**TO DO LIST**

* Aggiungere scenari:
  + in cui si specifica una posizione diversa da quella attuale dell’utente,
  + scenario intervento cars hub controller

**ANGELO**

* Intro (?)
* Mock up

**GABRIELE**

* UML (statechart se serve)

**MARCO**

* Alloy

La parte in cui vengono spiegati a grandi linee I ‘implementazione hardware e software penso si possa lasciare a dopo che avremo fatto il design document

**ASSUMPTION**

**ASSUMPTION: The only payment methods accepted are credit card (payment guaranteed by the bank)**

(GLOSSARY:

“Cost of the trip” is the raw price of the service calculated only on the base of the duration of the car’s usage, before discounts or additional charges are applied

"Virtuousness coefficient" is the factor by which to multiply the cost of the trip to get the amount of the bill. Its initial value is 1)

**ASSUMPTION: The system continues counting the minutes of usage while the car is outside the safe areas, also if it’s turned off**

**ASSUMPTION: A user uses a car only if the power remaining in the car’s battery is sufficient for his trip**

**DOMAIN PROPERTIES: There is a CAR HUB CONTROLLER where a “supervisor” can monitor the status of every car and dispatch the “recharge on site” if a car is left with less than 20% battery life remaining and it isn’t connected to a power grid. The supervisor is also responsible to dispatch the “car recovery” when a car is left outside a safe area for more than 5 hours**

(GLOSSARY:

“Supervisor” is a company employee which work at the Car hub controller

“recharge on site” is a company procedure: a worker is sent to recharge a low car that was parked detached from the power grid

“Car recovery” is a company procedure: a worker is sent to retrieve a car that has been forgotten outside a safe area and park in one of these)

**DOMAIN PROPERTIES: The cars can be recharged by the users only in the special parking areas with power grid**

**ASSUMPTION: Special parking areas with power grid are a subset of the predefined set of the safe areas**

**ASSUMPTION: When a user get into a car, he actually starts using it**

**ASSUMPTION: a user finishes using the car when he leaves the car in a safe area**

**ASSUMPTION: users are not permitted to delete a reservation**

**ASSUMPTION: when users get in or exit a car, they close the car’s door**

**ASSUMPTION: users do not leave the car without turning it off**

**ASSUMPTION: The passengers counted to obtain the 10% discount are them who are present in the car at the time of the first ignition**

(GLOSSARY:

“Guest” is a person which is not already registered to the system

“User” is a registered customer)

**SCENARI**

Scenario 1

Title: Deal with the strike

Luca should go to class this afternoon, but unfortunately, today there is a strike of transport. His university is on the opposite part of the city, so he decides to try the new car-sharing service “PowerEnJoy”. Since it’s the first time he tries the service, first he should download the app and register to the system; after filling in his own personal data, including the payment information, he clicks on the submit button and, after a few seconds, he receives a message with his password. Now he can start looking for a car near to him.

Scenario 2

Title: Friends in saving

Marco, an expert user of “PowerEnJoy”, has gone to see a concert with his roommates Mario and Matteo, and now they want to come back home. Due to the late hour, the public transportation is no more available and, as the evening was rather expensive, they aim to spend as little as possible. Marco decides to use the “PowerEnJoy” service and, after he found and reserved a car near to them, he checks the “Money saving” option. When they get into the car, they set up their destination and the system calculates the most convenient place to leave the car. When they arrive at their destination, they will have to walk a bit, but they will have saved a lot.

Scenario 3

Title: A busy businessman

William is a businessman always in a hurry; he has just arrived to his office, but he already knows that, as soon as he will finish the morning’s meeting, he will have to go to the opposite part of the city for urgent commitments. William saw a “PowerEnJoy” car parked at a few meters from his office, so he thinks that he can save time using the car-sharing service instead of wait for a taxi after the meeting. William then register quickly to the service without paying much attention to all warnings and book the car.

Unfortunately, the meeting dwells and William’s reservation expires and the system charges him 1€; when he leaves the office, he cannot get in the car because it was booked by another user, forcing William to call a taxi.

Scenario 4

Title: Desperate housewife

Laura went to the grocery store on foot, but when she exits the supermarket she realizes that it starts raining; she notices a “PowerEnJoy” car parked and, since his son has already registered her to the service to encourage her to use it, she decides to book the car to come back home without getting wet.

Once she arrives at destination, the car is low, but Laura’s first problem is to not get wet, so she looks for a park in a safe area as close as possible to her house, and she doesn’t mind the warning concerning the fact she will pay more if she won’t leave the car in a recharge park.

Scenario 5

Title: A Long Road

Yuri recently moved to a new built house so he needs some furniture.

The only store he knows is an Ikea so he decides to go there, also if it is quite far. Having in mind to buy some initial stuff like pillow and sheets, he knows that he won’t use the *Home Delivery*, as it would turn out in an extremely high fee for such small items. He thinks then to use a “PowerEnjoy”’s car.

After the registration, he finds out that there’s a car near his house, so he decides to reserve it. Then, he goes near the car and clicks on the “unlock” button in the “PowerEnJoy” app on his smartphone, he enters and starts driving. When he arrives at the IKEA, he finds out that there is no safe area where to park the car! Now he knows that he must do his errand quickly, because he will pay every minute for the car also if it’s switched off, and after 5 hours his reservation will expire!

