The car is correctly parked in a parking area
Main	1. The user check the screen on board to find the

Success Scenario:	nearest parking area
	2. The user parks the car in a parking area
	3. The user exits the car
	4. The system locks the car
	5.The user pays the fee
Extensions:	4 If the area is a special parking area and the user plug in the car into a charging station the system applies a discount.
	4 If the user deserves any other discount it will applied on the total amount before to charge him.
Frequency of Use:	Several times per hour
Priority:	P1 - very high

ID:	UC-13
Title:	Repair a car
Description:	A maintainer repairs a car
Primary Actor:	Maintainer
Preconditions:	The maintainer has received a request of support
Postconditions:	The car is repaired and he's now available for the users
Main Success Scenario:	1. The maintainer log-in the application
	2. The system shows to the maintainer all the request of support for the broken cars
	3. The maintainer goes to a broken car

	4. The maintainers takes care of the car repair 5. The maintainer taps on the "*car repaired*" button
Extensions:	4. If the maintainer doesn't repair the car the request remain pendant
Frequency of Use:	A few times a day
Priority:	P2 - high

ID:	UC-14
Title:	Maintainer log in
Description:	A maintainer can log-in to the application
Primary Actor:	Maintainer
Preconditions:	The Maintainer is not yet logged in the application
Postconditions:	The user is logged in the application and entered into the "List of requested of assistance" page
