SOFTWARE ENGINEERING 2

Requirement Analysis and Specification Document (RASD)

AUTHORS

Paolo Paterna, Lara Premi

Politecnico di Milano, Italy

Email: paolo.paterna@mail.polimi.it, lara.premi@mail.polimi.it

6th Nov 2015

Contents

1 Introduction

1.1 Purpose - Description of the Problem

This document is the Requirement Analysis and Specification Document (RASD) of a system, called myTaxiService, used to manage a taxi service in a city. The main goals of this document are to: describe the system in terms of functional and non-funcional requirements, specify the constrains and how the system will beheave in its application, once that it is deployed with some scenarios in example. This document is meant to be read by the developers of the software to make the point on how it must be developed, but also by the customer, to be sure that all what he asks is well defined and corresponds to what is going to be developed; this document, hence, can be used as a contract between the customer and the developers.

1.2 Scope

The scope of this system is to manage taxis in a city. A town is divided in zone of 2 square kilometers and, for each zone, the system defines a queue, composed by the identifier, which is the vehicle plate, of free taxis in that specific zone. A user can require a taxi ride from a zone, but can also book one for another moment, using the web application or the mobile one. About long-term reservations, the user can also, after have created one, modify the date or the hour or both of his/her booking, and he/she can delete it.

1.3 Actors

- **Users**: users can request a taxi ride or book one, using the mobile app or the website.

- **Taxi**: it is an entity, composed by the taxi driver and the car; it works for a taxi company and it provides the service to the users by transporting them to their destination.

- **Taxi not yet registered**: taxis that are not yet registered on the system and do not belong to any queue.

1.4 Goals

The software has to be able to:

- *G1:* booking a Taxi ride.

- *G2:* update a long-term reservation.

- *G3:* delete a long-term reservation.

- *G4:* locate the Users.

- *G5:* managing Taxis location.

- *G6:* managing Taxis queue.

- *G7:* allow Taxi to register in the system.

- *G8:* notify the users about their booking changes.

- *G9:* notify Taxis about new available rides.

1.5 Definitions, acronyms and abbreviations

1.5.1 Definitions

- myTaxiService: the system described in this document.

- User: someone who uses the services offered by myTaxiService, sometimes called also passenger.

- Taxi: entity composed by the taxi driver and the taxi car that transports the users.

- Booking: generic reservation, that is, it can be long-term reservation or a short-term one.

- Short-term reservation: booking, requested from the user, for the current time.

- Long-term reservation: booking, requested from the user, characterized by a time, at least two hours before the requested service, or by a different day from the current one.

- Zone: part of the city, large 2 square kilometers, that is disjoint with reference to the other zones.

- Queue: list of available taxis in a specific zone.

- Busy Taxi: taxi that is doing a ride.

- Available Taxi: taxi not busy.

1.5.2 Acronyms

- FIFO: First in, First out.

1.5.3 Abbreviations

- Gi: goal number i.

1.6 Reference Documents

- IEEE Std 830-1998 (Revision of IEEE Std 830-1993) document available on the `http://standards.ieee.org/` website.

- Assignment 1 from the *Assignment 1 and 2.pdf* file on BeeP university platform.

1.7 Overview

Section 2 and 3 of this document provide a synthetic but exhaustive description of the product that will be developed.

Specifically, section 2 provides a description of the general factors that affect the product and the requirements' background, specifying the product perspective, user characteristics product functions, product constrains, assumptions and dependencies, made to write this document.

Section 3, instead, focuses on the specific requirements of the product listing all the external, hardware, software and communication interfaces if any, functional and non-functional requirements, the description of *The World and The Machine* graph, listing some scenarios of interest in which the product will operate and other requirements that do not belong to any of the previous category.

2 Overall description

2.1 Product perspective

Whereas the system is too much little, considering its dimensions, we have consolidated the six subsections 2.1.1 through 2.1.8 of the IEEE Std 830-1998.