**GOALS**

G1: Ensure system’s accessibility

Requirements:

* The system must prevent guests from accessing any service before being registered or logged in
* The system must recognize already registered user
* The system must allow new user’s registration
* The system must allow user’s login
* The system must check data correctness (including payment method validity)
* If user is new and data are correct, system must provide a password to the user

G2: Supervisors must be able to check cars’ status

Requirements:

* The system must be able to check every car’s remaining power
* The system must be able to check every car’s position
* The system must be able to check if a car is in use
* The system must be able to check if a car is reserved
* The system must be able to check every car’s “availability state”
* The system must be able to check how many passengers are in the cars
* The system must be always able to communicate with each car
* The system must keep the information of every car updated

G3: Supervisor should be able to dispatch “recharge on site” correctly

Requirements:

* The system must notify the supervisor if a car is left with less than 20% of the battery and is not plugged into a power grid

G4: Supervisor should be able to dispatch “car recovery” correctly

Requirements:

* The system must notify the supervisor if a car is left outside a safe area for more than 5 hours

G5: Guarantee the correctness of each car’s “availability state”

Requirements:

* The system must consider a car “unavailable” if it has low battery (<20%)
* The system must consider a car “unavailable” if it has already been reserved by a user
* The system must consider a car “unavailable” if it is not parked in a safe area
* The system must consider a car “available” in any other case
* The system must consider a reservation expired after 60 minutes if the car reserved isn’t used
* The system must consider a reservation expired 60 seconds after the car reserved is parked in a safe area and the user switch off the car and exit
* The system must consider a reservation expired if the reserved car remains parked outside a safe area for more than 5 hours

G6: Allow user to find available cars within a certain distance from a specified place

Requirements:

* The system must be able to detect the user’s location according to the user’s device’s GPS.
* The system must be able to detect cars’ location according to the cars’ GPS.
* The system must be able to detect a specific location according to the address provided by the user
* The system must be able to determinate the distance between available cars and the indicated position
* The system must show to the user the position, on the app’s map, of the available cars that are within 1500 meters from the indicated position

G7: Allow user to reserve a single car

Requirements:

* The system must allow the user to select a car among the ones that are showed after the search
* The system must show the user the “reserve” button after he has selected a car
* The system must show the user the estimated autonomy of the selected car
* The system must not show the reservation button if no car has been selected first
* The system must reserve the selected car for the user after he clicks the “reserve” button
* The system must prevent a user to reserve more than one car at a time

G8: Discourage fake and too long reservation

Requirements:

* The system must notify the user about the fee he will pay if he won’t use the car that he is reserving within 60 minutes
* The system must emit a payment request of 1 € to the credit card of the user who has reserved a car and did not use it within 60 minutes

G9: Allow only the user who reserved the car (and his passenger) to access it

Requirements:

* The system must be able to check the position of the user
* The system must allow the user to click the “unlock” button after he has reserved a car
* The system must not accept request of unlock if the user is more than 100 meters away from the car
* The system must unlock the car when he receives the unlock request from the user
* The system must lock the car if no user enters the car within 60 seconds from the unlock request
* The system must lock the car if the user and all the passengers exit the car

G10: Guarantee the correctness of the “cost of the trip”

Requirements:

* The system must be able to check when the car’s engine ignites
* The system must start count the minutes of car’s usage as soon as the engine ignites
* The system must stop count the minutes of car’s usage as soon as the car’s reservation is considered expired (see G5)
* The system must calculate the “cost of the trip” based on the formula:

Cost of the trip =(timeOfExpiration-timeOfFisrtIgnition)\*(cost per minute)

(see class and sequence diagrams)

G11: Guarantee the correctness of the “virtuousness coefficient”

Requirements:

* The system must be able to recognize how many passengers are in the car
* The system must consider the virtuousness coefficient’s initial value equal to 1
* The system must subtract “0.1” to the “virtuousness coefficient” if the user has shared the trip with at least 2 other passengers
* The system must subtract “0.2” to the “virtuousness coefficient” if the user left the car in a safe area with at least 50% of its battery power
* The system must subtract “0.3” to the “virtuousness coefficient” if the user cares to plug the car into a power grid within 60 seconds he parked the car in a safe area
* The system must add “0.3” to the “virtuousness coefficient” if the user left the car with less than 20% of its battery power
* The system must add “0.3” to the “virtuousness coefficient” if the user left the car at more than 3 KM from the nearest power grid station

G12: Guarantee the correctness and the payment of the final bill

Requirements:

* The system must calculate the amount of the final bill using the formula: total amount = (cost of the trip) \* (virtuousness coefficient)
* The system must emit a payment request of the amount of the final bill using the payment method of the last user, after 60 seconds that the last reservation has expired

G13: Support the users saving money

Requirements:

* The system must allow the user to check the “money saving” option when he reserves a car
* The system must allow the user to insert his destination after he checks the money saving option
* The system must be able to recognize the place entered by the user
* The system must know the availability of each power plug
* The system must calculate the nearest safe area to the user’s destination, with free power grids where to attach the car

**UML Models**

**Use Case Diagram**

****

**Description use case 1:**

**Title:** User registration

**Actor:** Guest

**Entry condition:**

* The guest user is on the home page of the app on his mobile device

**Flow of events:**

1. The guest clicks on “Register”
2. The system shows the form on screen
3. The guest inserts his name, mail, mobile phone number and the data of his payment method and then clicks on “Continue”
4. The system checks if the mail and phone number are syntactically valid, and if the payment method inserted is valid; if so, it generates the user’s password and sends it to him via email
5. The system shows the form for the password insert
6. The guest inserts the password that he received and clicks on “continue”
7. The guest is now a customer registered to the service

**Exit condition:**

1. If the password inserted is correct, the guest is now a customer registered to the service and the system redirects the user to the page dedicated for searching a car
2. If the password inserted isn’t correct, the system redirects the user to the home page

**Exception:**

* The guest is already registered
* One or more fields are not well-formed
* The payment method results not valid

**Description use case 2:**

**Title:** Login

**Actor:** User

**Entry condition:**

* The user is on the home page of the app on his mobile device

**Flow of events:**

1. The user clicks on “Login”
2. The system shows the form on screen
3. The guest inserts his email and password and clicks “Continue”
4. The system checks if the email and password are correct.