Main Success Scenario:	1. The maintainer starts the application 2. The maintainers inserts a valid username and password 3. The system verifies the username and password 4. The maintainer is redirect to the "*List of requested of assistance*" page
Extensions:	1. If the last log-in of the maintainer is still active (within 30 minutes) he's directly redirect to the "List of requested of assistance" page 3. If the username and password are not correct the system asks to the maintainer to reinsert them

|	
Frequency of Use:	A few times a day
Priority:	P2 - high

ID:	UC-15
Title:	Log-out
Description:	A user log-out to the application
Primary Actor:	User
Preconditions:	The user is logged in the application
Postconditions:	The user is not logged in the application
Main Success Scenario:	1. The user taps the "log-out" button 2. The application is closed
Extensions:	There're no extensions for this use case
Frequency of Use:	Several times in an hour
Priority:	P1 - very high

The UC-11(charge a car) and UC-12 (park a car) can both be executed by the maintainers if and only if the user doesn't do it and so a request of assistance is automatically sent by the system to the maintainer who has to do these operations in order to bring the system in a correct state.

Also the UC-15 can be performed by the maintainer in order to log-out to the application.

3.6.2. Class diagram

Here is presented the first version of the Class diagram of our system, that shows the entities and the relations between them. The diagram will be updated during the developing process, especially adding attributes and methods.
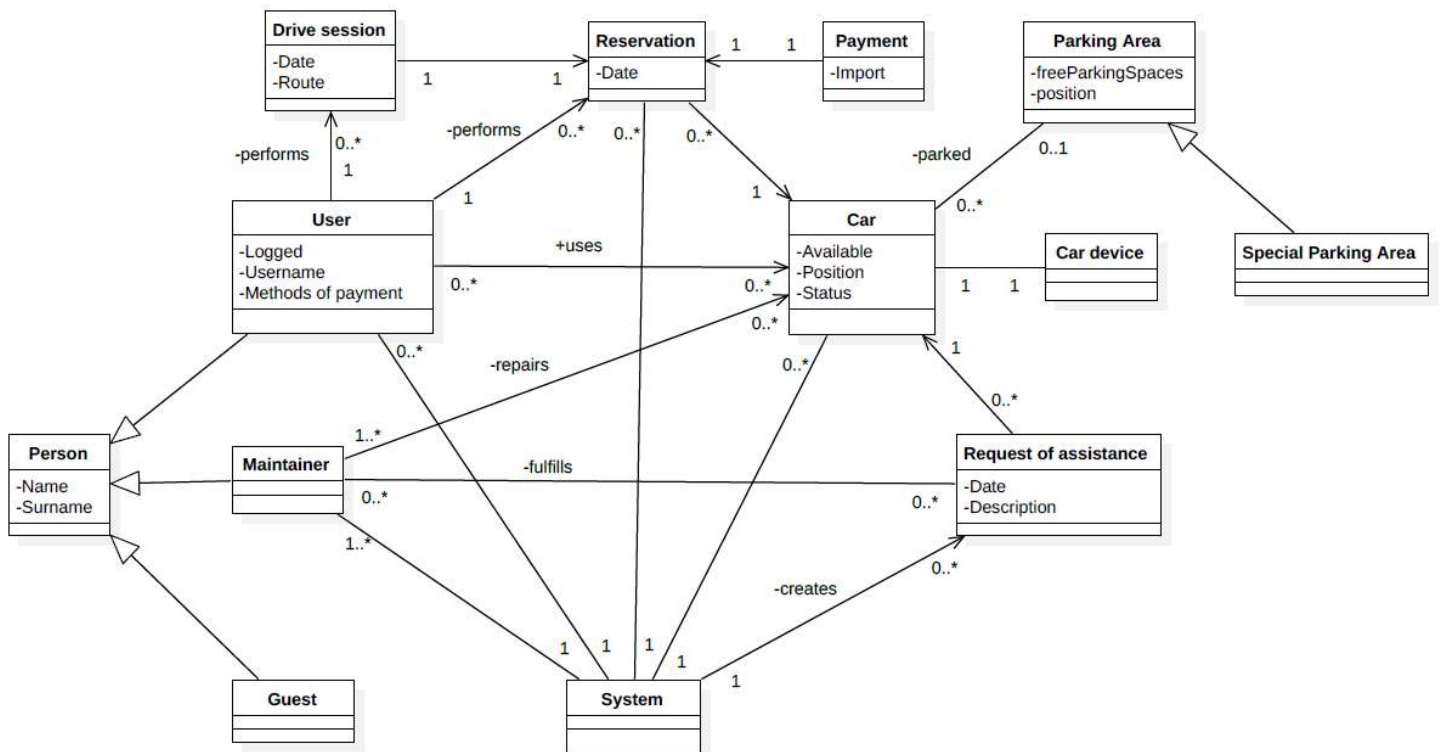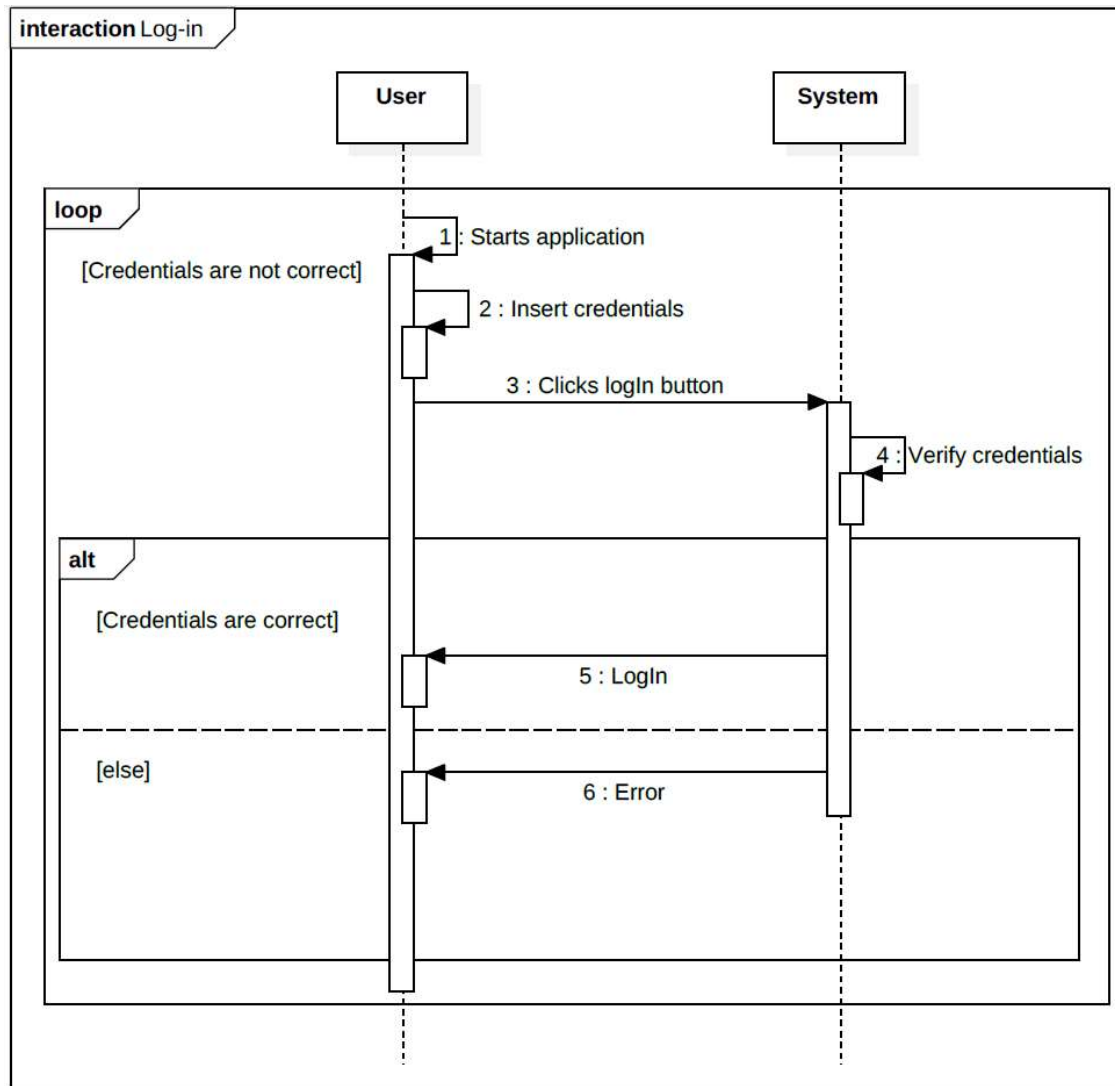
3.6.3. Sequence diagrams

