POLITECNICO
MILANO 1863

RASD

Software Engineering 2 // AA 2016 - 2017

Botta Daniel Enrico
806545

Bruschi Agnese
810762

Cano Adriana
812233

Index

1 Introduction **3**

1.1 Description of the given problem 3
1.2 Purpose 3
1.3 Scope 4
1.3.1 Domain properties 4
1.3.2 Assumptions 5
1.4 Definitions, Acronyms, Abbreviations 7
1.5 Overview 9
1.5.1 Proposed System 9
1.5.2 Identifying stakeholders 9

2 Actors identifying **9**

3 Specific Requirements **10**
3.1 Functional Requirements 10
3.2 Non Functional Requirements 13
3.2.1 Non Functional Requirements 13
3.2.2 User Interface 13
3.2.3 Performance requirements 16
3.2.4 Design constraints 16
3.2.5 Software System Attributes 16

4 Scenarios **17**
4.1 Identifying scenarios 17

5 UML Models **20**
5.1 Use case diagrams 20
5.2 Use case description 21
5.2.1 Register 21
5.2.2 Login 22
5.2.3 Visualize profile 23
5.2.4 Modify personal information 24

5.2.5 Search for a car 25
5.2.6 Reserve a car 26
5.2.7 Pick up a car 27
5.2.8 Cancel reservation 28
5.2.9 Pending state option 29
5.2.10 Money saving option 30
5.2.11 Payment 31
5.2.12 Field operator work 34

5.3 Use case sequence diagram 35

5.3.1 Register 35
5.3.2 Login 36
5.3.3 Search and Reserve 37
5.3.4 Payment 38

5.4 Class Model 38

6 Alloy Model 40
6.1 Alloy analysis 49

Work division details 52

Tools 52

1 Introduction

In this section we will introduce the problem we were addressed with and the stakeholders we answer to. We will describe the main objectives that will be taken into account, the world properties and the overall assumptions that were made to treat the problem.

1.1 Description of the given problem

The system to be developed is PowerEnJoy, a digital management system for car-sharing service that exclusively employs electric cars.

The system main service is to allow registered users to search for available cars within a certain distance from a given position (the user current location or a specified address). The user should be able to reserve the available cars up to an hour before the reservation time and should be allowed to pick it up for an hour after the said time. There are no time limitations on the duration of the reservation.

Given the user's behaviour during the rental, the system should grant discounts or apply extra fees. The system should also provide a special money saving option which would allow the user to save money and realize a fairer distribution of cars around the city.

1.2 Purpose

After analysing the stakeholder's requests, we identified the following goals for our system.

PowerEnJoy should provide these features for the **user**:

[G.1] Allow users to search for available cars within walking distance from their current location or from a specific address.

[G.2] Allow users to reserve an available car.

[G.3] Offer a money saving option that allows the user to gain discounts.

[G.4] Charge the user with a fee at the end of the service.

[G.5] Encourage the users to cooperate to improve their overall experience.

PowerEnJoy should provide these features for the **operator**:

[G.6] Notify that a car needs charging, it's damaged or malfunctioning.

1.3 Scope

1.3.1 Domain properties

The following properties will hold in the model designed by us.

[D.1] A user can enter in a car only if the car is unlocked.

[D.2] To ignite the engine, the driver must press the power button when the car is off.

[D.3] To switch off the engine the driver must press power button when the car is on.

[D.4] Each car has a socket where the car charger can be plugged in.

[D.5] Every car is equipped with sensors that can tell:

- the number of people in the car
- if the car has suffered consistent external damages
- if the car suffers internal damages
- the battery charging state

[D.6] Every car is equipped with an external display to insert the reservation code.

[D.7] Each car offered has five seats.

1.3.2 Assumptions

Some facts or descriptions are not mentioned or ambiguous in the document, so we will assume some of them.

[A.1] The system is able to check the correctness of the facts stated by the user over:

- Email: can check the validity of an e-mail address
- Driving licence: can check the existence and belonging of the driving licence ID. Name and surname of the user must match the driving licence name and surname.
- Credit card: can check the legitimacy of the credit cards information.
- Credentials: can ensure the existence of the address

[A.2] When a driving licence or a credit card expires, the system temporarily disables the account, sending an e-mail to the user, to notify him/her the fact.

The account will be reactivated once the user updates the expired credentials with valid ones.

[A.3] If the credit card is not able to cover the costs of the fee, the system is able to detect it.

- The system will send an e-mail to inform the user that its payment was unsuccessful.
- The service will be disabled until the payment is confirmed.
- The user has 24 hours to replace the credit card.
- Eventual payments failures and problems will be handled by external partners.

[A.4] Every mail sent always arrives to the designated address.

[A.5] The user's clock is in tune with the time zone of his location.

[A.6] The system allows the user to *freeze* its reservation:

- for up to two hours
- The user is charged with a lower, constant fee per minute.
- If the user does not come back after two hours, the system ends the reservation, calculates the fee
- if the car is parked in a safe area, the user is charged with an extra fee and the car is settled as available again
- if the car is parked in a non-safe area, the user is charged with an extra fee and an operator is called to move the car to a safe area.

[A.7] The system discourages the user to park outside the safe area by keeping active the charging system if the user leaves.

[A.8] The system generates a unique reservation number for each reservation made.

[A.9] The user has a *working phone*.

[A.10] The system employs field operators. Each field operator has his own profile. Fields operators can:

- log in the profile
- check notifications
- confirm resolution of problems

[A.11] Car issues are handled by the field operators.

Issues include**:**

- Car not parked in a safe area
- Low battery
- Eventual interactions with third parties

[A.12] At the end of each reservation, the sensors check the state of the car. If any suspicious value is found, the car informs the system, which sends a notification to the operator.

The system informs the field operator through a notification on his profile.

1.4 Definitions, acronyms, abbreviations

Guest — A person who has yet to sign up to the system. Cannot use any functionality apart from registering to the system.

User — Person registered to the system. To register into the system, the following information is given:

1. *Credentials*: name, surname, address.
2. *Driving Licence*: ID, Name and Surname of the driver, expiring date of the licence.
3. *Credit Card*: name and surname of the possessor of the card, card number, card expiring date, CVV.

Reservee — User who has reserved a car.

Reservation time — Time of reservation of a car.

Reservation code — Code given to the user once he has reserved a car. It is needed to unlock the car. (7 digits code).

Freeze — The car is freezed when the reservee, that is already using it, parks and decide to stop for a certain amount of time, all while still reserving the car.

Area — The space occupied by a car. It may contain a Charging Point.

Safe area — Area predefined by the management system where the user can park the car once he/she has finished using the service.

Travellers — Driver and passengers.

Locked / Unlocked — When the travellers exit the vehicle the car is in a locked state. To unlock it the user must insert the same reservation number.

Sound — A user is sound if he has reserved a car and if the current time satisfies this inequality:

reservation_time < current_time < reservation_time + 1h

Uniform distribution of cars: the system aims to provide a distribution of cars that ensures the best possible coverage of the city, so that the possibility that a user finds a car near the defined address is higher.

Walking distance — A distance that would take up to 10 minutes to reach on foot.

Working phone — A phone that has the GPS signal and internet connection working correctly and the PowerEnJoy app installed.

Coherent car A car is coherent with a search if it's available and it its position is within the range of the search.

Car labels:

- **AVAILABLE:** Car that is not reserved and that is parked in a safe area.
  Can be reserved by users.
- **RESERVED:** Car is already reserved and waiting to be used.
- **IN USE:** The reservee is using the car.
- **PENDING:** The reservation is frozen.
- **MAINTENANCE:** The car has an internal or external damage that needs to be dealt with or car needs to be recharged by an operator.
  Either way, the car will be marked as AVAILABLE again once the specific problem has been dealt with.
- **LOCKED:** A car is locked when no user is using it.
- **UNLOCKED:** A car is unlocked when a user is using it or when it is under maintenance.
- **TEMOPORARY LOCKED:** a car is temporarily locked when it is temporarily parked somewhere and the user is not done using it.

Reservation states:

- **VALID:** A reservation is VALID when the car is reserved and the user is still in time to pick it up.
- **CONFIRMED:** A reservation is CONFIRMED when the user enters the car and ignites the car.
- **FROZEN:** A reservation is FROZEN when the PENDING option is chosen. The car must be parked and turned off.
- **COMPLETED:** A reservation is COMPLETED when the payment has been successfully concluded.
- **CANCELED:** A reservation is CANCELED by the user (at least 10 minutes before the reservation time) or by the system.

1.5 Overview

1.5.1 Proposed System

We propose a mobile application that will be able to give users the power to enjoy driving through the city.

Users will be able to reserve cars through the application, using a simple interface via touchscreen. To access any service offered by the system they shall first register to the system. In the profile created, the user can change the information provided. The users will be able to search for available cars in the chosen area, and to reserve the one that they prefer. They also have the possibility to modify their reservations or cancel them.

1.5.2 Identifying Stakeholders

We consider the main stakeholder the professor, to whom we will deliver the document.

For the first document, the RASD, we are required to describe the goals of our project, identify the domain properties and determine the requirements that will regulate our system behaviour. Moreover, we will analyse the different use case scenario and fully describe the components that will constitute the body of the application, deepening our evaluation describing the functionalities with Alloy.

We will focus first on the main functionalities of the system, then we will deepen our description, trying to cope with every aspect of the system that may improve the user's experience of the application. Starting from this last consideration, we can imagine that a theoretical stakeholder for our system could be a company that would have asked us to realize an application to give the chance to users to benefit PowerEnJoy car service functionalities.

2. Actors Identifying

The actors of our system are the following:

- **GUEST:** A person who has yet to sign up to the system. Cannot use any functionality offered by the system.
- **USER:** A person who is registered to system, he is in possession of a verified driving license and a valid credit card.
- **FIELD OPERATOR:** employee at service of the system.

3. Requirements

Requirements are the shared phenomena between the real world and our software-to-be that can be used by the machine to control the events to achieve the goals.

Requirements can be functional or non-functional.

3.1 Functional Requirement

A functional requirement is necessary to describe the correct functioning of the system.

Management of the user's profile:

These requirements are essential to reach the goals, but do not belong to a specific one.

As a matter of fact, to achieve every goal required the user must be registered and able to modify and visualize his/her profile.

[F.R.1]: A guest can benefit the application's services only if he/she is registered.

[F.R.2]: A user's registration is confirmed if and only if his/her credentials match the credentials associated to the driver licence inserted.

[F.R.3]: A user should be able to log in in the system.

[F.R.4]: A user can log in if and only if he/she inserts the right combination of e-mail address and password.