**Exit condition:**

1. If the password and email are correct, the system redirects the user to the page dedicated for searching a car
2. If the password or email aren’t correct, the system redirects the user to the home page

**Exception:**

* The email inserted doesn’t correspond to any already registered user
* The password inserted doesn’t correspond to the password related to the email

**Description use case 3:**

**Title:** Dispatch recharge on site

**Actor:** Supervisor

**Entry condition:**

* The system notices that the remaining power of the battery of a car has fallen to 20%

**Flow of events:**

1. The system waits until the reservation for that car has expired
2. The system checks if the car has been plugged into a power grid:
3. If this hasn't happened, the system shows to the supervisor the “dispatch recharge on site” button
4. The supervisor clicks on the “dispatch recharge on site” button
5. The information about the car’s position are sent to a worker

**Exit condition:**

* The worker go to the car
* The car has been plugged into a power grid

**Exception:**

* There are no exceptions for this use case

**Description use case 4:**

**Title:** Look for a car

**Actor:** User

**Entry condition:**

* The user is on the page dedicated to search cars

**Flow of events:**

1. The user clicks on the “search a car” button
2. The system recognizes the position of the user through his device’s GPS
3. The system calculates the distance between the position of the user and the position of each available car
4. The system sends to the user the position of the available cars whose distance from the user is less than 1500 meters

**Alternate Flows:**

1. The user inserts an address in the text box
2. The user clicks on the “search a car” button
3. The system recognizes the position of the address inserted by the user
4. The system calculates the distance between the position of the address inserted and the position of each car
5. The system sends to the user the position of the available cars whose distance from the inserted address is less than 1500 meters

**Exit condition:**

* The system shows on the app’s map, on the user’s device, an icon for each car found

**Exception:**

* There are no available cars nearby the position indicated
* The address inputted is not valid

**Description use case 5:**

**Title:** Reserve a car

**Actor:** User

**Entry condition:**

* The user searched a car and the system has found at least one available

**Flow of events:**

1. The user select a car among the ones shown on his app by clicking on its icon
2. The system shows the user the “reserve” button
3. The user clicks on the “reserve” button
4. The system creates the user’s reservation
5. The system change the availability state of the reserved car into “unavailable”

**Exit condition:**

* The system informs the user about the succeed of the reservation

**Exception:**

* The car that the user is trying to reserve has already been reserved by another user in the meantime
* The user has another active reservation

**Description use case 6:**

**Title:** Unlock a car

**Actor:** User

**Entry condition:**

* The user has reserved a car
* The user is on the “Your reservation” page

**Flow of events:**

1. The user clicks on the “unlock” button
2. The system recognizes the position of the user through his device’s GPS
3. The system calculates the distance between the position of the user and the position of the car

**Exit condition:**

* If the user is within 100 meters from the car, the system unlocks the car and notifies the user
* If he user isn’t within 100 meters from the car, the system notifies that he needs to approach the car

**Exception:**

* If no one get in the car within 60 seconds from the unlock, the system relocks the car

**Description use case 7:**

**Title:** Pay the bill

**Actor:** User

**Entry condition:**

* The user’s reservation has expired

**Flow of events:**

1. The system check why user’s reservation expired
2. If the reservation has expired for inactivity, the total amount of the bill Is 1€
3. In the other case, the total amount of the bill is calculated in the standard way