This system is a stand alone system, but it can be extended using the featured API.

In the case of web application, a user must have a computer to access the service site. It's necessary also a browser to allow navigation. On the user side the application is not heavy and so it doesn't require any particular hardware. Using the site, users can update himself/herself about this service and, then he/she can book a taxi, for a short-term reservation or a long-term one. After that, he/she can modify his/her reservation or delete it. In addition of these actions, the user can view his/her reservation and he/she can view the system notifications.

In the case of mobile application, a user must have a smart phone with an Internet connection to access to the service app. It's necessary that the cell phone has got sufficient memory to allow the installation of this app, that is compatible with all of the OSs (i.e. Android, iOS and Windows Phone). Through this mobile application, the user can do the same stuff that he can do using the web application: in particular, he/she can book a taxi (always for a short-term or a long-term reservation), he/she can modify his/her reservation and he/she can delete it.

The taxi driver, through the mobile application, can view some information about the current ride, the notifications from the system, previous rides. Then the taxi driver can send to the system notifications about delays that will be forwarded to the user by the system.

2.2 Product functions

2.2.0.1 User side

The system allows the users to book a taxi using the website or the mobile application installed on their smart phones. They also can reserve a taxi but they must do it at least 2 hours before the chosen hour; in this case the system assigns a unique code for every reservation in order to allow users to modify or delete their reservation. The system also sends notification to the users about the status of their bookings and reservation or information

about waiting time for their ride.

2.2.0.2   Taxi side

The system allows taxi to register into it in order to be added to the queue and be given new passengers. The system also sends notification to taxis about new jobs. It also allows taxis to notify users the system about delays.

2.3   User characteristics

The users that intend to use the product are people that want to travel using taxis service. They should know how to use a web browser or a smarthphone in order to use the web application or the mobile app. Also the taxi drivers should how to use the smartphone to be able to use the mobile application.

2.4   Constraints

- The system must maintain the privacy about personal data of users and taxi drivers.

- The system must allow operations to run in parallel.

- The system must work 24 x 7.

- In case of long-term reservation, the user must book a taxi at least two hours before the meeting time.

- In case of long-term reservation modification or elimination, the user must do it at least 15 minutes before the meeting time.

2.5   Assumptions

- The terms "code" and "identifier" relate to the same thing, that is the taxi plate.

- The fair queue management is characterized by: at most 10 minutes of wait; a FIFO method to organize the taxis queue.

- The zones in which the city is divided are always the same, don't change position and they never overlap.

- The taxis that are outside of the city or aren't in any defined zone are considered busy until they return in a known zone.

- The taxis that don't answer to the system notifications of new incoming rides are considered as busy.

- The GPS works always, pooled every 2 minutes.

- As soon as the Taxi logs in, he/she is assumed "Available" and the system puts him/her in the corresponding queue of the zone.
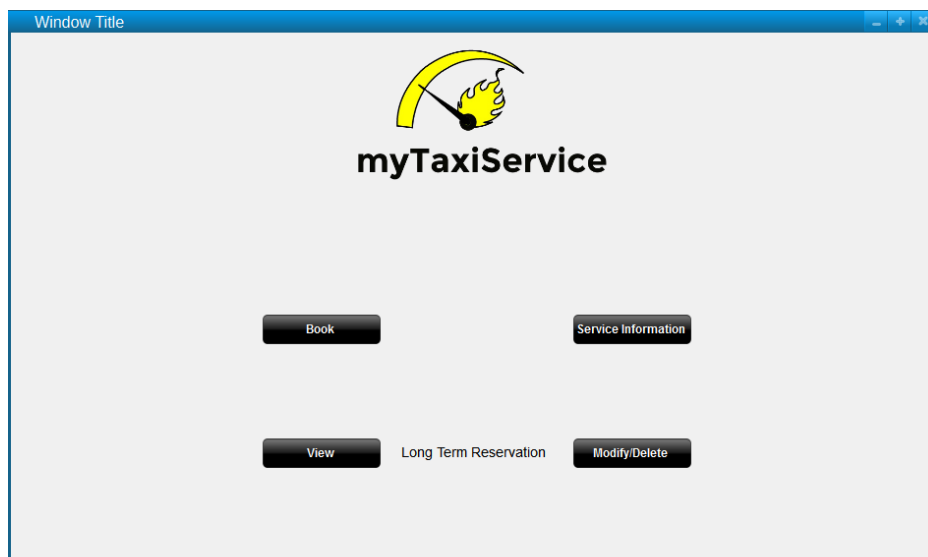
3 Specific requirements