Here we want to represent some sequence diagrams that better clarify the interactions between entities in our system.
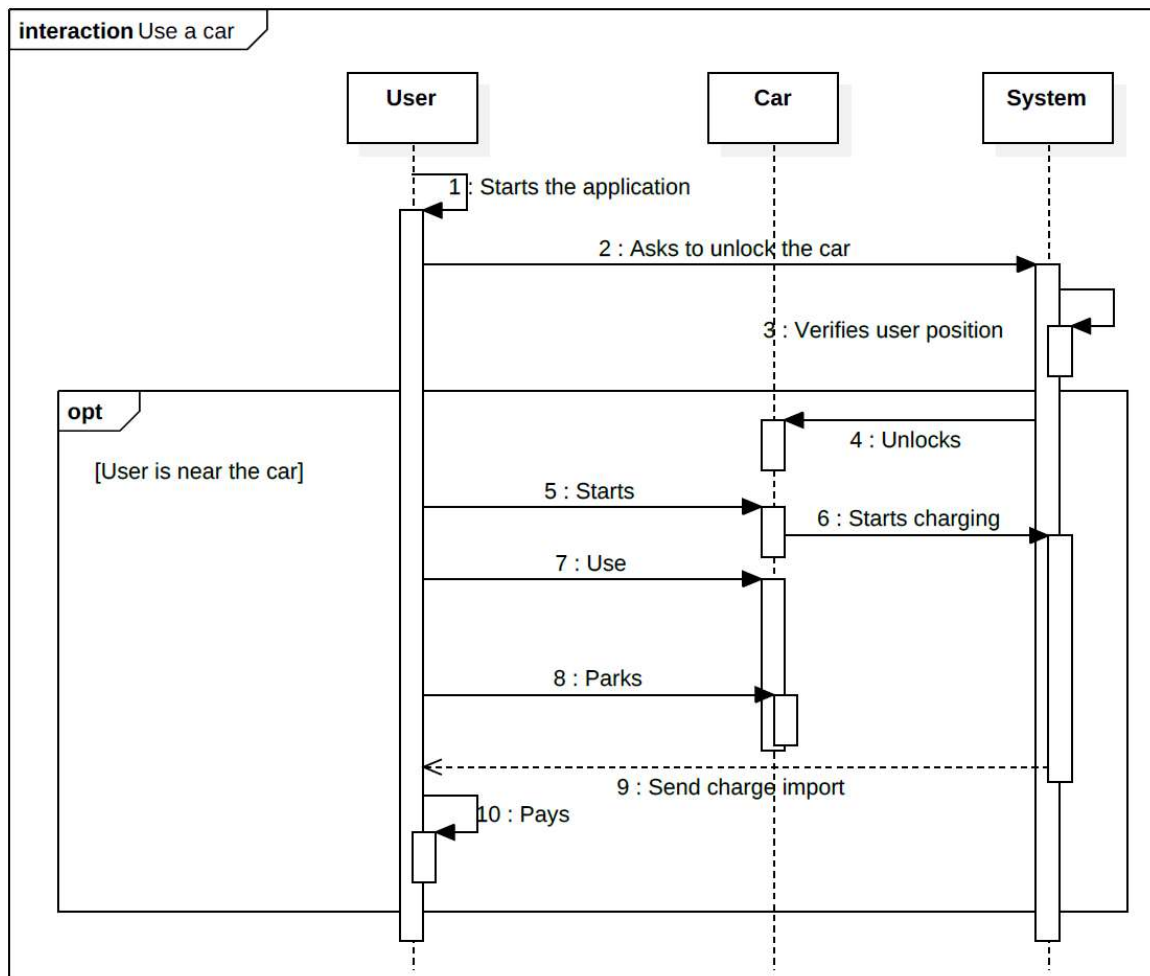
The First sequence diagram represents the interactions between a guest and the system in order