[F.R.5]: A user should be able to request a password recovery.

[F.R.6]: A user should be able to update his/her payment method.

Search of a car [G.1]:

[F.R.7]: A user can search a car if and only if he/she is registered.

[F.R.8]: The system shows to the user only the cars that are coherent with his/her search.

[F.R.9]: A car is coherent with a search if and only if it's available and its position is within the search range.

[F.R.10]: A car can be selected if and only if it's shown in the search.

[F.R.11]: A selected car can be reserved.

Management of car reservation [G.2]:

[F.R.12]: A user can reserve a car if and only if there is not another car reserved by him/her in the desired time interval.

[F.R.13]: To reserve a car the user must insert the reservation time.

[F.R.14]: The reservation time must be: $t_{RESERVATION} <= t_{CURRENT} + 1$ Hour

[F.R.15]: The picking up time must be: $t_{PICKUP} <= t_{RESERVATION} + 1$ Hour

Pick up the car [G.2]:

[F.R.16]: A reserved car is unlocked if and only if the correct *reservation code* is inserted.

[F.R.17]: A car parked in a safe area is locked if there are no travellers in the car.

Money Saving Option [G.3]:

[F.R.18]: The money saving option provides the address of the station near the destination address which will guarantee a discount.

The station will be chosen taking into account the distribution of cars around the city and the capacity of the stations. The calculation will be handled by an external system.

The discount is not assured, until the car is actually parked in the station and connected to the power grid.

Payment [G.4] [G.5]:

[F.R.19]: The system calculates the fee once the ride is concluded.

[F.R.20]: The ride is concluded when the user parks in a safe area, turns off the car and exits from the car without choosing the freeze option.

[F.R.21]: The fee is calculated based on a fixed amount of money per minute.

The amount of money to be paid is chosen by the administrators of the system.

[F.R.22]: Before applying the fee, the system checks if the ride is eligible for a discount.

[F.R.23]: If for the whole ride there were more than three travellers, the system applies a 10% discount.

[F.R.24]: If the user connects the car to a power grid of one of the special parking areas after the ride, the system applies a 30% discount.

[F.R.25]: If the user leaves the car with more than 50% of the charge left, the system applies a 20% discount.

[F.R.26]: If the user leaves the car more than 3 km away from the nearest power grid station, the system charges him/her 30% more.

[F.R.27]: If the user leaves the car with less than 20% of the charge left after the ride, the system charges him/her 30% more.

3.2 Non-Functional Requirements

3.2.1. Non Functional Requirements

[N.F.R.1]: The server should be available 24 hours a day, 7 days a week.

[N.F.R.2]: The field operators must be available throughout the whole availability period.

This means that they will be required to take turns.

3.2.2 User Interface

The user will be able to interact with the system through an interface which will allow him to perform all the requested activities.

The first page that the interface will show is a log in / sign in page.

Log in:

Figure 3.A

When the user accesses to the system, he / she is able to choose between the following services:

- Press the top right button, which will show the personal profile with all the data. This will allow the user to keep updated all his / her personal information and the payment and driving licence data.
- Start a search
- Log out
- Check the current reservation.

Search:

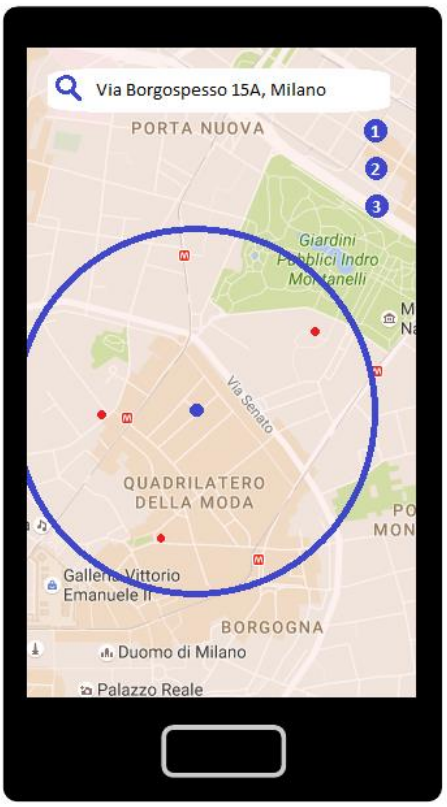


Figure 3.B

The user can choose between three ranges of search (the blue dots 1, 2, 3) and the app shows the search address (blue dot), the range limit (blue circumference) and the available cars which are coherent to the search. (as shown in figure 3.B).

To visualize car properties, the user needs to press the red dot representing the car.

Figure 3.C represents the car's main properties. From here the user can proceed to reserve the car.

Figure 3.C

At the end of the reservation the user will be informed of the finale fee. He can have access to the reservation page through the main page of the application during the whole time that the reservation is in progress.

Figure 3.D

3.2.3 Performance requirements

The software product requires that every web page shall download in 10 seconds or less.

3.2.4 Design constraints

The software product must be designed and implemented with JEE technologies, in particular EJBs for the business logic.

3.2.5 Software system attributes

Reliability

The software product ensures reliability in the maintenance of the data and the operations of payment.

Availability

The system shall be available 24 hours per day,7 days a week.

Security

The software product shall provide secure storage of the personal information inserted as well as the password generated which is used to access the personal account.

Maintainability

The database has a backup of the information in use and it is periodically updated so that in case of malfunction there is no data loss.

Portability

The software product can be installed in any smartphone that supports the java virtual machine and its dependent components.

Other requirements

The software product must inform and instruct the user in case of errors through clear textual messages.

4.1 Scenarios Identifying

[SCENARIO 1] – *Michael registers*

Michael has a driving license but doesn't have enough money to buy his own car. He searches for various options on the internet and finds PowerEnjoy. After reading how it works, he decides that he wants to try it out, and decides to download the application on his smartphone. He makes the first step in order to create an account. A form which he must complete. He must insert the following personal information: name, surname, e-mail, phone number, driving licence number, credit card number. The system accepts his registration and Michael can visualize his personal page.

[SCENARIO 2] – *Michael searches and reserve a car*

Michael is interested to utilize PowerEnjoy. So he goes onto the application and searches for a Car using the "Search for car" service. He turns on his GPS and selects the radius distance. The system then shows him on the map the available cars in the designated area.

After some thinking he selects the best option for him, chooses a time and the system reserves the car for him sending him a reservation code. Michael presents himself at the car at the specified time, goes on his mail account and looks at the reservation code. He can now unlock the car with it and drive.

[SCENARIO 3] – *Michael cancels the reservation*

Michael has reserved in advance a car but being a busy man he has many appointments to keep track of. He realizes that he has reserved a PowerEnjoy car at the time of his meeting, and so he couldn't pick it up. He decides to cancel the reservation, going to his profile and choosing the "Cancel reservation" option.

[SCENARIO 4] – *Michael does not show up*

Michael has reserved a car. He decided to get some sleep before he had to hit the road. He clumsily overslept and didn't manage to make it in time to pick up the car. A notification arrived to his phone informing that 1 EUR would be deducted from the credit card due to the fact that he reserved a car without picking it up.

[SCENARIO 5] – *Michael changes his personal profile*

Michael has changed some of his personal information and would like to update them in the PowerEnjoy app. He goes into his profile and can modify manually the credentials he desires. The system will verify if the changes are valid. If not the system will refuse them.

[SCENARIO 6] – *Michael uses the Pending option*

Michael has to do some routine shopping. He realizes he can use a PowerEnjoy car to make things easier. He travels to the shop and parks in a non-safe area. He puts the car in a “Pending” state. This reduces the fee per minute and can be used for a maximum of 2 hours per ride. Once he returns he can unlock the car and put the car back in “InUse” state.

[SCENARIO 7] – *Michael brings two friends*

Karen and Robert, Michael’s best friends, want to go out with him but don’t have a driving license. Michael decides to use the PowerEnJoy app. After he has reserved a car and that the specified time has arrived, Michael, Karen and Robert get in the car. The system detects via sensors that there are 2 passengers in the car and informs Michael through notification one his phone that he will receive a 10% discount on the last ride.

[SCENARIO 8] – *Michael gets a battery discount*

Michael is driving on board of a PowerEnJoy car. He will soon arrive at destination. Once he has parked in a safe area, he realises that the battery has more than 50% charge. A few moments later a notification arrives on his phone, informing him that he will receive a 30% discount on the last ride due to the battery charge.

[SCENARIO 9] – *Michael gets a penalty*

After a few weeks of using PowerEnJoy, Michael feels more confident with the application. He lets his guard down and carelessly uses a lot of battery charge. He also drives far away from the power grid stations and parks the car. Once he has parked the car in a safe area a notification reaches his phone, and informs that he will get a penalty for leaving the car with less charge than 20% or leaving the car at least 3 KM from the nearest power grid.

[SCENARIO 10] – *Michael uses the money saving option*

Michael has learnt his lesson the hard way receiving a penalty and decides he has to save money on the next rides. He searches for options to do so and realises he had overlooked a functionality called “money saving option”. By choosing it he has to insert the final destination. The system will then process the information and suggest where to leave the car to get a discount.

[SCENARIO 11] – *A field operator charges a car*

A field operator has got the request from an operator to travel to a PowerEnJoy car which has a nearly empty battery. He drives to the car with the equipment needed to charge the car manually. After finishing his duty, he informs the operator that he has completed the task. After this the car becomes available again.

5. UML MODELS

5.1 Use Case Diagram

We can derive some use cases from the scenarios identified in the previous paragraph:

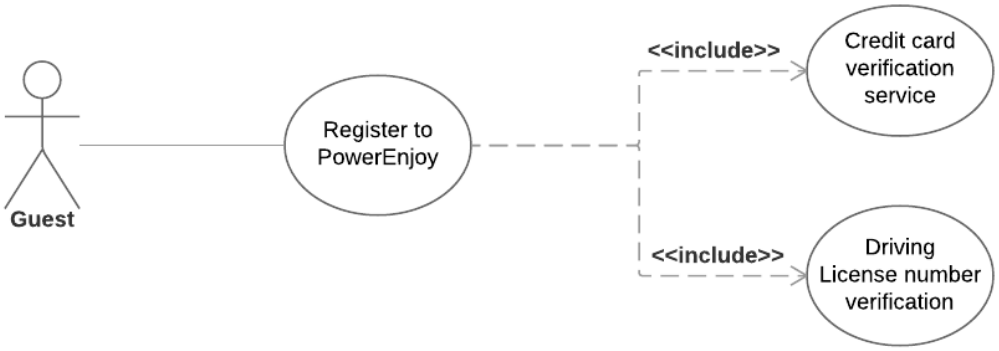
- Register
- Login
- Visualize profile
- Modify personal information
- Search for a car
- Reserve a car
- Pick up a car
- Cancel reservation
- Pending State Option
- Money saving option
- Payment
- Field Operator Work

5.2 Use Case Description

A detailed description of the provided use cases will be given.