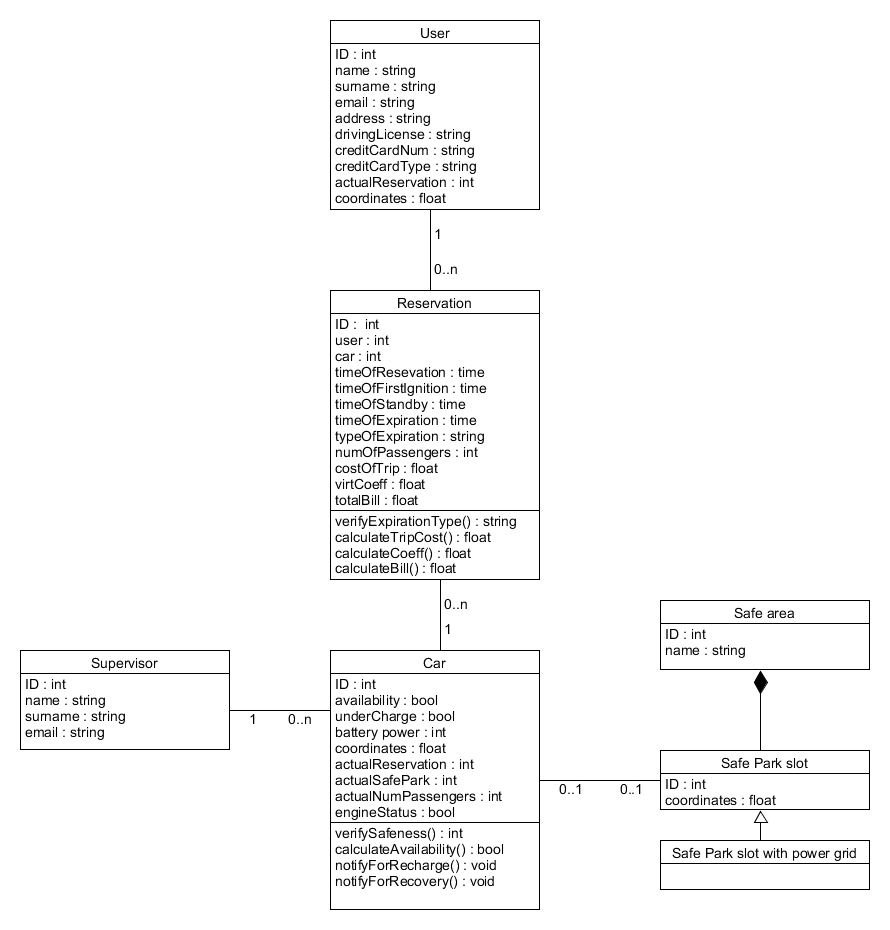
**Exit condition:**

* The system emits the payment request

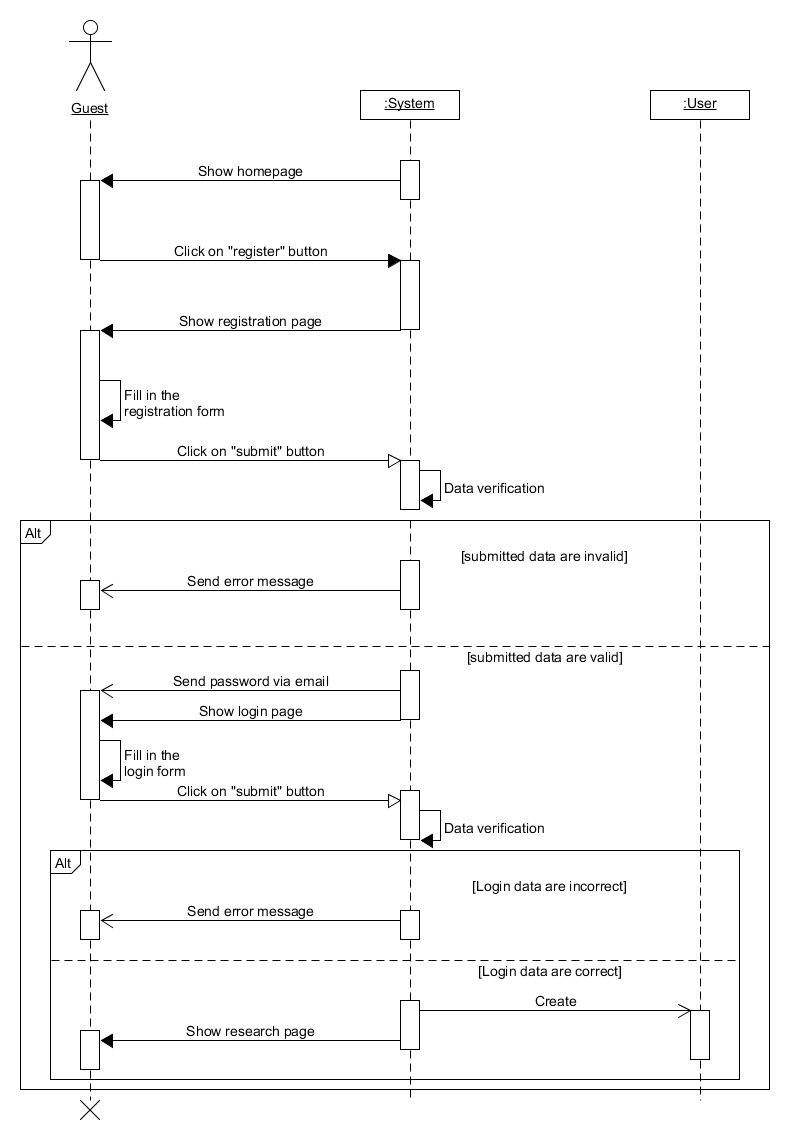
**Exception:**

* If no one get in the car within 60 seconds from the unlock, the system relocks the car