to register him. (related with UC-1 Sign-up)
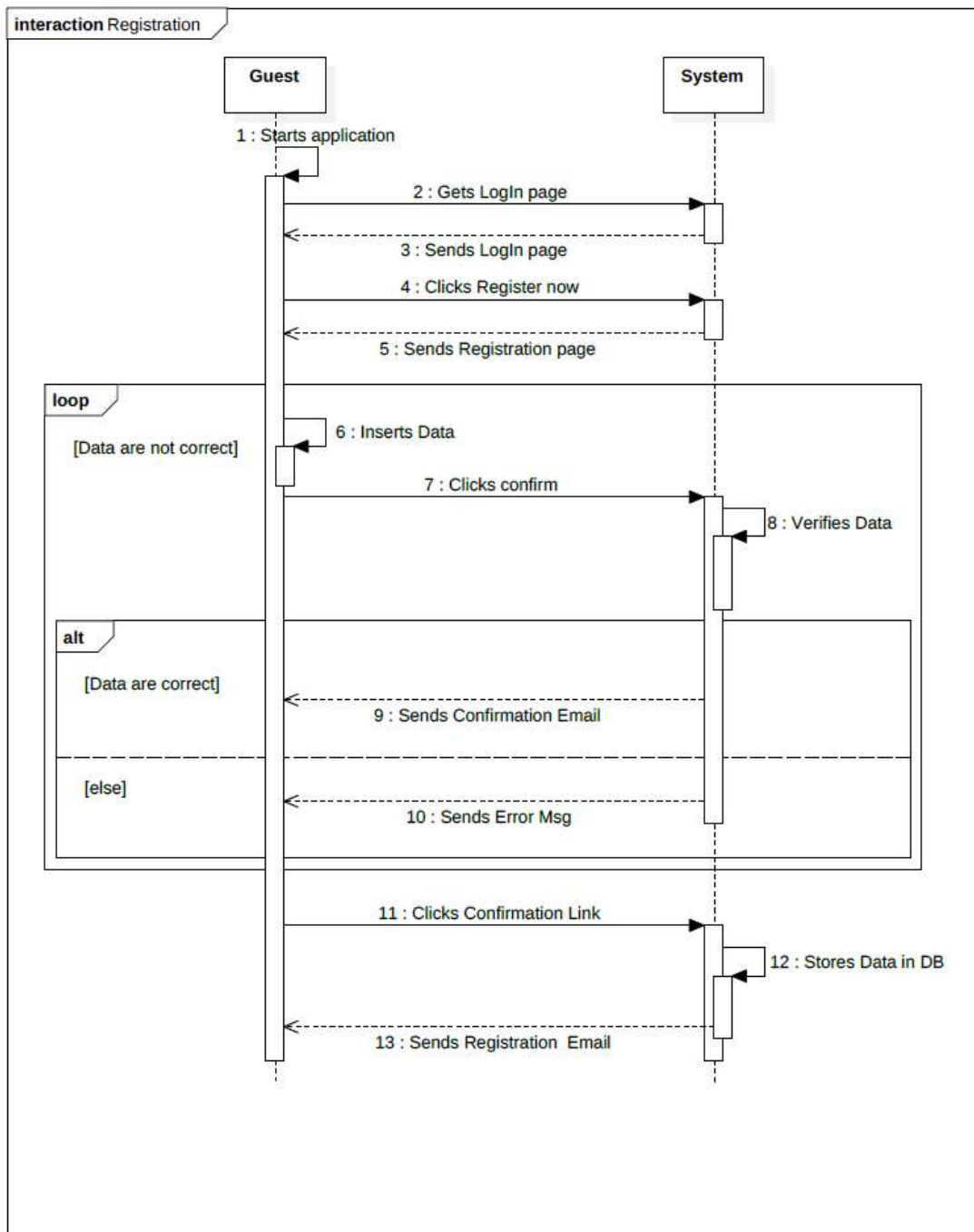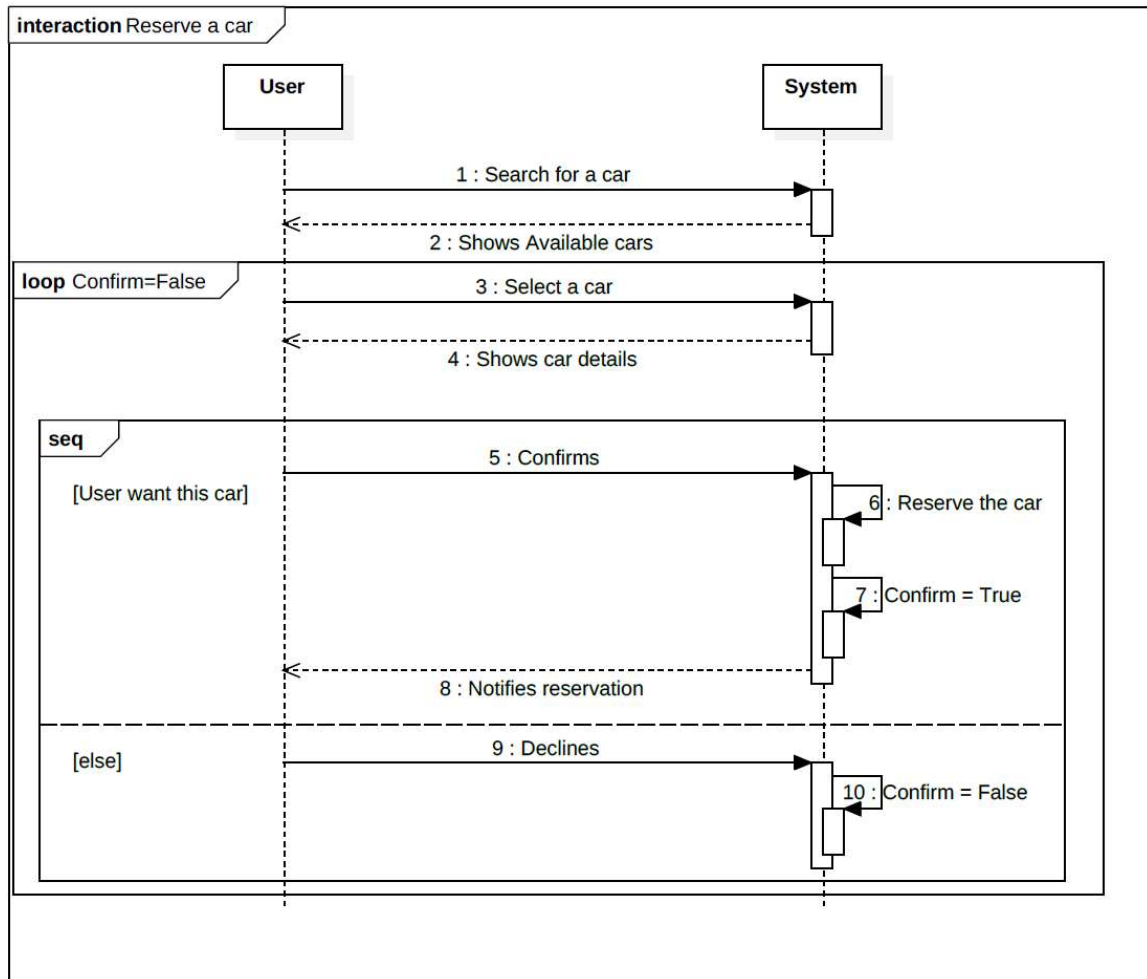
The second sequence diagram represents the log-in of a user.
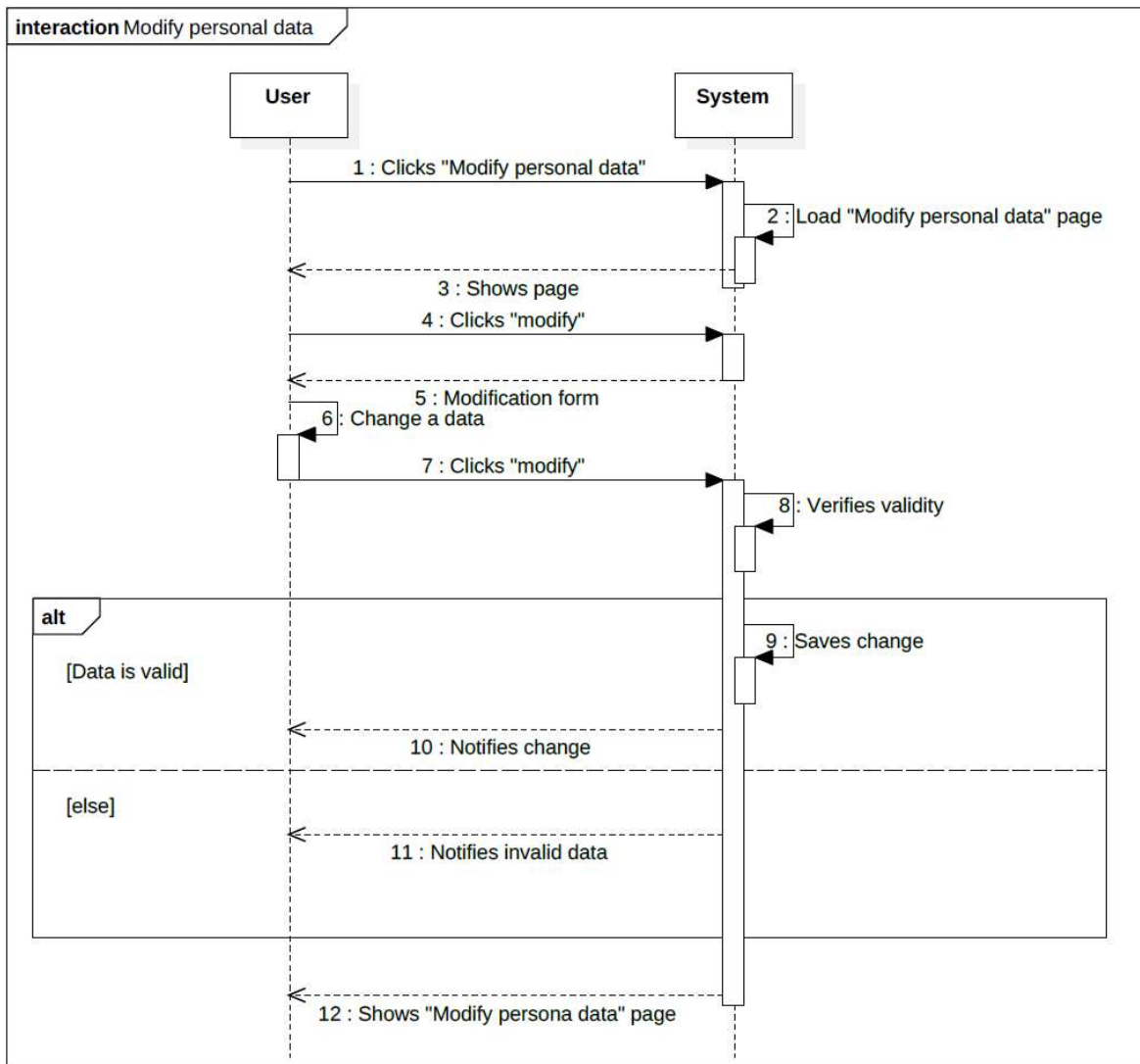
(related with UC-2 Log-in)