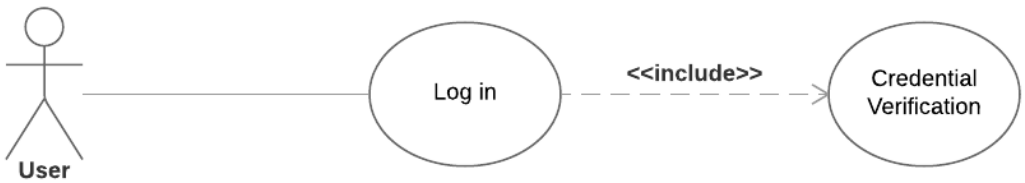
5.2.1 Registration

Name	Registration
Code	UC001
Actors	Guest
Entry conditions	Guest on the first page ready to register to the system.
Flow of events	The guest opens the application and selects the “Register” option. The guest fills in the form where he has to insert: – Name – Surname – E-mail – Date of birth – Driving license information – Credit card information When the guest completes the registration an email with a password is sent to him. The login page is shown.
Exit conditions	Successful registration.
Exceptions	– One or more fields aren’t filled – E-mail is not valid – Credit card or driving license number are not valid



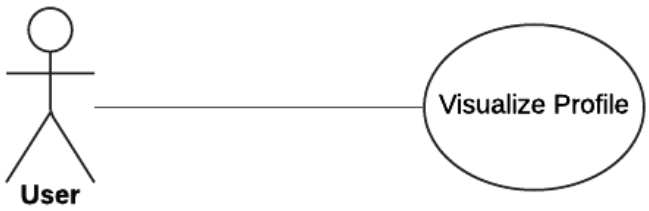
5.2.2 Login

Name	Login
Code	UC002
Actors	User
Entry conditions	User would like to login.
Flow of events	The user opens the application and inserts his/her email and password. Then selects “Login”.
Exit conditions	The system shows the user’s profile.
Exceptions	Username and/or password don’t match. *An error message is shown, along with the first page of the application, to restart.*



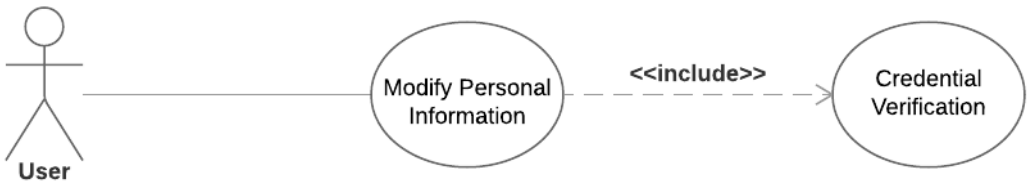
5.2.3 Visualize Profile

Name	Visualize profile
Code	UC003
Actors	User
Entry conditions	• Logged in correctly • Selected “Profile”
Flow of events	Just visual. No actions required. The system shows the user his/her profile.
Exit conditions	Another option is chosen.
Exceptions	No exceptions.



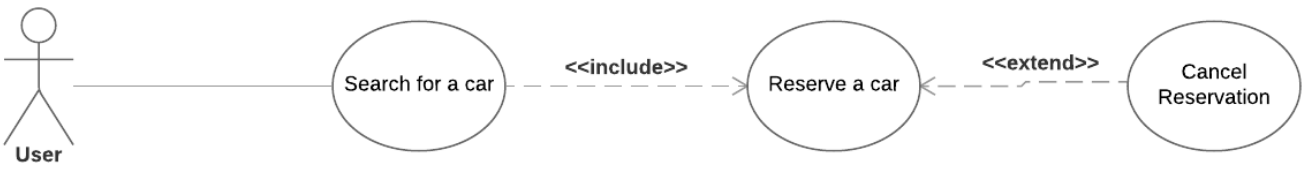
5.2.4 Modify personal information

Name	Modify personal information
Code	UC004
Actors	User
Entry conditions	User must be logged in and on his/her profile page.
Flow of events	User selects “Modify personal information”. The system then shows him a form to complete with: – email, – credit card information – driving license information When finished, the user can select “save changes” to conclude the operation, or cancel, to go bck. The changed profile is shown.
Exit conditions	“Save changes” option has been selected.
Exceptions	– One or more fields aren’t filled – Credit card or driving license number aren’t valid



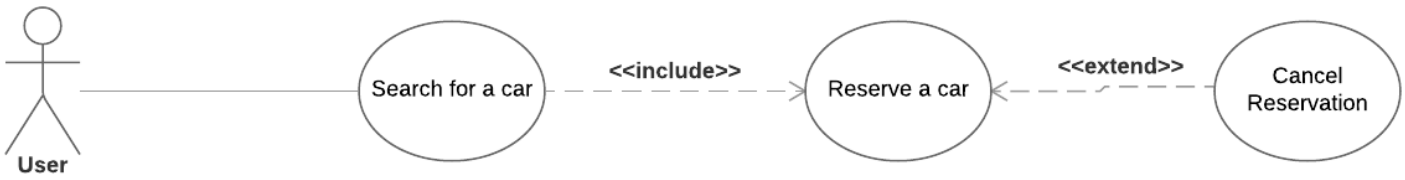
5.2.5 Search

Name	Search for a car
Code	UC005
Goal	G.1
Actors	User
Entry conditions	Used logged in and has selected “Search” option
Flow of events	The user selects the radius from which to search for cars. Selects the location from where to search. The system processes the input given and retrieves a list of available cars for the user to select.
Exit conditions	– Application is closed – User chooses other options – User finishes the operation
Exceptions	No exception.



5.2.6 Reserve

Name	Reserve car
Code	UC006
Goal	G.2
Actors	User
Entry conditions	User must have just searched for a car.
Flow of events	Once the user has decided the car to reserve he can select it to reserve it. He must select the time he would like to reserve it for. Once the operation is completed a reservation code is sent to the user email account. He can visualize it when he desires.
Exit conditions	– Application is closed – User chooses other options – User finishes the operation
Exceptions	The user has already reserved a car. Only one reservation at a time is permitted



5.2.7 Pick up a car

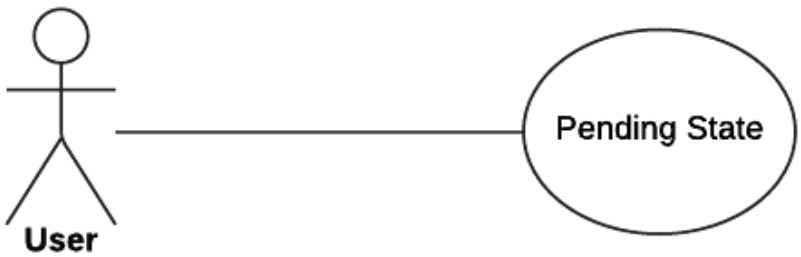
Name	Pick up a car
Code	UC007
Goal	G.2
Actors	User
Entry conditions	The user is located next to the car he reserved and he has his reservation code, which can be found in his profile.
Flow of events	The user inserts the reservation code into the car. If it's correct the system unlocks the car and he can enter.
Exit conditions	Code is correct
Exceptions	– Insertion keypad damaged/malfunctioning

5.3.8 Cancel

Name	Cancel reservation
Code	UC008
Goal	G.2
Actors	User
Entry conditions	User logged in and on his/her profile page
Flow of events	User can select the reservation he has and choose “Cancel reservation” to cancel it. The user can cancel a reservation up to 10 minutes before the reservation time.
Exit conditions	– Application is closed – User chooses other options – User finishes the operation
Exceptions	No exceptions

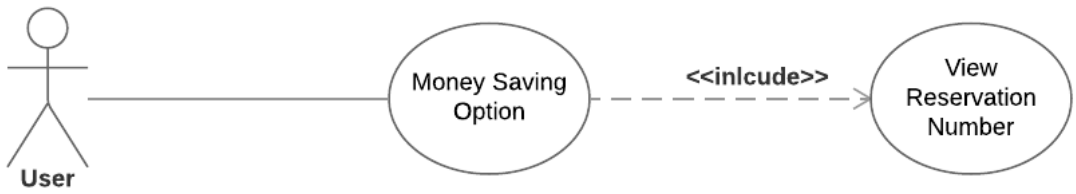
5.2.9 Pending

Name	Pending State Option
Code	UC009
Goal	G.2
Actors	User
Entry conditions	User must have parked the car and selected “Pending” on the reservation page.
Flow of events	Once he leaves the vehicle, the car will go in a “Pending” state. In this state the fee will be reduced by 30%. Max pending time is 2 hours
Exit conditions	User turns on the engine, or deactivates the option from the application.
Exceptions	No exceptions.



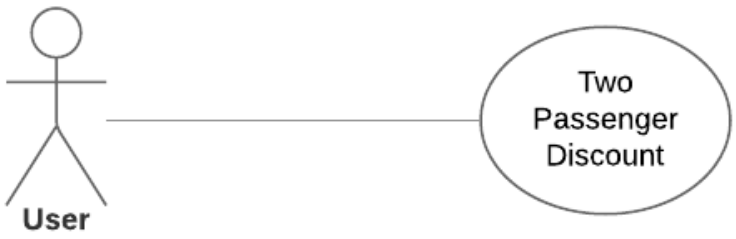
5.2.10 Money saving option

Name	Money saving option
Code	UC010
Goal	G.3, G.5
Actors	User
Entry conditions	User logged in and has selected “Money saving option” before starting the engine.
Flow of events	User can choose a destination. The system will process the input and retrieve a location of where to leave the car once finished using it. This will provide a discount on the last ride.
Exit conditions	– Application is closed – User chooses other options – User finishes the operation
Exceptions	No exceptions.