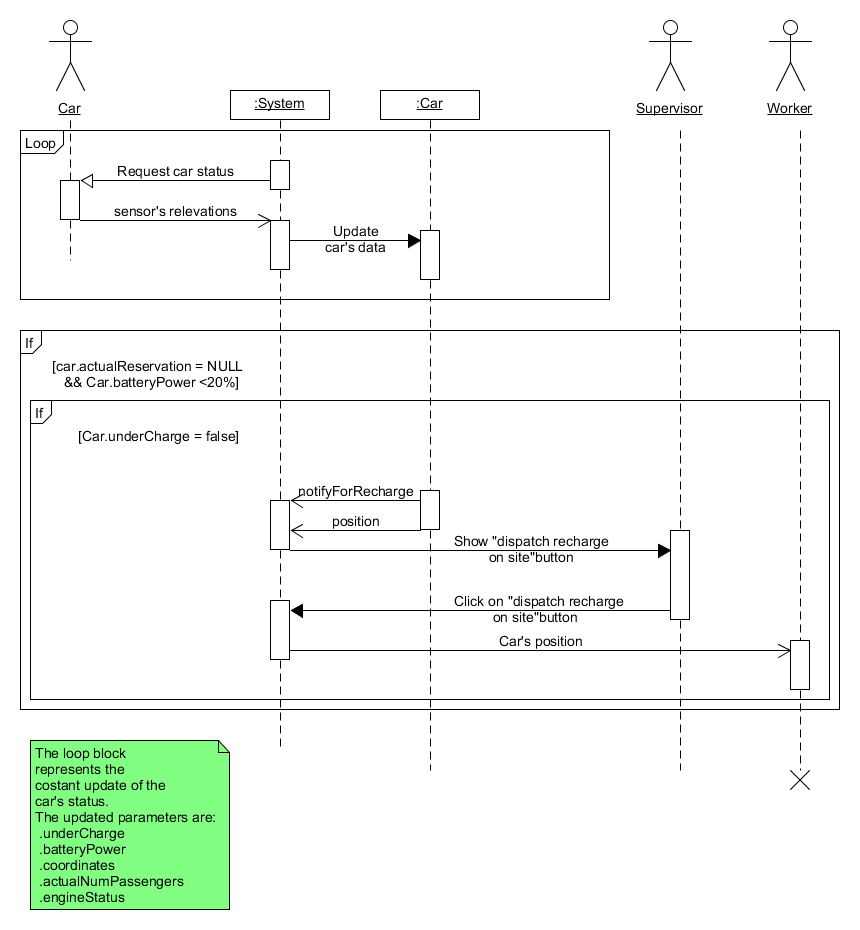
**Class diagram**



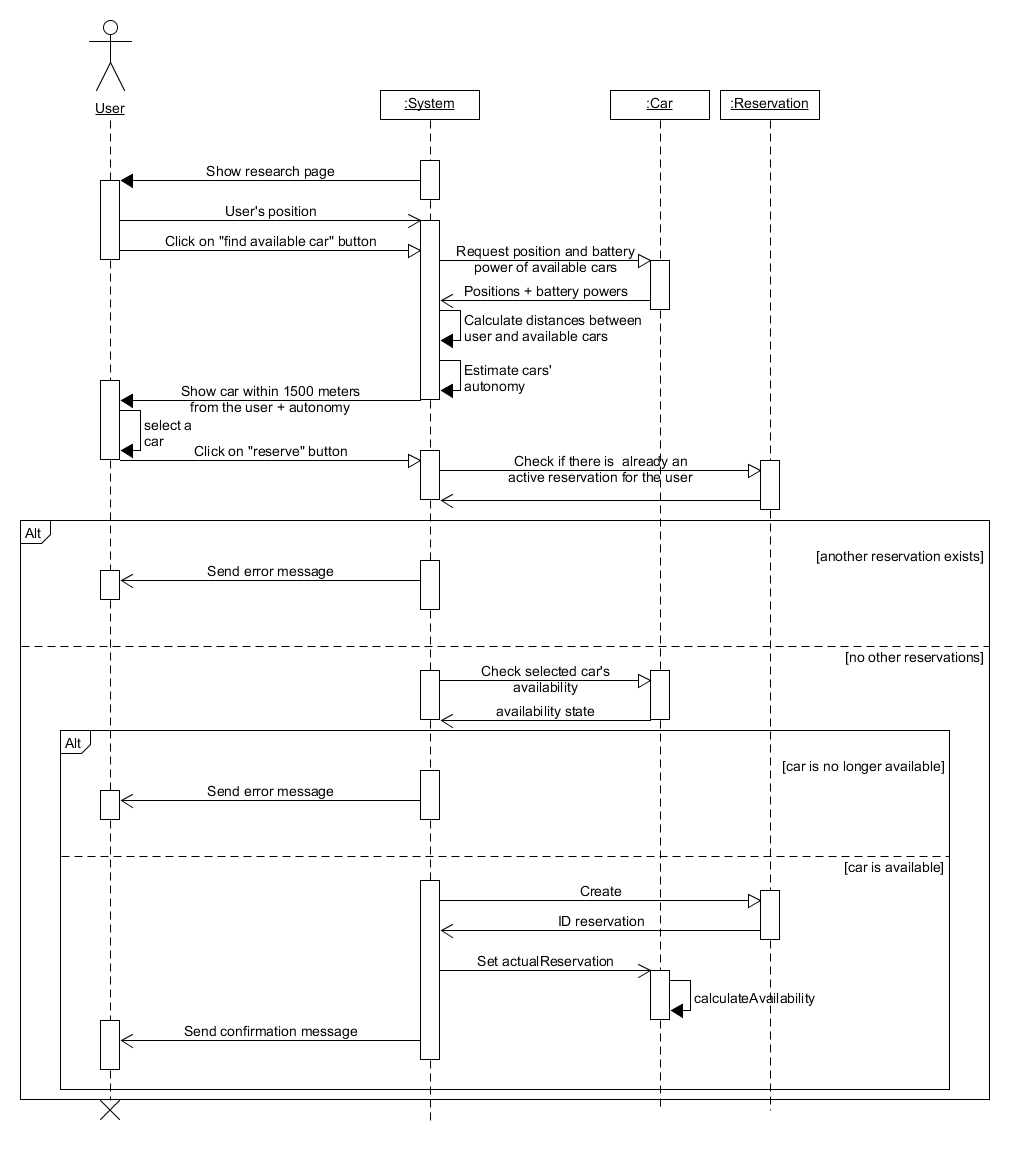
**Sequence diagrams**

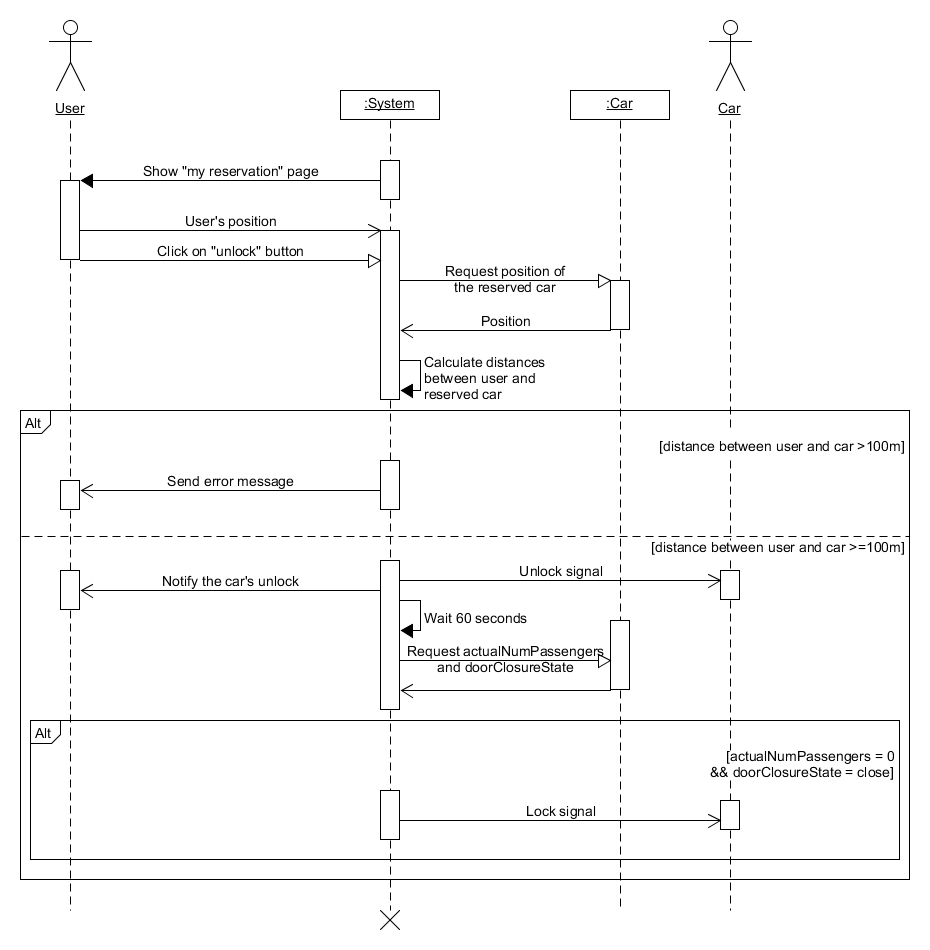
****

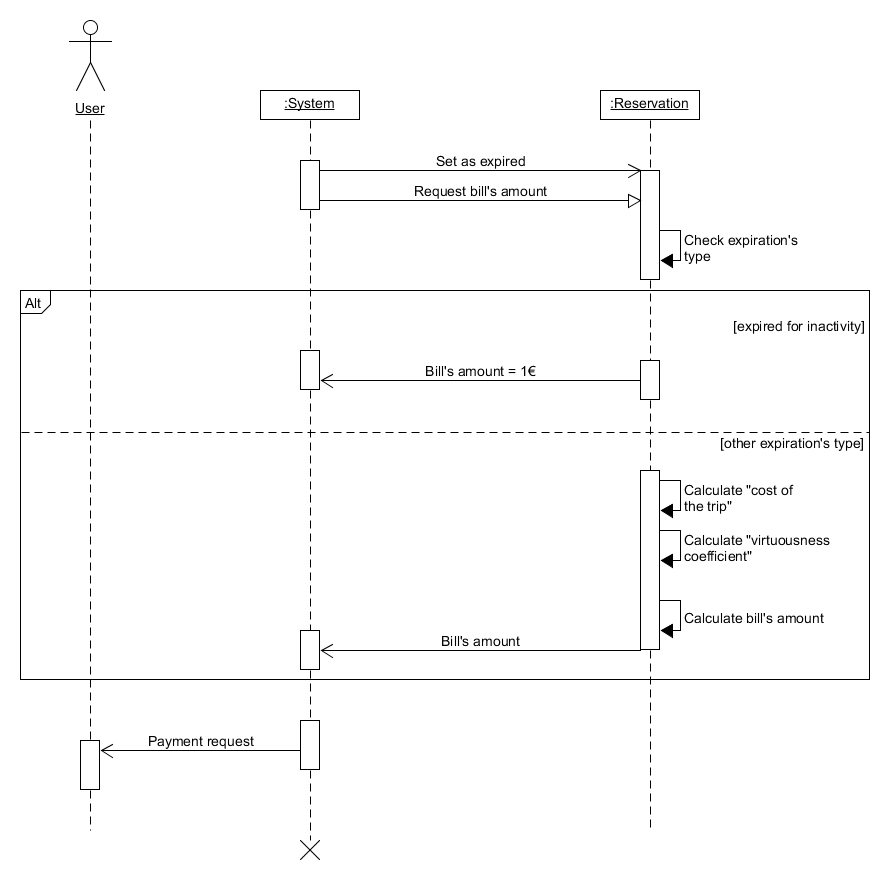
Sequence diagram “registration”

****

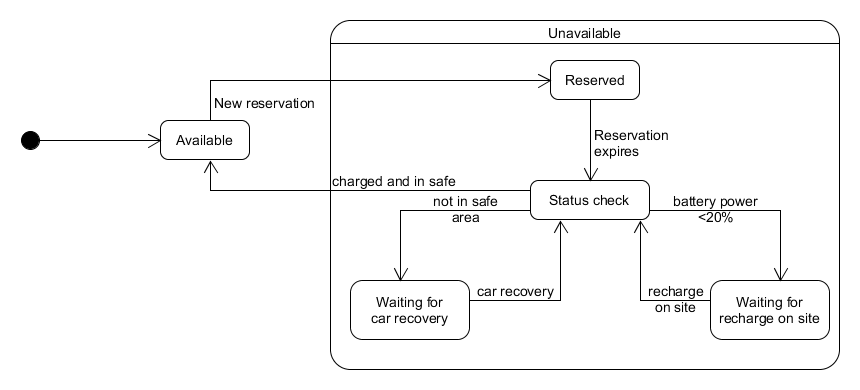
Sequence diagram “dispatch recharge on site”

Sequence diagram “Look for and reserve a car”