This sequence diagram represents the interaction between a user, a car and the system when a he want to use a car. (related with UC-3 Use a car)

This sequence diagram represents the interactions between a user and the system when he want to reserve a car (related with UC-4 Reserve a car)

This sequence diagram represents the interactions when a user want to modify some personal data. (related with UC-7 Modify personal data)
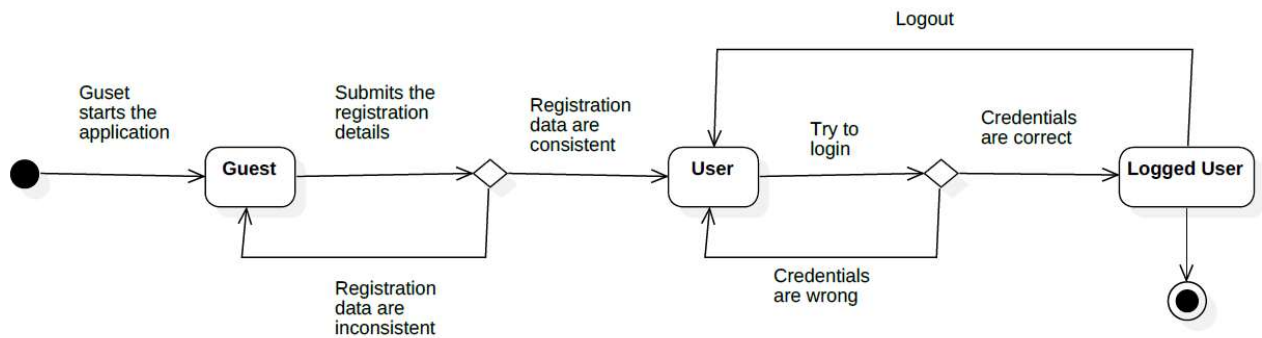
Finally we represents the interactions when a user wants to cancel his reservation. (related with UC-9 Cancel reservation)
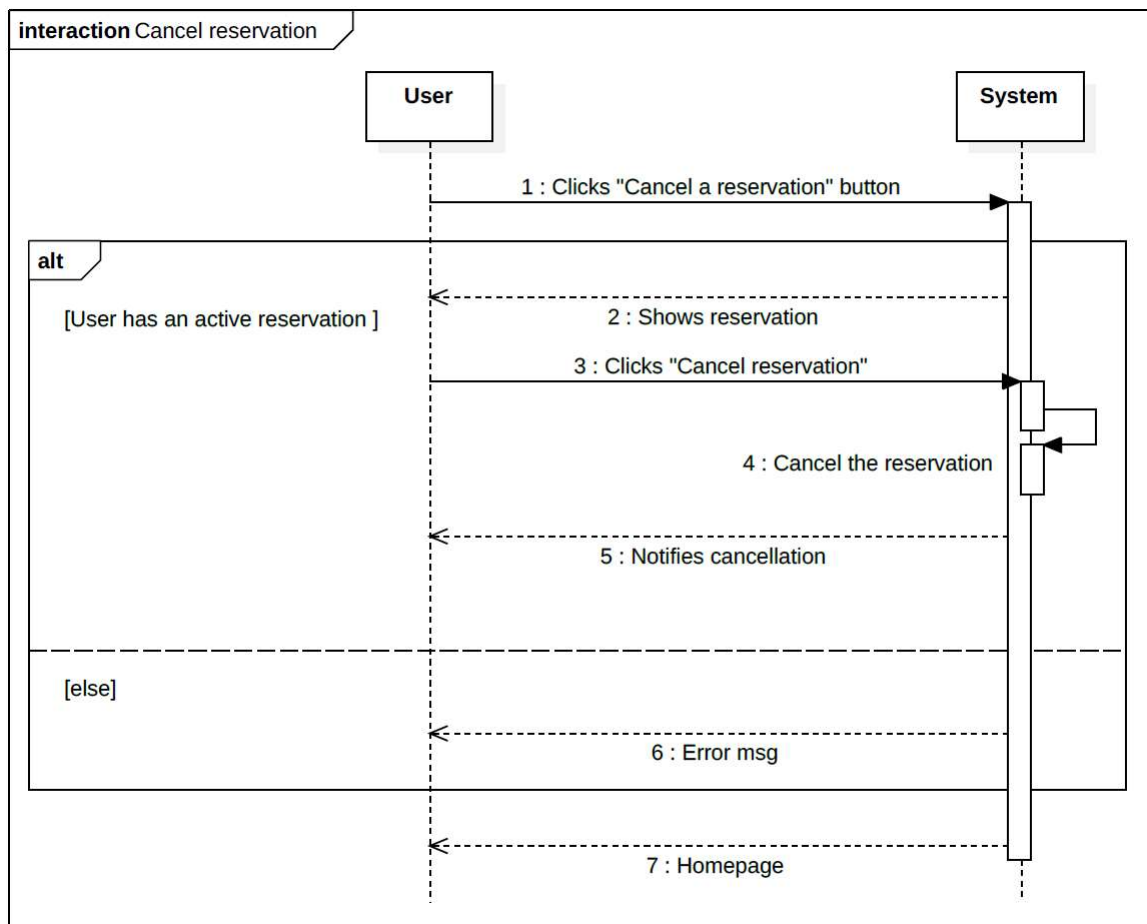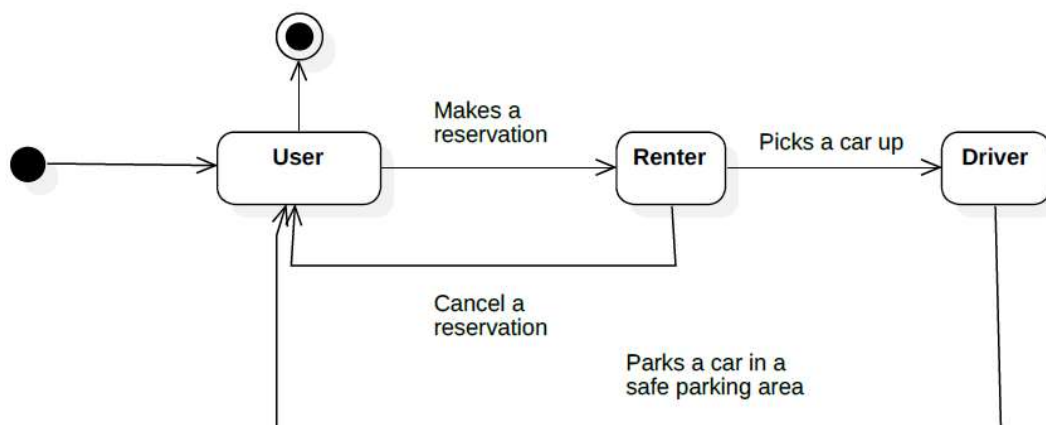
3.6.4. State Chart Diagrams

Guest to Logged User

Here we show the different states which can assume a person since the moment in which he is a Guest untill when he becomes User and performs the login procedure.


User uses a Car

In this diagram are visible all the states a user can assume while he is using the application.