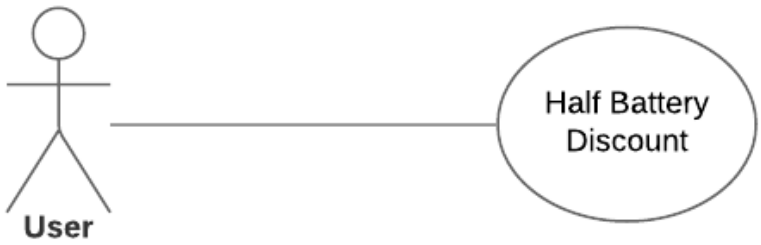


5.2.11 Payment

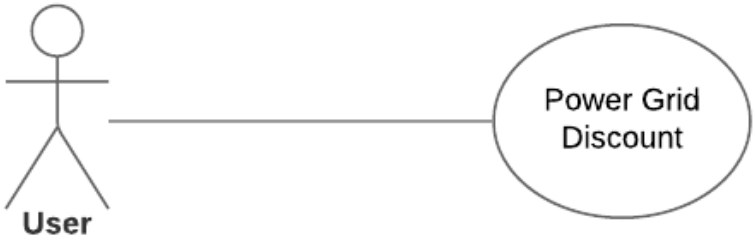
Name	Two Passenger Discount
Code	UC011
Actors	User
Goal	G.4, G.5
Entry conditions	User has started and finished their last ride with at least two passengers other than him/her.
Flow of events	The car sensor detects the number of passengers on the car during the whole ride. Once the car has been parked in the safe area a notification is sent to the user. He is informed that he will get a 10% discount on his last ride due to the “Two Passenger Discount”.
Exit conditions	Just visual. No actions required
Exceptions	If the passengers exits the vehicle and for a part of the ride there are less than two



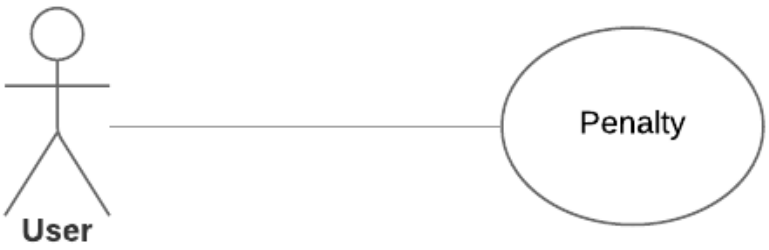
Name	Half Battery Discount
Code	UC012
Actors	User
Goal	G.4, G.5
Entry conditions	User has finished ride with at least 50% battery left.
Flow of events	Once the car has been parked in the safe area a notification is sent to the user. He is informed that he will get a 20% discount on his last ride due to the “Half Battery Discount”.
Exit conditions	Just visual. No actions required.
Exceptions	No exceptions.



Name	Power Grid Discount
Code	UC013
Goal	G.4, G.5
Actors	User
Entry conditions	User has parked the car in a special parking area
Flow of events	Once the car has been parked in the safe area a notification is sent to the user. He is informed that he will get a 30% discount on his last ride due to the “Power Grid Discount”
Exit conditions	Just visual. No actions required
Exceptions	No exceptions.

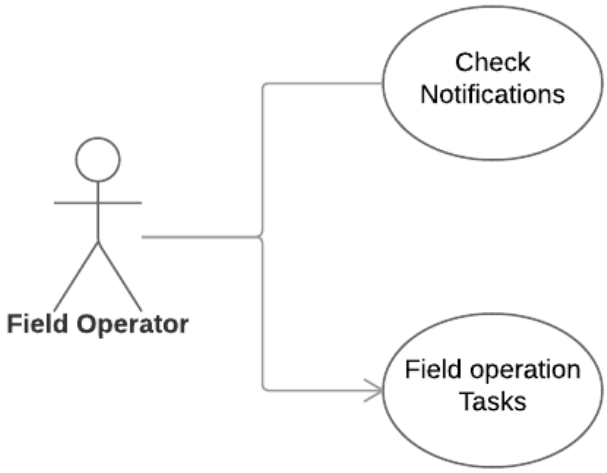


Name	Penalty
Code	UC014
Actors	User
Goal	G.4, G.5
Entry conditions	User has parked car at least 3 KM away from power grids, or has left the car with only 20% battery remaining
Flow of events	Once the car has been parked in the safe area a notification is sent to the user. He is informed that he will get a 30% increase of price on his last ride due to the “Penalty”
Exit conditions	Just visual. No actions required
Exceptions	No exceptions.



5.2.12 Field Operator Work

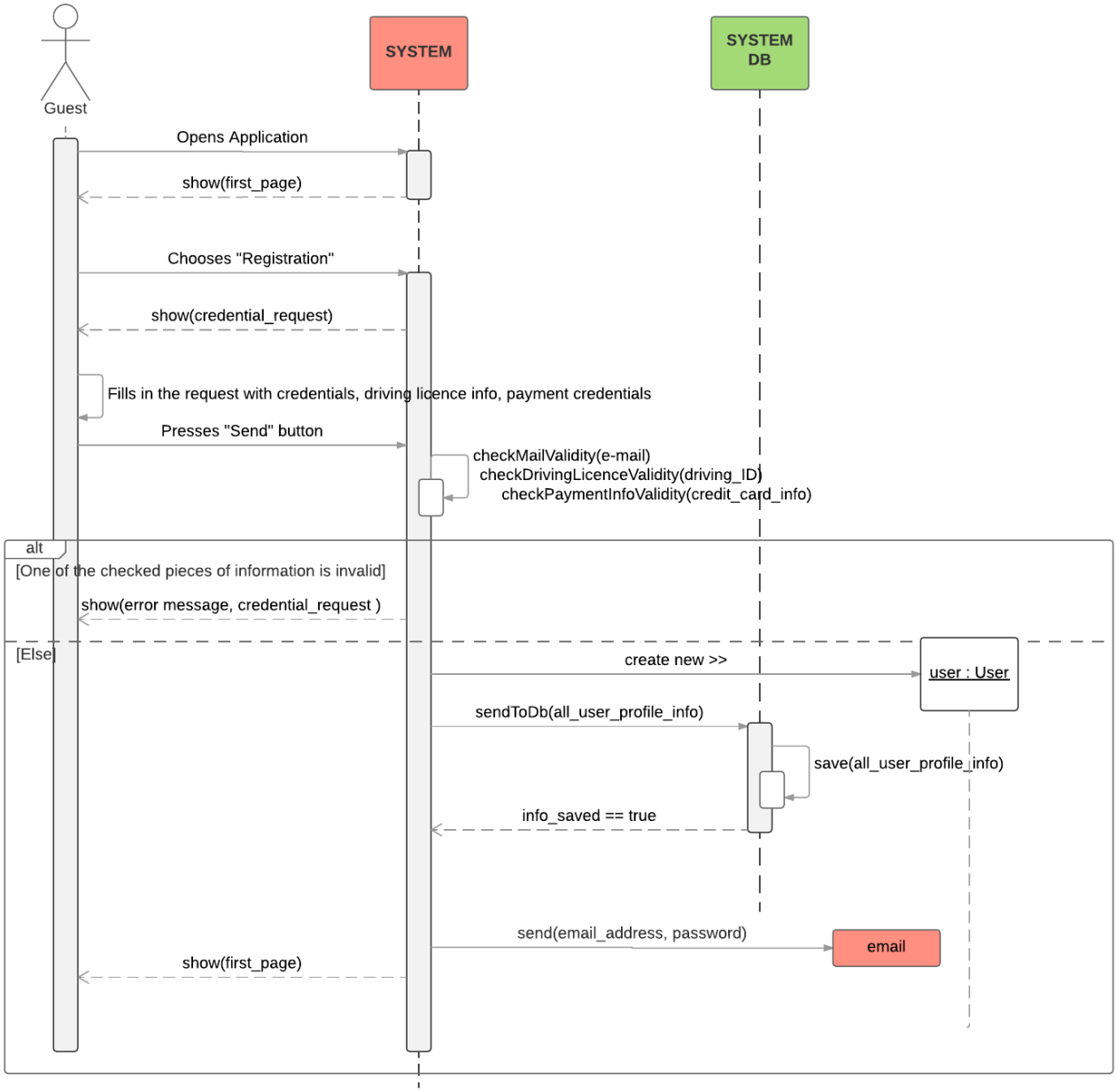
Name	Field Operator Work
Code	UC015
Goal	G.6
Actors	Field Operator
Entry conditions	Notification has arrived on his profile.
Flow of events	The operator checks the notification. He gathers information about car location, type of work. He then travels to the location with the right equipment and does his work
Exit conditions	Finishes his work
Exceptions	No exceptions.