3.1 External interface requirements

3.1.1 User Interfaces: Web Page

In this section is presented a minimal mock-up of the web and mobile interfaces on both users and taxis side.

These have to be considered as an example of what at least is needed in the interface and don't are final version of the product.

3.1.1.1 Index

3.1.1.2   Book a Taxi

3.1.1.3   View/Modify/Delete a Long Term Reservation

3.1.1.4   Who We Are
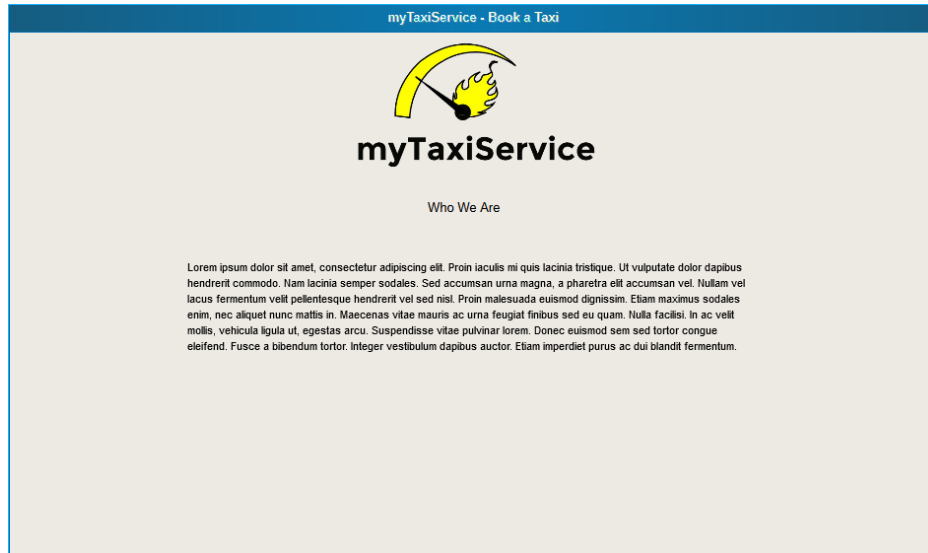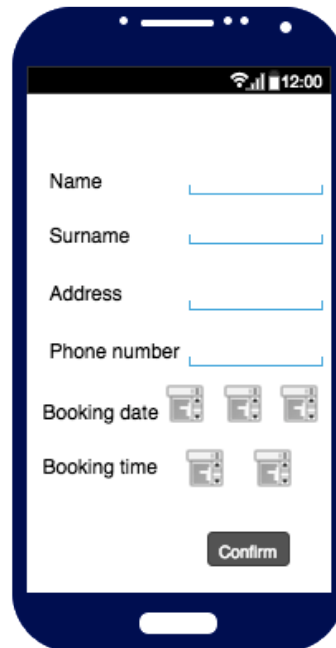
3.1.2   User Interfaces: Mobile Application