3.6.5. Activity Diagram

This diagram is to simply illustrate the activities done by a user of the application who reserves a car, drives it, and then parks it. It doesn't pretend to be a precise representation which will be done in one of the next phases of projecting.
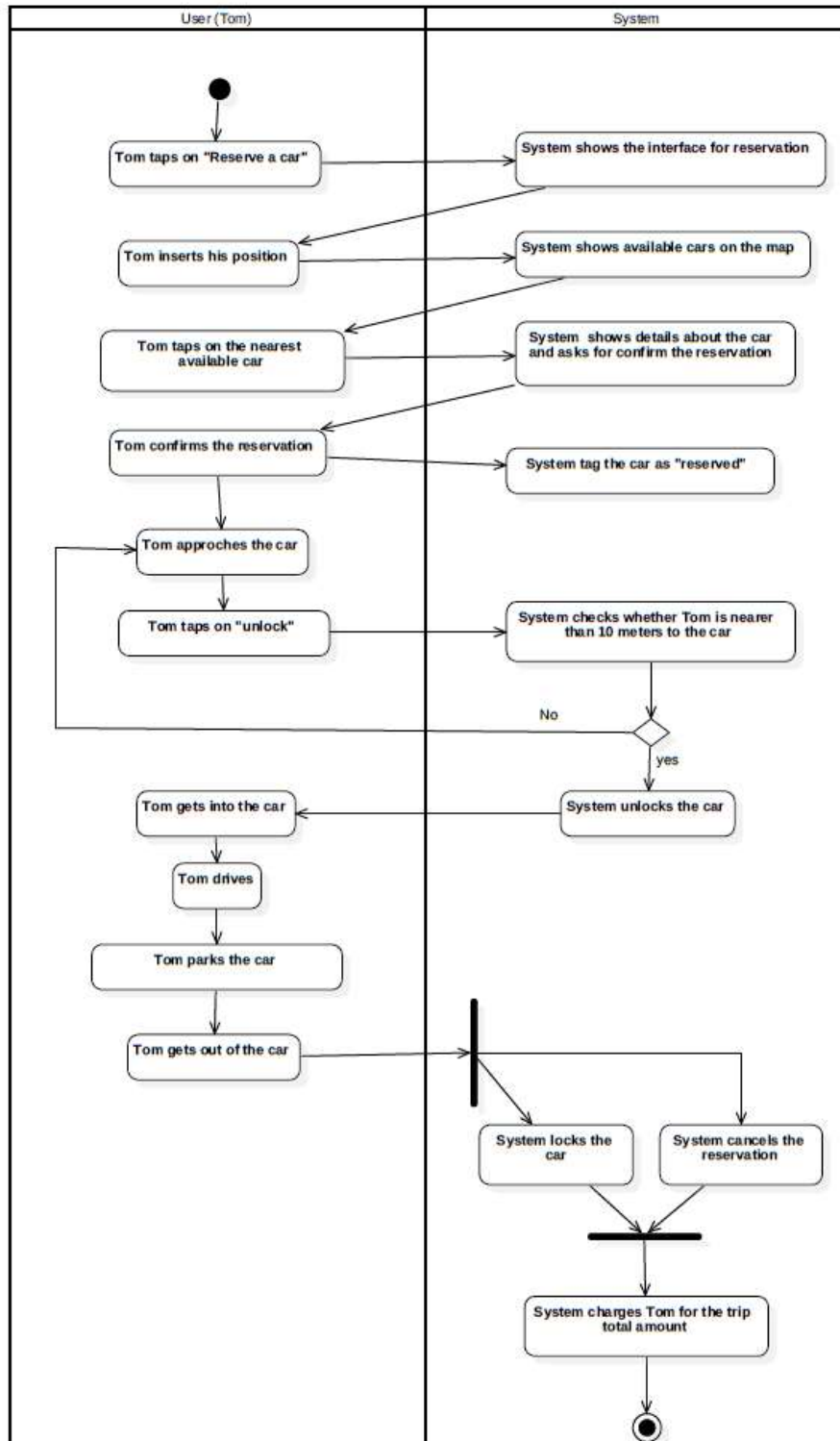
3.7. Design constraints

The application is written in java so will inherit all the language's constraints.
The application will run on Android version 5.0 or newer, and so will inherit all the consequent constraints.

3.8. Software system attributes

3.8.1. Availability& reliability

The application will work 24 hours a day and every day during the year. We guarantee a grade of availability of 98% thanks to our secondary server network that operates when the primary server goes down.

3.8.2. Maintainability

The application does not provide any specific API, but the whole application code will be documented to well inform future developers of how application works and how it has been developed. The effort for the ordinary maintenance is two person-months.

3.8.3. Robustness

Our system failure percentage is 0,1% and when the system falls down it takes about 3 minutes to restart it.

3.8.4. Scalability

Our system is able to manage an increasing number of users and data by increasing the dimension of our DB, and eventually adding new servers to our network.

3.8.5. Portability

The application will be runnable on any device which has installed Android version 5.0 or newer.

3.8.6. Security

The system is protected by the login procedure which allow only to registered user to use the application. Moreover, the system limit each account to one login at a time, in this way we avoid overlapping problems.

Through the registration procedure is possible to ensure the real identity of a person, thanks to the upload of documents copy. Each user who is able to drive must have uploaded a valid copy of his drive license, in this way we are sure he is able to drive our cars.

By this kind of validation is also possible to prevent double registration of people who, for example, have been banned from the system and would try to create a new account with the same data.

In order to protect our Data Bases our application should not be vulnerable to SQL injection attacks so the it shall not execute commands embedded in data provided by users that forces the application to manipulate the database table in unintended ways.

In order to guarantee the security of the payments, PowerEnJoy is supported by an external company.

Chapter 4

Alloy

4.1. Code

```
// Signatures - Data type //


sig string {}
sig Car{}


// Signatures //

abstract sig Person{
   name: one string
}

sig User extends Person{
   username: one string,
   password: one string,
   email: one string
}

sig Maintainer extends Person{
   username: one string,
   password: one string,
   repair: lone Car
}

sig SessionDrive{
   user: one User,
   car: one Car
}

sig Reservation{
   user: one User,
   car: one Car
}

sig ParkingArea{
   cars: set Car
}
```

```
// Facts //

//unique attributes
fact uniqueAttributes{
all disj u1,u2: User | u1.username != u2.username and u1.email != u2.email
}
//atLeastOneOf

fact atLeastOneMaintainer{
#Maintainer>0
}
fact atLeastOneParkingArea{
#ParkingArea>0
}

//a car must be driven by at most one user at time
fact noTwoDriversForACar {
all disj s1,s2 : SessionDrive | s1.car != s2.car
}

//a user can drive only one car at time
fact noTwoDrivingSessionByTheSameUser{
all disj s1,s2 : SessionDrive | s1.user != s2.user
}

//a car can be in oly one parking area at time
fact noCarInTwoParkingAreas {
all disj p1,p2 : ParkingArea | no (p1.cars & p2.cars)
}

//if a car is in a parking area then cannot exits a user who is driving it
fact noDriveCarParked{
no (ParkingArea.cars & SessionDrive.car)
}


//if a car is reserved then it must be driven by a user or it's in a parking area
fact noReserveCarNotInParkingAreas{
Reservation.car in (ParkingArea.cars + SessionDrive.car)
}

//a car under repair cannot be driven by a user
fact noDriveOrReserveBrokenCar{
no (Maintainer.repair & SessionDrive.car) and
no (Maintainer.repair & Reservation.car)
}

// a car can have only one reservation at time
// and a user can only one only one reservation at time
fact noTwoReservationForACar{
all disj r1,r2 : Reservation | r1.car != r2.car and
all disj r1,r2 : Reservation | r1.user != r2.user
}

//if and only if a user has reserved a car can drive it
fact driveReservedCar{
all s1 : SessionDrive | one r1: Reservation | s1.user=r1.user and s1.car=r1.car
}
```

```
// ASSERTIONS //

//doesn't exists a car that is both parked and driven
assert noDrivenAndParkedCar{
no c: Car | some p:ParkingArea, s: SessionDrive | c in p.cars and c in s.car
}

check noDrivenAndParkedCar for 5

//doesn't exitst a car that is reserved by a user and driven by a different one
assert notDifferentDrivenAndReservedUser{
no r: Reservation, s: SessionDrive | r.car=s.car and r.user != s.user
}

check notDifferentDrivenAndReservedUser for 5


// PREDICATES //

//add a car to a parking area
pred addCarToParkingArea(p,p': ParkingArea,c: Car){
p'.cars = p.cars + c
}

pred show{}
```

Follow others predicates which shows different worlds, each of which is briefly described.