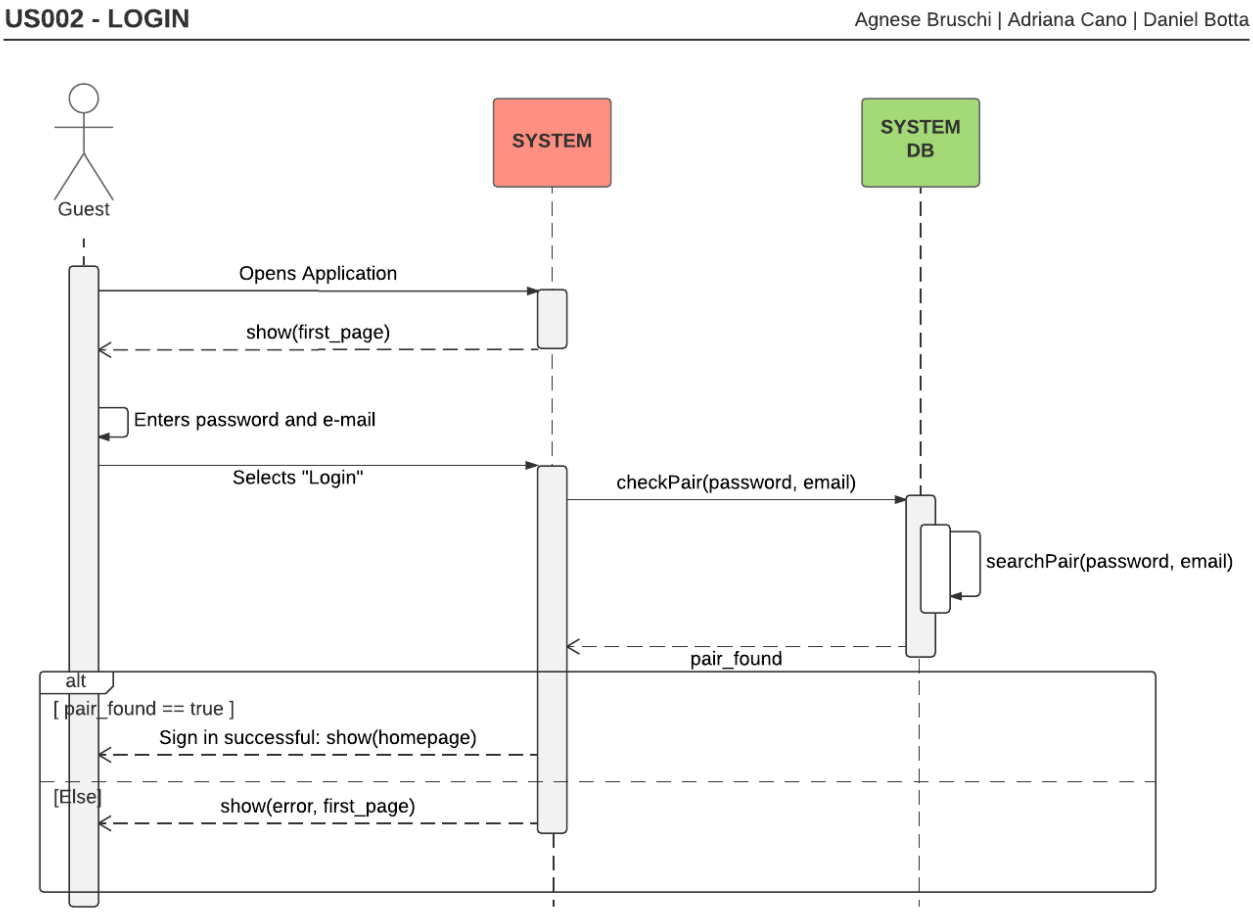
5.3 Use case sequence diagram

5.3.1 Registration

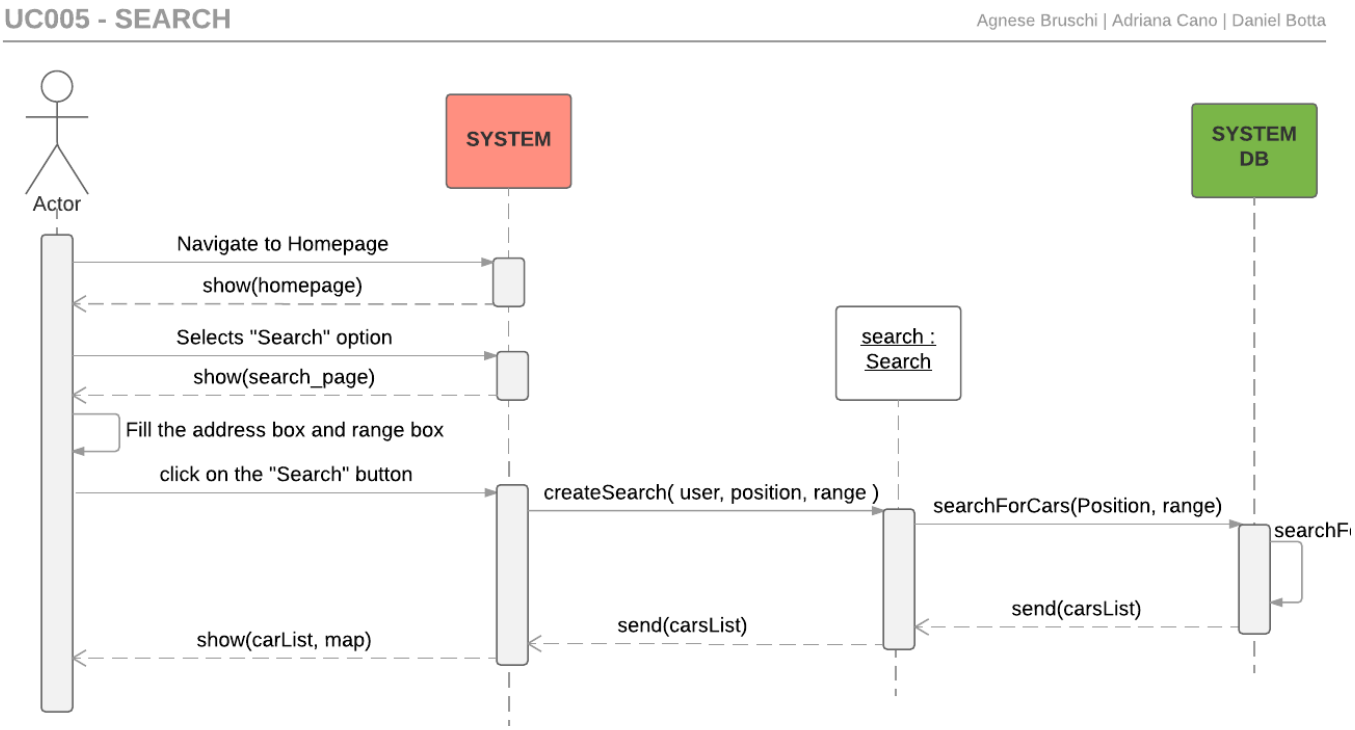
UC001 - REGISTRATION Agnese Bruschi | Adriana Cano | Daniel Botta