3.1.2.1   Home Page
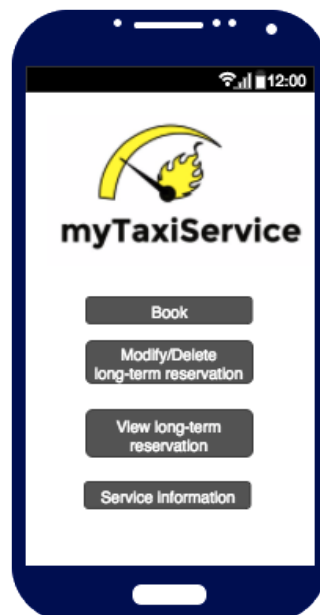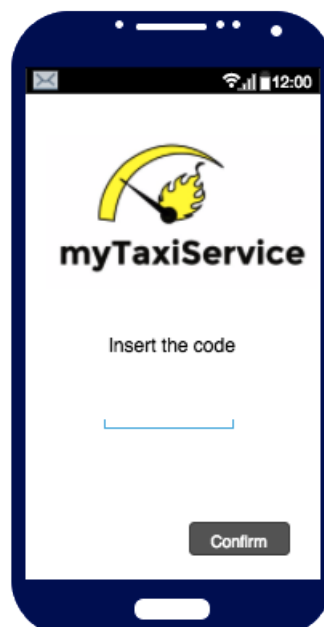
3.1.2.2 Booking

3.1.2.3 Insertion of the long-term reservation code

3.1.2.4 Choosing of long-term reservation modification or elimination
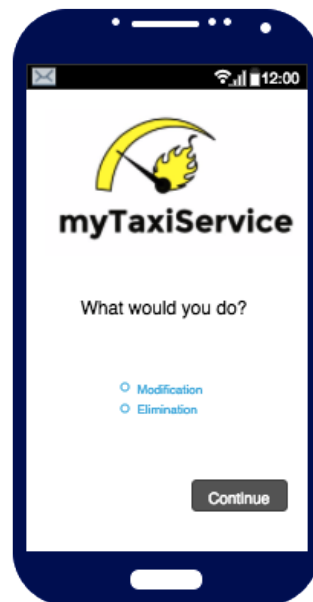
3.1.2.5 Choosing of new long-term reservation date and time
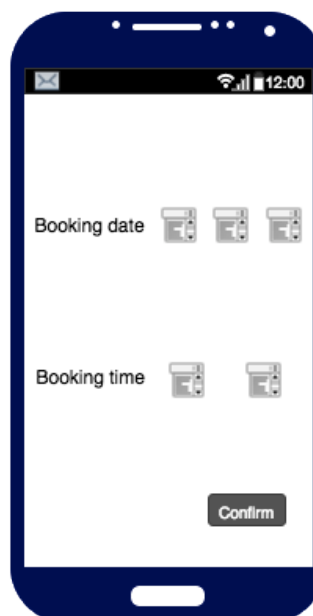
3.1.2.6 Service Information
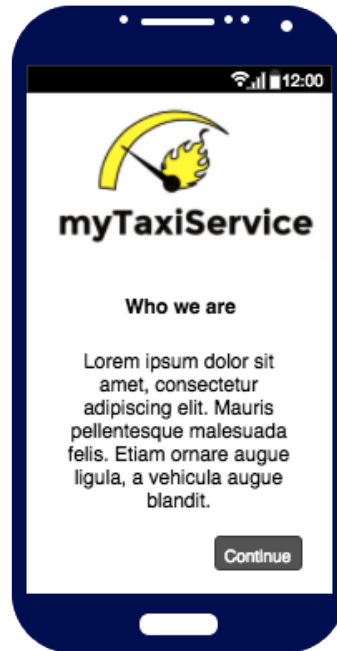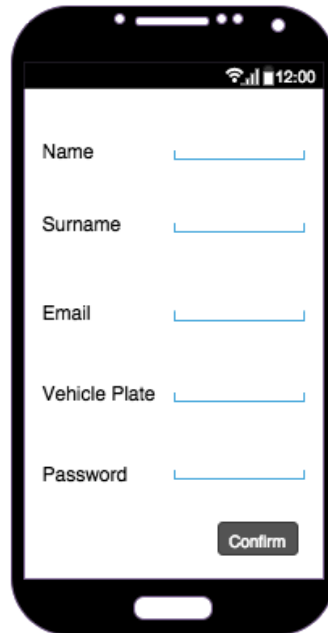
3.1.3 Taxi Interfaces: Mobile Application