4.2. Worlds Generated

Each user or maintainer have some string attribute related to his name, surname, username and password. We recommend not to give much importance to the relations generated between user, maintainer and these attributes as they are not so relevant.

World 1

```
//create World 1
pred showWorld1 {
    #User = 2
    #SessionDrive=1
    #ParkingArea=1
    #Reservation=2
    #Maintainer=1
    #ParkingArea.cars=2
    #Car=3
}

run showWorld1
```

We have forced the system to generate a world with 2 users, they both have reserved a car but only one of them have already picked it up. There is only a parking area in which are parked the two remaining cars, one of which is reserved by the other user. The third car is actually not available because of some technical problem, in fact, a maintainer is taking care about it.
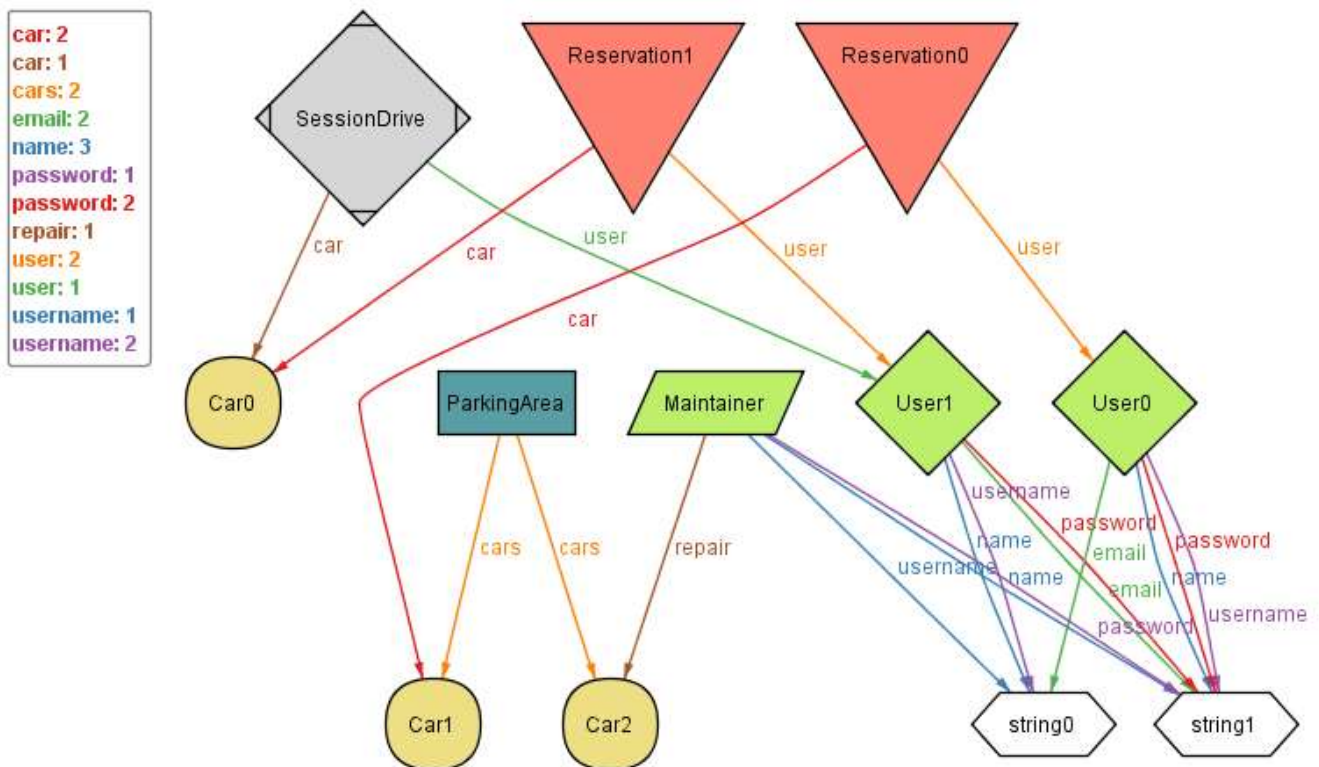
World2

```
//create World 2
pred showWorld2{
#Person=3
#Maintainer=1
#SessionDrive=3
#ParkingArea=2
}

run showWorld2 for 4
```

We asked the system for a world with 3 people, one of whom is a Maintainer, so the other 2 are Users. But we asked also to have 3 drive sessions active, which is obviously impossible since a maintainer can't reserve and drive a car. So the alloy result was:

Executing "Run showWorld2 for 4"

Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20

*2951 vars. 220 primary vars. 5674 clauses. 46ms.*

No instance found. *Predicate may be inconsistent. 11ms.*

World3

```
//create World 3
pred showWorld3{
#User=2
#Car=4
#ParkingArea=2
#SessionDrive=0
}

run showWorld3 for 4
```