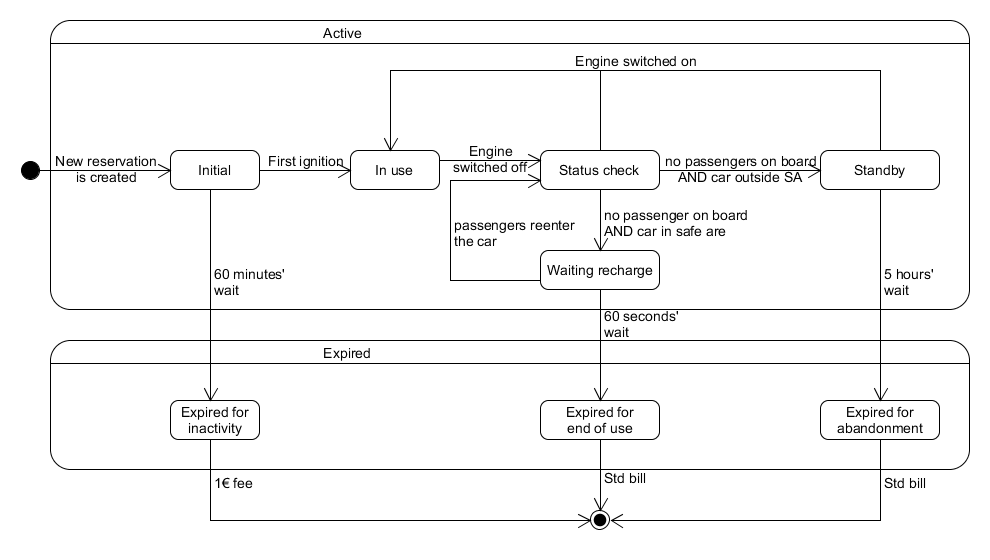
Sequence diagram “unlock the reserved car”

****Sequence “pay the bill”

**Statechart diagrams**

****

Car’s availability statechart



Reservation’s statechart

ALLOYS

////////////////////////////////ENUMS///////////////////////////////////////////

enum CarStatus {

AVAILABLE,

UNAVAILABLE

}

enum LotStatus {

EMPTY,

USED

}

enum ReservationStatus{

EXPIRED,

ACTIVE

}

////////////////////////////////PREDICATES///////////////////////////////////////////

sig ValidString{}

sig Supervisor {

//SupervisorID : Int,

manages: set Car

}{

}

sig RegisteredUser {

name: one ValidString,

//books : lone Reservation,

creditCard: one Int,

position: one Position

}{

#creditCard>0

}

sig ReservationDB{

contains: set Reservation

}

sig Reservation {

reservationID : Int,

bookedBy: one RegisteredUser,

relatedBy: one Car,

reservationStatus: one ReservationStatus

}{

reservationID>0

}

sig Position {

position\_x: one Int,

position\_y: one Int

}

sig Car {

carID: Int,

parkedInto: lone ParkingLot,

position: one Position,

carStatus: one CarStatus,

}{

carID>=0

}

sig SafeArea {

position : one Position

}{

}

sig ParkingLot {

lotNumber : Int ,

safeArea : one SafeArea,

status: one LotStatus

}{

lotNumber>=0

}

sig CHParkingLot extends ParkingLot{

}

////////////////////////////////FACTS///////////////////////////////////////////

// There aren't duplicated Cars

fact noDuplicatedCar {

no car1 , car2 : Car |(car1!=car2)&&( car1.carID = car2.carID)

}

// Two parking lot with the same number must be in different Safe Areas

fact noClonedParkingLot {

no parkingLoot1 , parkingLoot2 : ParkingLot |(parkingLoot1!=parkingLoot2)&&( parkingLoot1.lotNumber = parkingLoot2.lotNumber)&&

( parkingLoot1.safeArea = parkingLoot2.safeArea)

}

// Two car can't be parked at the same ParkingLot

fact noAbusedParkingLot {

no car1,car2 : Car |(car1!=car2)&&( car1.parkedInto=car2.parkedInto)

}

//Check for correct Pl's statuses

fact PLStatusesconsistency{

all pl : ParkingLot | (no c: Car | (c.parkedInto=pl))=>(pl.status!=USED) else (pl.status=USED)

}

// No cloned reservations

fact noClonedReservations {

no reservation1 , reservation2 : Reservation |(reservation1!=reservation2)&&( reservation1.reservationID = reservation2.reservationID)

}

// There's at least a Safe Area in the world

fact atLeastASafeArea{

#SafeArea>0

}

// And no SafeAreas are without a ParkingLot

fact atLeastASafeArea{

all s: SafeArea | some pl : ParkingLot | (pl.safeArea=s)

}

//There's only a Supervisor

fact thereIsOnlyASupervisor{

#Supervisor=1

}

//Every car is managed by the Supervisor

fact everyCarIsManaged{

all c : Car | c in Supervisor.manages

}

//Every reservation is stored, even the Expired ones

fact theReservationDBContainsAllTheReservations{

all r: Reservation | one rDB: ReservationDB | r in rDB.contains

}

// A car in a zone X should be parked in a ParkingLot...in a zone X

fact carAndLotInTheSameZone{

all c : Car |#c.parkedInto>0 => ( c.position=c.parkedInto.safeArea.position)

}

// A user can't have two "Active" reservations

fact{

no r1, r2: Reservation |( (r1!=r2)&&(r1.reservationStatus=ACTIVE)&& (r2.reservationStatus=ACTIVE))&&(r1.bookedBy=r2.bookedBy)

}

//A car should be "Active" for just one reservation at a time

fact{

no r1, r2: Reservation |( (r1!=r2)&&(r1.reservationStatus=ACTIVE)&& (r2.reservationStatus=ACTIVE))&&(r1.relatedBy=r2.relatedBy)

}

//If a car is being used, should be "Unavailable"

// we don't say the opposite, because a car could be Unavailable for other reasons, unrelated to the Reservation (e.g. battery)

fact {

all r : Reservation |(r.reservationStatus=ACTIVE)=>(r.relatedBy.carStatus=UNAVAILABLE)

}

////////////////////////////////ASSERTIONS///////////////////////////////////////////

pred show{}

//A Car Should Be in the same zone of his parkingLot

assert zoneConsistency{

all c : Car , zone1 : c.parkedInto , zone2 : c.parkedInto |(zone1=zone2)