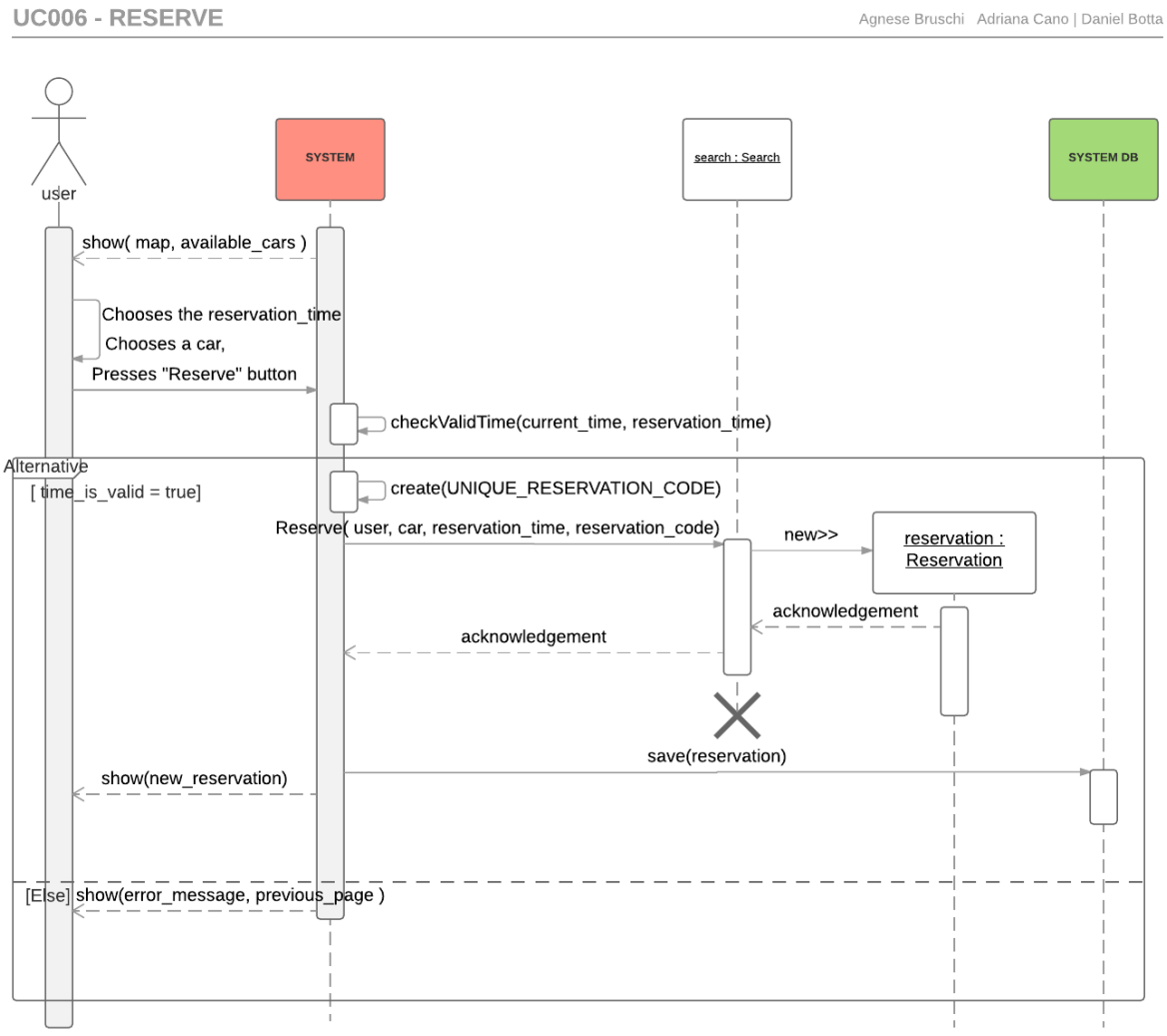


5.3.2 Login

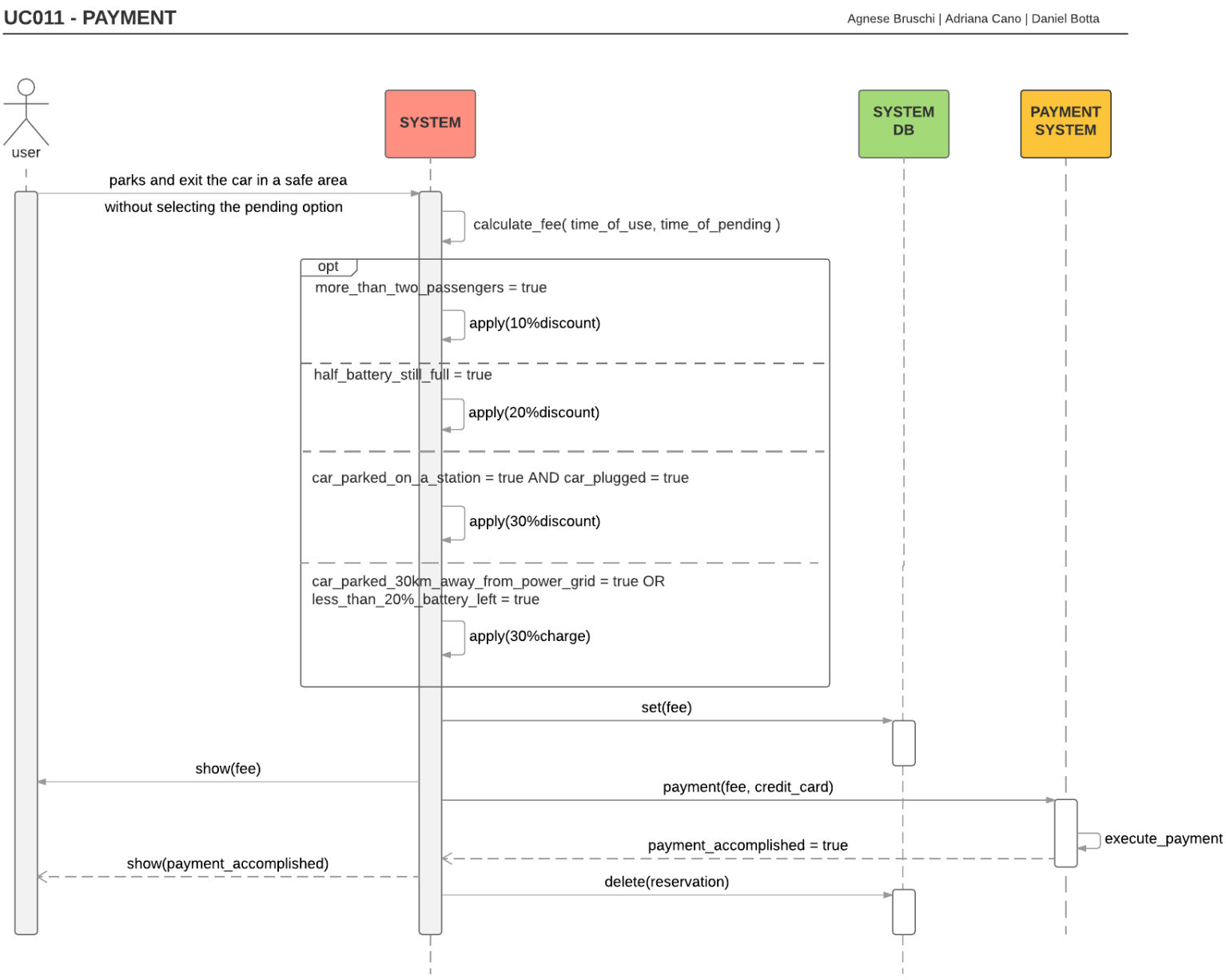

US002 - LOGIN
Agnese Bruschi | Adriana Cano | Daniel Botta
Guest
SYSTEM
SYSTEM DB
Opens Application
show(first_page)
Enters password and e-mail
Selects "Login"
checkPair(password, email)
searchPair(password, email)
pair_found
alt
[pair_found == true]
Sign in successful: show(homepage)
[Else]
show(error, first_page)

5.3.3 Search and Reserve





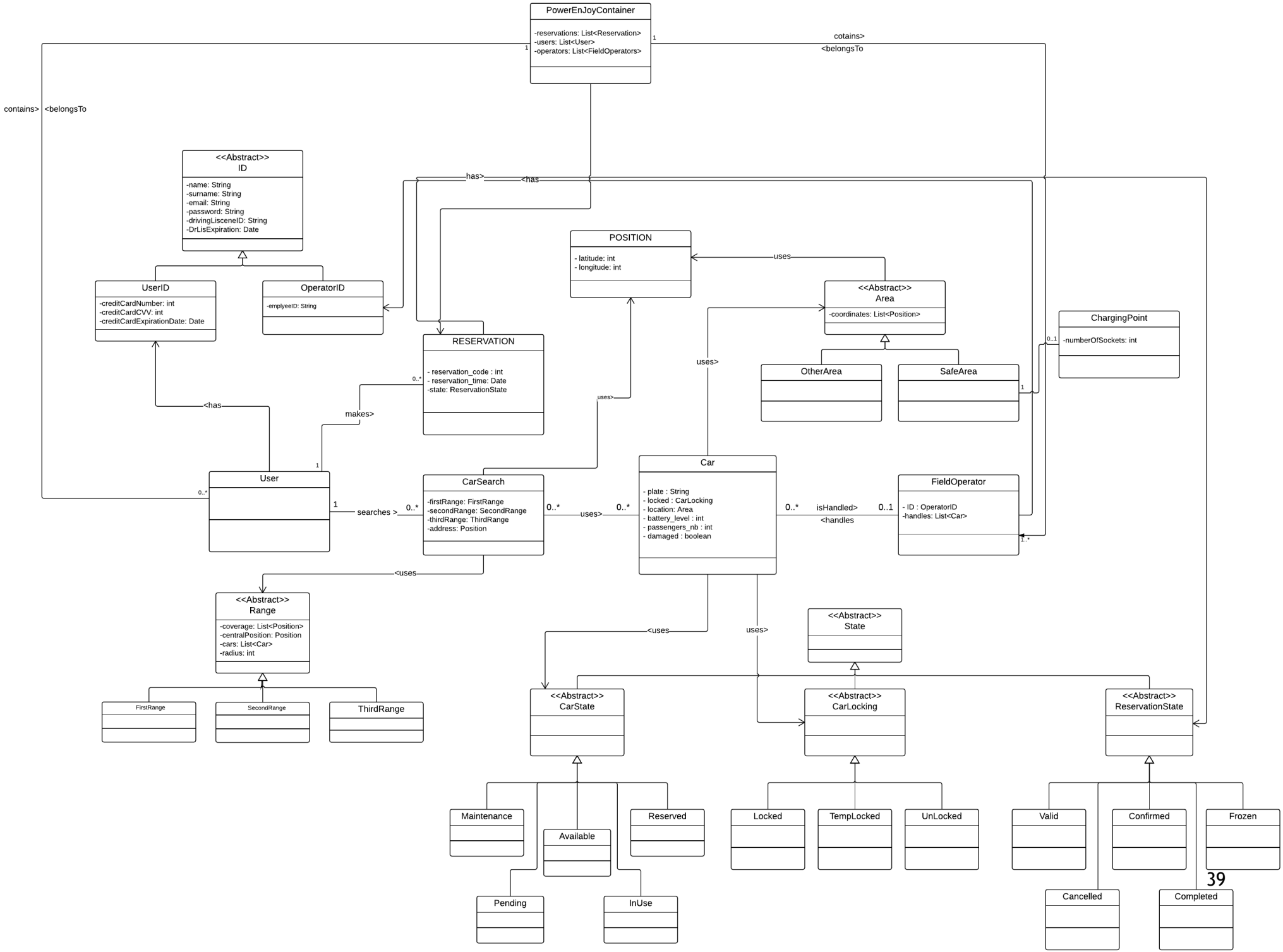
5.3.4 Payment



5.4 Class model

The class model is designed to represent the objects and people interacting in the system and their specific interaction.

It is also detailed by an alloy specification [view chapter 6].


PowerEnJoyContainer
-reservations: List<Reservation>
-users: List<User>
-operators: List<FieldOperators>
1
1
cotains>
<belongsTo
contains>
<belongsTo
<<Abstract>>
ID
-name: String
-surname: String
-email: String
-password: String
-drivingLisceneID: String
-DrLisExpiration: Date
has>
<has
UserID
-creditCardNumber: int
-creditCardCVV: int
-creditCardExpirationDate: Date
OperatorID
-emplyeeID: String
POSITION
- latitude: int
- longitude: int
uses>
<<Abstract>>
Area
-coordinates: List<Position>
ChargingPoint
-numberOfSockets: int
0..1
RESERVATION
- reservation_code : int
- reservation_time: Date
-state: ReservationState
0..*
makes>
<has
1
uses>
OtherArea
SafeArea
1
User
0..*
1
searches >
0..*
CarSearch
-firstRange: FirstRange
-secondRange: SecondRange
-thirdRange: ThirdRange
-address: Position
0..*
uses>
0..*
Car
- plate : String
- locked : CarLocking
- location: Area
- battery_level : int
- passengers_nb : int
- damaged : boolean
0..*
isHandled>
<handles
0..1
FieldOperator
- ID : OperatorID
-handles: List<Car>
1..*
<uses
<<Abstract>>
Range
-coverage: List<Position>
-centralPosition: Position
-cars: List<Car>
-radius: int
FirstRange
SecondRange
ThirdRange
<uses
uses>
<<Abstract>>
State
<<Abstract>>
CarState
<<Abstract>>
CarLocking
<<Abstract>>
ReservationState
Maintenance
Available
Reserved
Pending
InUse
Locked
TempLocked
UnLocked
Valid
Confirmed
Frozen
Cancelled
Completed