Here we generated a world with 2 Users, 4 Cars and 2 Parking Areas. As we forced to have 0 drive sessions active, each park should be parked, except Car 3, which is actually in a place not labelled as Parking Area. This could mean (i.e.) that a user has abandoned it somewhere without coming back. The maintainer is taking care of it probably bringing it one of the safe parking areas. We can notice that in each world there is always at least one Maintainer in order to ensure an assistance service.
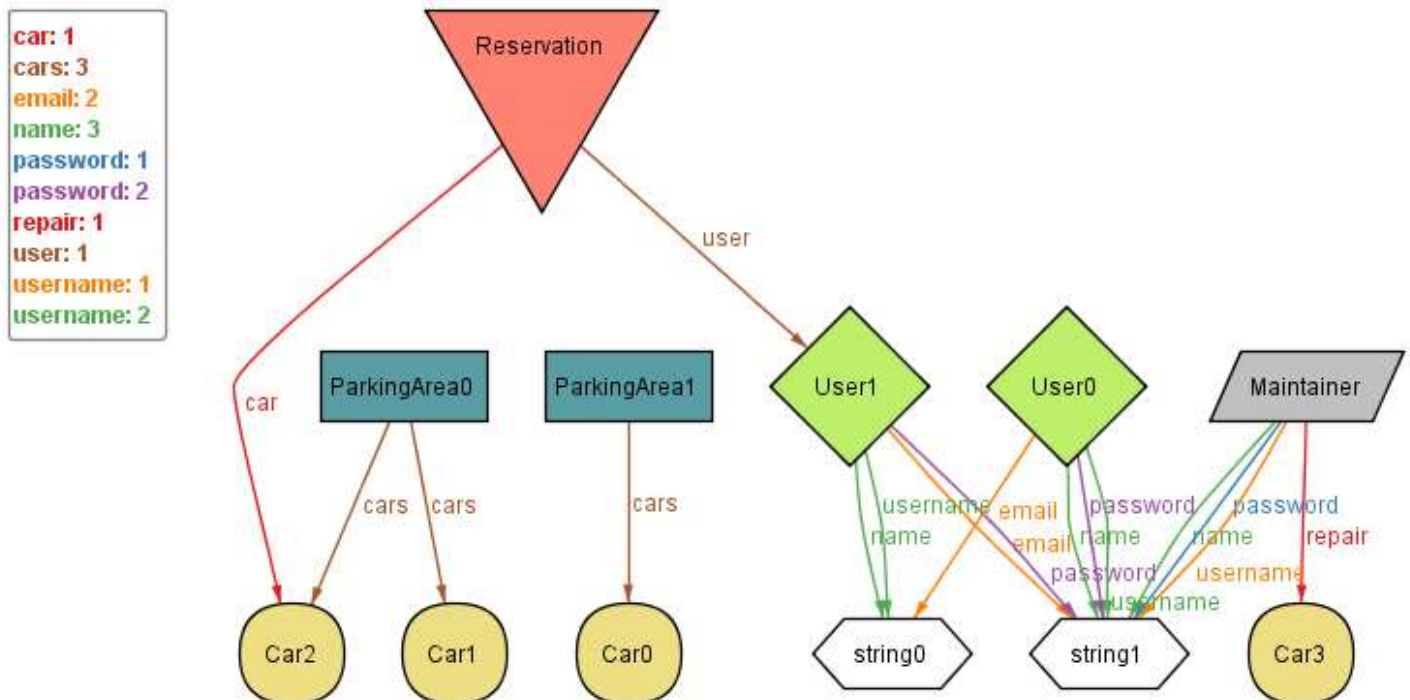
World 4

```
//create World 4
pred showWorld4{
#SessionDrive=3
}
run showWorld4 for 6
```

We have asked for a world with 3 drive sessions active, in this way most of the choices are left to the case. We expect a world with at least 3 users, 3 reservations and 3 cars.

In the result we have 6 cars, 2 of which are parked in Parking Area 5, one of these is being repaired by Maintainer 1, all the others parking area are free. Car0 is not parked in a safe parking area, this means that it is to be recovered by a maintainer, but this isn't already happened.
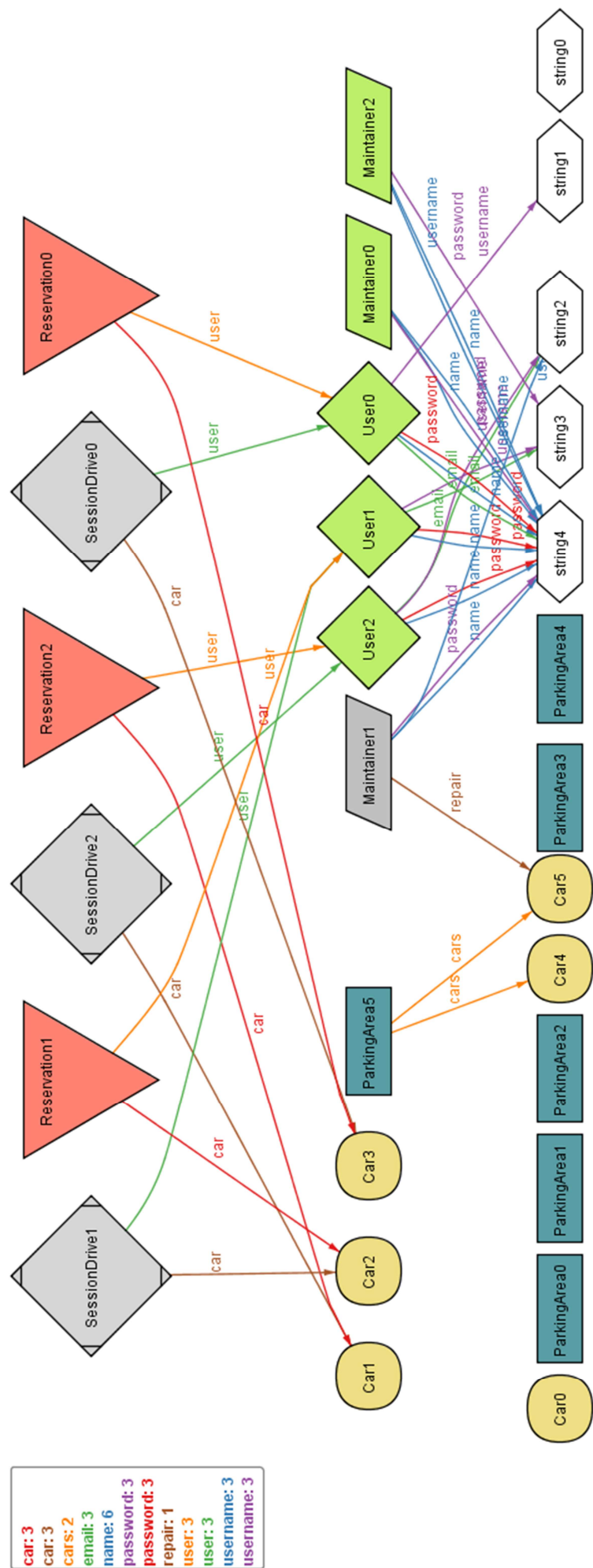
Chapter 5

Other information

5.1. Future development

Add a functionality that permits users to share also their journey (for long travels) with other users in order to save more energy and money.

5.2. Tools

- Microsoft Office World 2016: to redact and format this document
- StarUML: to create Use Case diagram, Class diagram, sequence diagrams, activity diagram and sequence diagrams
- Paint: to create and adapt the images of this document
- Alloy Analyzer 4.2 : to prove the consistency of our model
- FluidUI: to create the mocks of our application
- GitHub: to save and control the version of the document

5.3. Working hours

Date	Antonio's hours	Davide's hours
2016/10/21	2h	2.30h
2016/10/22	--	1.30h
2016/10/23	1.30h	3h
2016/10/28	2.30h	--
2016/10/29	3h	2h
2016/11/01	4h	4h
2016/11/02	5h	5h
2016/11/05	2.30h	2.30h
2016/11/06	--	4h
2016/11/09	6h	6h
2016/11/10	5.30h	5.30h
2016/11/11	6h	6h
2016/11/12	4h	2h
2016/11/13	1h	1h
Total	**43h**	**45h**