}

//check zoneConsistency for 10

//A Car parked in a ParkingLot, marks it as USED ????????????????????????????????????????????????????????????

assert plConsistency{

all c: Car | (c.parkedInto.status!=EMPTY)

}

//check plConsistency for 5

//An empty PL should be empty ???????????????????????????????????????????????????????????? non va

assert plConsistency2{

all c: Car | all pl: ParkingLot | ((#c.parkedInto>0)&&(c.parkedInto!=pl))=>(pl.status!=USED)

}

//check plConsistency2

/\*assert reservationConsistency{

// no r1,r2 : Reservation |(r1!=r2)&&( r1.relatedBy = r2.relatedBy)&&(r1.reservationStatus=ACTIVATED)&&(r2.reservationStatus=ACTIVATED)

}\*/

/\*

assert noReservationActive{

no r1, r2, r3: Reservation |(r1!=r2)&&(r2!=r3)&&(r1!=r3)&&(r1.reservationStatus=ACTIVE)&&(r2.reservationStatus=ACTIVE)&&(r3.reservationStatus=ACTIVE)

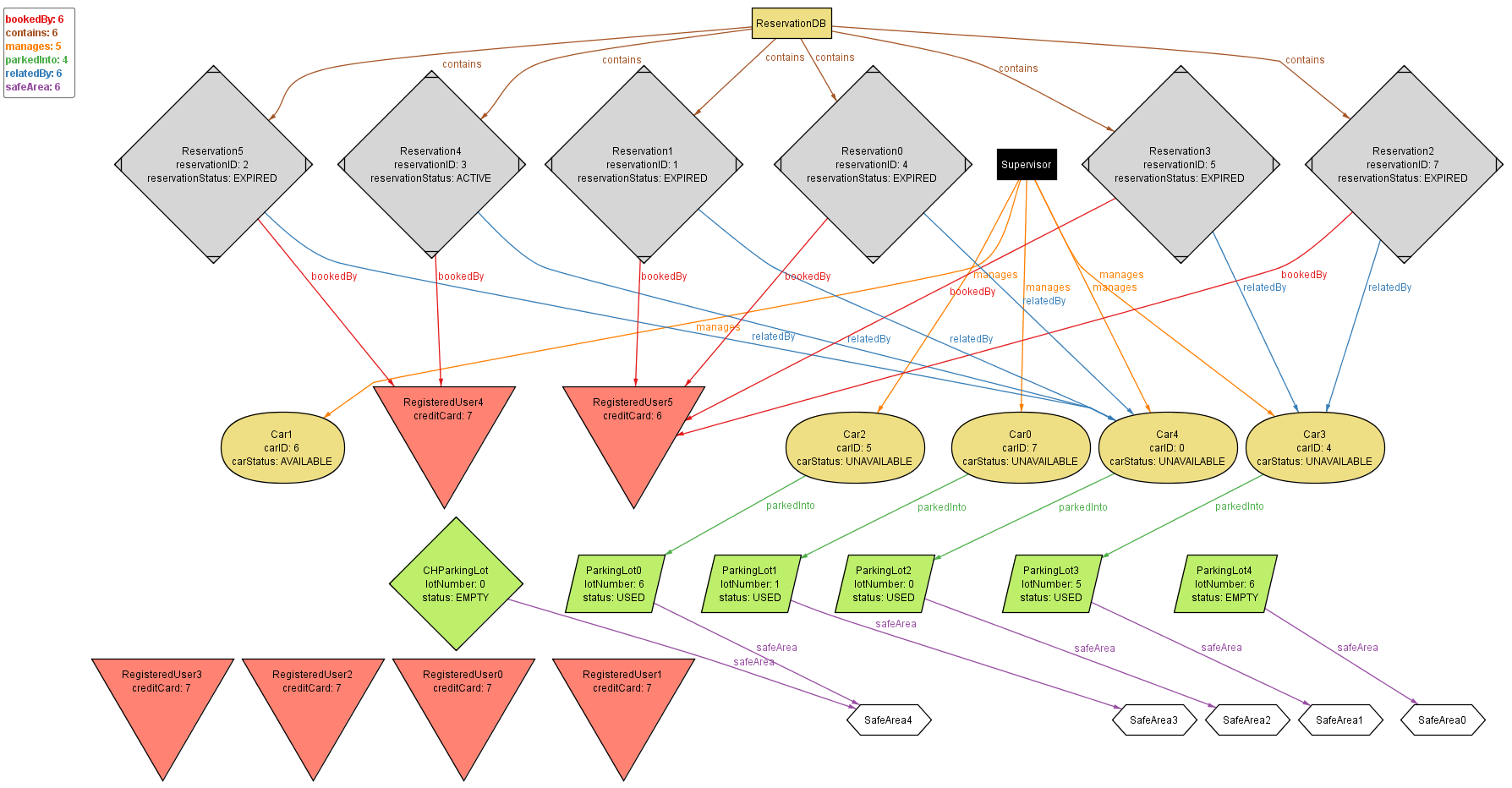
}

check noReservationActive for 5

\*/

///////////////////////////////////////////////////////////////////////////////////

run show for 10



**Conteggio lavoro:**

GABRIELE

22/10/16: 1h

23/10/16: 2h

24/10/16: 30’

26/10/16: 3h

27/10/16: 1h

29/10/16: 1h

30/10/16: 2h

31/10/16: 2h

02/11/16: 1h

03/11/16: 3h

05/11/16: 2h 30’

06/11/16: 3h

07/11/16: 2h

MARCO

24/10/16: 1h

26/10/16: 1h

27/10/16: 2h

30/10/16: 1h

05/11/16: 2h 30’

06/11/16: 1h

07/11/16: 2h

8/11/16: 8h