6. Alloy model

In the next paragraph we will introduce the Alloy model written for our application in order to check the integrity of elements compounding our system and how they interact with one another.

```
*******************************************************************************

module powerEnJoy

/*****************************USER*****************************/
sig User{
	userID: UserID,
	searches: set CarSearch,
	reserves: set Reservation
}{
	#reserves <= #searches
}

abstract sig ID {}

sig UserID extends ID{}
sig OperatorID extends ID{}


/***************************CAR*************************************/

sig Car{
	plate: Int,
	state: CarState,
	locked: CarLocking,
	location: Area
}{
	plate>0
}

abstract sig State{}
abstract sig CarState extends State{}

sig Available extends CarState{}
sig Reserved extends CarState{}
sig Maintenance extends CarState{}
sig InUse extends CarState{}
sig Pending extends CarState{}

abstract sig CarLocking extends State {}

sig Locked extends CarLocking{}
sig UnLocked extends CarLocking{}
```

```
sig TempLocked extends CarLocking{}

sig ChargingPoint{}{
	ChargingPoint=SafeArea.chargingPoint
}

abstract sig Area {
	coordinates: some Position
}

sig OtherArea extends Area{}
sig SafeArea extends Area{
	chargingPoint: lone ChargingPoint
}


/**********************Reservation**********************************/

sig Reservation{
	resID: Int,
	state: ReservationState,
	car: Car

}{
		resID>0
}

abstract sig ReservationState extends State{}

sig Valid extends ReservationState{}
sig Confirmed extends ReservationState{}
sig Frozen extends ReservationState{}
sig Completed extends ReservationState{}
sig Cancelled extends ReservationState{}


/**************************CarSearch*****************************/

sig CarSearch{
	firstRange: FirstRange,
	secondRange: SecondRange,
	thirdRange: ThirdRange,
	address: Position,
	reserve: lone Reservation
}

sig Position{}

abstract sig Range{
	coverage: some Position,
```

```
        centralPosition: Position,
        cars: set Car,
        radius:  one Int
}

sig FirstRange extends Range{
}{
        int radius=1}

sig SecondRange extends Range{
}{
        int radius=2}
sig ThirdRange extends Range{
}{
        int radius=4}

/*****************************FieldOperator************************/
sig FieldOperator{
        operatorID: OperatorID,
        handling: lone Car
}

/*******************************FACTS*****************************/
fact userProperties{

        //Unique ID for every user
        all disjoint u1, u2: User | u1.userID!=u2.userID

        //A user can't have two different reservations going on at the same time
        all u: User {no disjoint r1,r2: Reservation| hasReserved[u, r1] &&
        hasReserved[u, r2] && r1.state in Valid + Confirmed && r2.state in Valid
        +Confirm}
}

fact uniquePlates{
        no disj c1,c2: Car | c1.plate=c2.plate
}

fact ids{
        no disjoint fo1,fo2: FieldOperator | fo1.operatorID=fo2.operatorID
}

fact locations{
        //Two cars can not be in the same area at the same time
        no disj c1,c2: Car | c1.location=c2.location

        //The cars in a search result are all centered in the requested address
        all s: CarSearch | s.address=s.firstRange.centralPosition +
        s.secondRange.centralPosition +s.thirdRange.centralPosition
```

```
        all disjoint s1,s2: SafeArea{no cp: ChargingPoint | cp in s1.chargingPoint && cp
        in s2.chargingPoint}

        all disjoint a1,a2: Area {no p1: Position | p1 in a1.coordinates && p1 in
        a2.coordinates}
}

fact reservationProperties{

        //Unique reservation number for each reservation
        no disjoint r1,r2: Reservation | r1.resID=r2.resID

        //For every reservation there exists a search that led to it
        all r: Reservation {one s: CarSearch |  searchTurnedReservation[s,r]}

        //The search and the reservation in a association have the same qualities
        all r: Reservation{one s: CarSearch| one u: User |hasReserved[u,r] &&
        hasSearched[u,s] &&  carBelongsToSearch[r.car, s] }


        //A reservation is done only by one user
        all r: Reservation {one u: User | hasReserved[u,r]}
}

fact reservationStatePropertie{

        //A reservation is valid the instance after is being created until the car is
        unlocked by the reservee or until the reservation is cancelled
        all r: Reservation | r.state in Valid <=> r.car.state in Reserved && r.car.locked in
        Locked && r.car.location in SafeArea

        //A reservation is Confirmed the moment the reservee unlocks said car
        all r: Reservation | r.state in Confirmed <=> r.car.state in InUse && r.car.locked
        in UnLocked && r.car.location in OtherArea

        //After a user is done with the car, the car can be in two types of conditions: Low
        battery, with mainatance problems so in Maintance
        // or the user has a stellar behavior and the car is Available for yet another
        reservation
        all r: Reservation | r.state in Completed <=> r.car.state in Maintenance +
        Available  && r.car.locked in Locked && r.car.location in SafeArea

        //A reservation is Canceled if the user doesn't pick up the car in time or if the
        user himself/herself canceles the reservation
        all r: Reservation | r.state in Cancelled <=> r.car.state in Available &&
        r.car.locked in Locked && r.car.location in SafeArea

        //A reservation is Frozen if the user decides to make a stop but is not done with
        his/her reservation
```

```
        all r: Reservation | r.state in Frozen <=> r.car.state in Pending && r.car.locked in
        TempLocked && r.car.location in SafeArea + OtherArea
}

fact generalFacts{
        Car= Reservation.car +FirstRange.cars + SecondRange.cars +ThirdRange.cars
        ReservationState= Reservation.state
        Area=Car.location
        CarState=Car.state
        CarLocking=Car.locked
        CarSearch=User.searches
        Reservation=User.reserves
        FirstRange=CarSearch.firstRange
        SecondRange=CarSearch.secondRange
        ThirdRange=CarSearch.thirdRange
        UserID=User.userID
        OperatorID=FieldOperator.operatorID
        FirstRange.coverage in SecondRange.coverage
        SecondRange.coverage in ThirdRange.coverage
        Position= FirstRange.coverage + SecondRange.coverage + ThirdRange.coverage
+ Area.coordinates
        CarState=Available + InUse + Pending + Reserved + Maintenance
        CarLocking=Locked + UnLocked + TempLocked
        ReservationState= Valid + Confirmed + Completed + Cancelled + Frozen
        ChargingPoint=SafeArea.chargingPoint

}

fact carStateProperties{

        //An available car is a car that appears on the searches the user can make and it is
        locked
        all c: Car{c.state in Available => {no r: Reservation | r.car=c }}
        all c: Car {c.state in Available =>c.locked in Locked
                && c in FirstRange.cars + SecondRange.cars + ThirdRange.cars
                && c.location in SafeArea}
        all c: Car{c.state in Available => no fo: FieldOperator | c in fo.handling}

        //A car is Reserved if there is a user that has reserves it. A reserved car is still
        Locked
        all c: Car { c.state in Reserved => {one r: Reservation|r.car=c && r.state in
        Valid }}
        all c: Car {c.state in Reserved =>c.locked in Locked && c.location in SafeArea}
        all c: Car{c.state in Reserved => no fo: FieldOperator | c in fo.handling}

        //A car is in use if there exists a confirmed reservation, the car in unlocked and
        not located in a SafeArea
        all c: Car { c.state in InUse <=> {one r: Reservation | r.car=c
                        && r.state in Confirmed && c.locked in UnLocked
                        && c.location in OtherArea  }}
```

```
	all c: Car{c.state in InUse => no fo: FieldOperator | c in fo.handling}

	//A car is Pending if the reservee makes a stop but the reservation is ongoing. He
	can park the car in a non Safe area.
	all c: Car{c.state in Pending <=>{one r: Reservation |  c.locked in TempLocked
					&& r.car=c && r.state in Frozen}}
	all c: Car{c.state in Pending => no fo: FieldOperator | c in fo.handling}

	//A car is said to be in maintanance if it is being handled by a FieldOperator or if
	it is charging
	all c: Car {c.state in Maintenance => {no r: Reservation |r.car=c  }}
		all c: Car {c.state in Maintenance => {some fo: FieldOperator | c in
	fo.handling } || #c.location.chargingPoint=1}

	all c: Car {c.state in Maintenance =>c.location in SafeArea && c.locked in
	UnLocked }

	all c: Car{c.state in Maintenance  => c not in  FirstRange.cars +
	SecondRange.cars + ThirdRange.cars  }

	all c: Car{c.state in Maintenance  => no fo: FieldOperator | c in fo.handling}
		all c: Car{c not in  FirstRange.cars + SecondRange.cars +
	ThirdRange.cars => c.state in Maintenance  + Reserved + InUse + Pending}
}

fact rangeProp{
	all s: CarSearch {all c: Car | c in s.firstRange.cars <=> c.location.coordinates in
	s.firstRange.coverage }

	all s: CarSearch {all c: Car | c in s.secondRange.cars <=> c.location.coordinates
	in s.secondRange.coverage }

	all s: CarSearch {all c: Car | c in s.thirdRange.cars <=> c.location.coordinates in
	s.thirdRange.coverage }
}



/**************************PREDICATES**************************/

pred hasReserved [u: User, r: Reservation]{
	r in u.reserves
}

pred hasSearched[u: User, s: CarSearch]{
	s in u.searches
}
pred searchTurnedReservation[s: CarSearch, r: Reservation]{
	r in s.reserve
}
```

```
pred carBelongsToSearch[c: Car, s: CarSearch]{
        c in s.firstRange.cars +s.secondRange.cars + s.thirdRange.cars
}

pred addUser[c,c': PowerEnJoyContainer, u: User]{
        c'.users=c.users + u
}

pred addOperator[c,c': PowerEnJoyContainer, o: FieldOperator]{
        c'.operators=c.operators + o
        c'.users=c.users
        c'.reservations=c.reservations
}
pred addReservation[c,c': PowerEnJoyContainer, u,u': User, r: Reservation]{
        r.state in Valid
        c'.users=c.users
        c'.reservations=c.reservations + r
        u'.reserves=u.reserves + r
}
assert reservationCheck{
        no r: Reservation {no s: CarSearch| searchTurnedReservation[s,r]}
        no r: Reservation {one c: Car | r.car=c && c.state in Maintenance}
        all r: Reservation {no c: Car | r.car=c && r.state in Completed && c.state not in
        Available}
        all r: Reservation {no c: Car | r.car=c && r.state in Cancelled && c.state not in
        Available}
        all r: Reservation {no c: Car | r.car=c && r.state in Confirmed && c.state not in
        InUse}
        all r: Reservation {no c: Car | r.car=c && r.state in Valid && c.state not in
        Reserved}
}

assert carCheck{
        no c: Car {one r: Reservation | c.location in SafeArea && r.car=c && r.state in
        Confirmed}
        no c: Car | c.state in Available && c.locked in UnLocked + TempLocked
        no c: Car | c.state in Reserved && c.locked in UnLocked + TempLocked
        no c: Car | c.state in InUse && c.locked in Locked + TempLocked
        no c: Car | c.state in Maintenance && c.locked in Locked + TempLocked
        no c: Car | c.state in Pending && c.locked in UnLocked + Locked
        no c: Car | c.location in SafeArea && c.locked in TempLocked
        no c: Car | c.location in OtherArea && c.locked in Locked
        no c: Car | c.state in Available + Reserved +  Maintenance && c.location in

OtherArea
        no c: Car | c.state in InUse + Pending && c.location in SafeArea
                no c: Car | c.state in Maintenance  && c in FirstRange.cars +
        SecondRange.cars + ThirdRange.cars
}
```

```
pred show{
        #User=2
        #User.searches=3
        #User.reserves=2

}

run show for 3
run addUser for 2
run addReservation for 3
run removeUser
run removeReservation
check reservationCheck for 3
check carCheck for 3
```

****

We can see, in the next page [Figure 6.A] the first world generated by the alloy representation of the system.

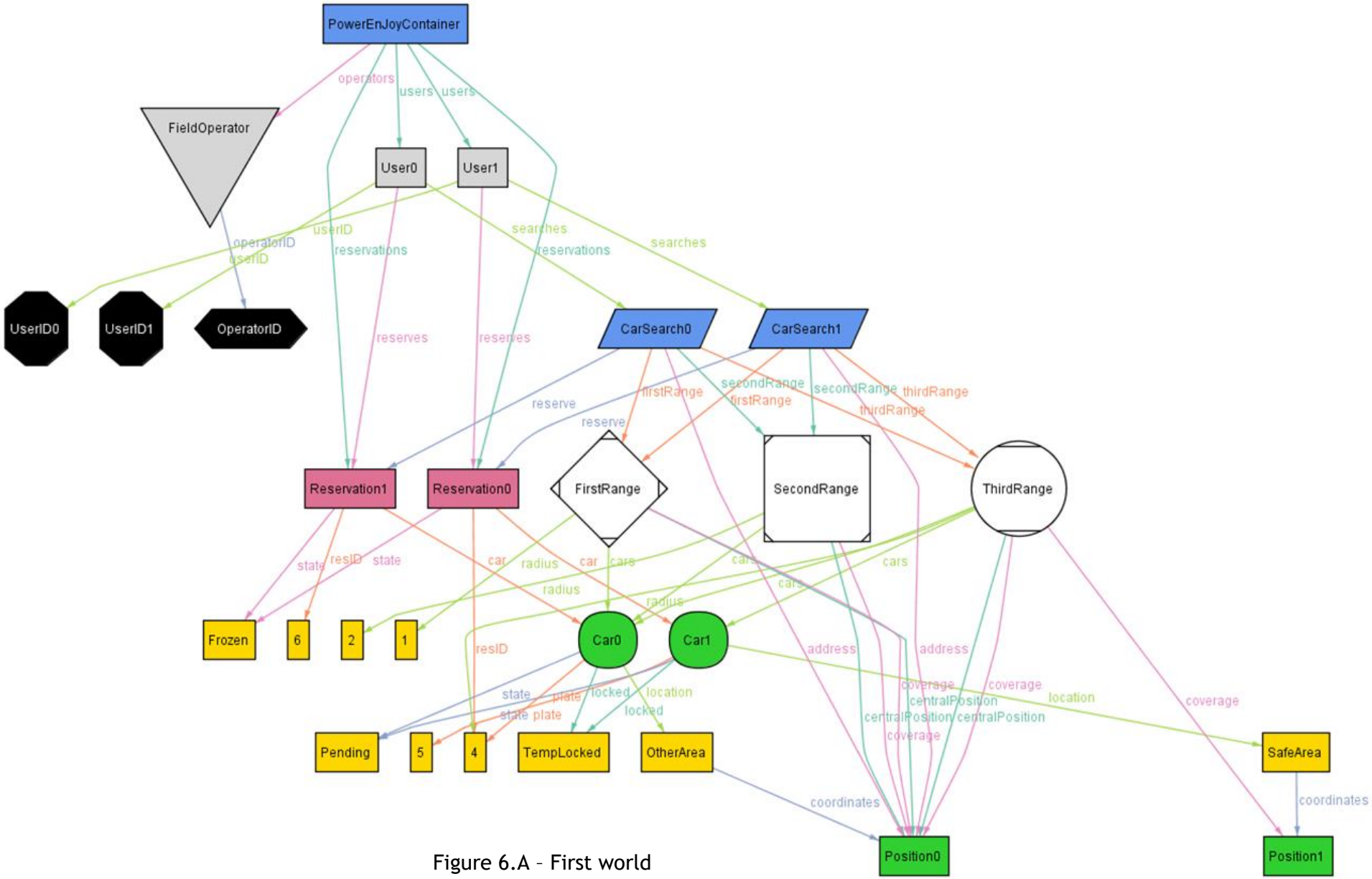


Figure 6.A – First world

6.1 Alloy analysis

In the first world generated we can see that the system is structured as follows, following a top-down approach while reading it:

1. We have a container where all the users, operators and reservations are stored. This is the PowerEnJoyContainer.

2. Every user has a unique identification. The same applies for the Operators.

3. A user can make as many searches as he pleases. In this world both users have done only one search.

   For each search we have the address the user inserted when making the search. Based on this address and the availability of the cars offered by PowerEnJoy, three different ranges centered in said address are shown to the user.

   In each range there are shown the available cars from which the user can choose from.

4. For the reservations there are some conditions that need to be verified:

   – There needs to be a search before a making a reservation.

   – After the search is done, there needs to be at least a car in the result of the search for the user to make a reservation, which can be in any of the three ranges shown by the system.

     In this world we have that CarSearch0 has Car0 and Car1 in the result. We then can see that Reservation0 is associated with the Car1 and Reservation1 with Car1.

   – Each reservation has a corresponding state.

     In this case we see that both reservations are Frozen which means that the user has made a stop but is not done using the car yet.

5. For each car we have their location which is expressed through the Area where the car occupies.

   Car0 is in a not Safe Area while Car1 is in a Safe Area.

Both cars are in a Pending state due to the fact that the car is stopped but the user is not done using them yet. As far as the locking of the car goes, being in a pending state, both cars are temporarily locked.

6. For area we can see the positions that define it.

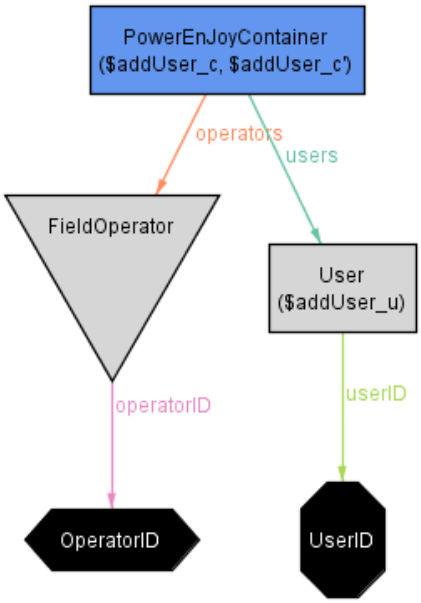


Figure 6.B – Second world

In this world generated we can see the operation of adding a User to our container.

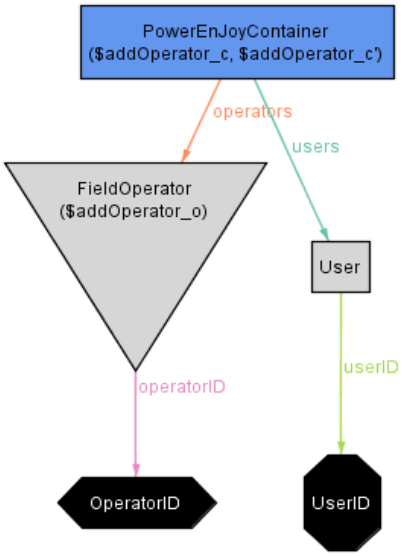


Figure 6.C – Third world

In this world we can see an operator being added to our container.

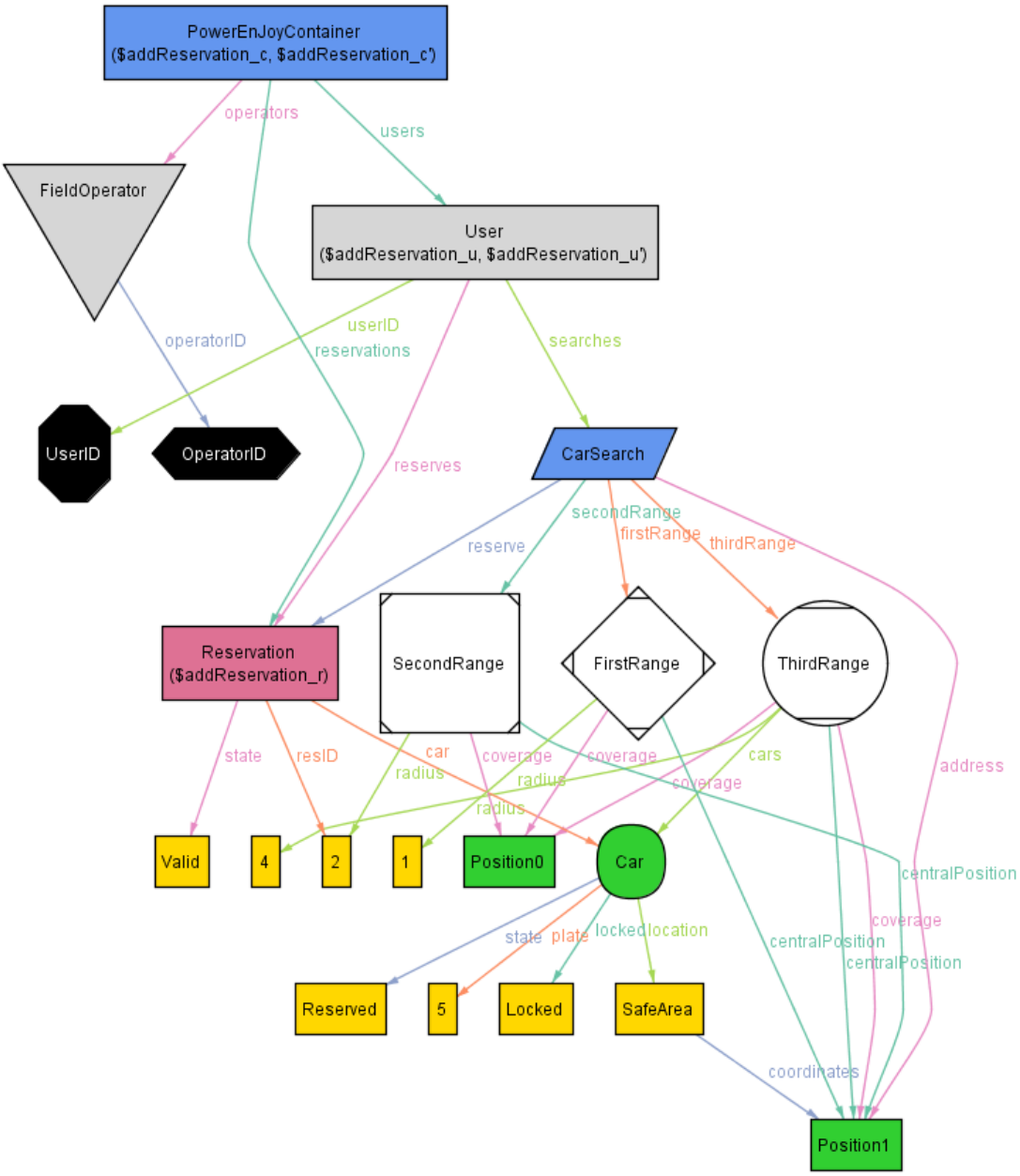


Figure 6.D – Fourth world

In the fourth world we can see a reservation being added to our container.

The model is consistent:

6 commands were executed. The results are:
#1: **Instance found.** show is consistent.
#2: **Instance found.** addUser is consistent.
#3: **Instance found.** addReservation is consistent.
#4: **Instance found.** addOperator is consistent.
#5: No counterexample found. reservationCheck may be valid.
#6: No counterexample found. carCheck may be valid.

Work division details:

Every team member had tasks to complete. At the same time, subjects such as requirements, goals, domain properties and assumptions and a general overview of the project were treated by the whole group. Also, the group supervised every task, so that everyone participated in all the important decisions.

Group:

- Project analysis and discussion	5h
- Domain Properties	2h
- Assumptions	5h
- Requirements	5h

Adriana Cano:

- Alloy	30h
- UML	4h

Agnese Bruschi:

- Layout	8h
- UML	4h
- User Interface	3h
- Sequence diagrams	15h

Daniel Enrico Botta:

- Scenarios	5h
- Use Cases and description	15h

Tools used:

- Word: layout
- Google drive: to share material and work simultaneously on the same part of the project
- Lucidchart: diagrams, images and UML
- Alloy