3.1.3.1 First Page

3.1.3.2 Registration Page

3.1.3.3 Login

3.1.3.4 Home Page

3.1.3.5 View Information About The Current Ride

3.1.3.6 View Information About The Previous Rides
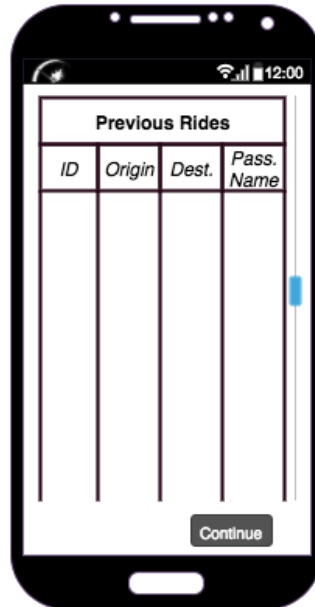
3.1.3.7 Service Confirmation/Rejection
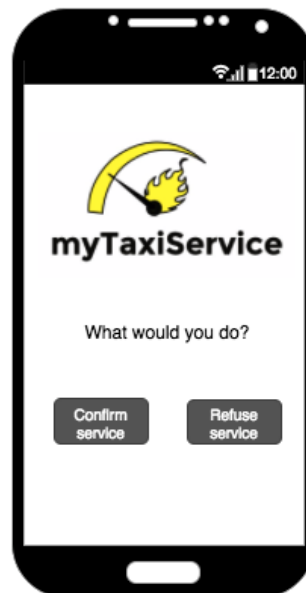
3.1.4   Hardware interfaces

There is the necessity to have:

- a machine dedicated to database-server, capable to manage high-volume transactions;

- a machine to run the main software and the web-interface;

- a private redundant gigabit network connecting both machines;

- a public redundant connection to internet for the web-server machine, with adequate speed and scalable traffic quota.

Hardware can be purchased or rented as virtual-server or cloud. In any case, the server must be colocated to appropiate server farm to guarantee safety and security.

3.1.5   Software interfaces

There is the necessity to have:

- a web server to provide information to browsers and mobile apps;

- a DBMS to store all the data managed by the system.

3.1.6   Communication interfaces

The system will use the API, supplied by the external provider, being able to send SMS, as notifications, to the user.

3.2   Functional Requirements

3.2.1   Booking a Taxi ride

- First of all, the system checks the date and the hour of the reservation.

- In case of long-term reservation:

  1. the system produces an alphanumeric code;
  2. the system stores this code, with name and surname of the user, in the relative database;
  3. the system assigns this code to the user.

- In case of short-term reservation:

1. the system checks what is the address inserted by the user;

2. the system checks in which zone of the city is that address, using the GPS information;

3. the system controls the taxis queue of that zone;

4. if the taxis queue contains at least the identifier of one taxi:
   - the system sends a notification to the first taxi driver of the queue;
   - when it will receive the confirm, the system tells the user that the reservation is been realized;

5. if the taxis queue hasn't got any taxis identifiers:
   - the system checks the queue of the four adjacent zones of that area;
   - the system continues in this way as long as it finds an available taxi and then it will do the same things of the previous point;
   - when the system will receive the confirm, it tells the user that there will be a delay;
   - if the user decides to accept, the system tells him/her that the reservation is been realized, otherwise not.

3.2.2 Update a long-term reservation

- The system receives the alphanumeric code from the user.

- The system checks the DB of the users.

- The system displays the information, collected from the DB, to the user.

- The system receives the modifications.

- The system saves the changed information in the DB.

- The system confirms to the user the modifications.

3.2.3 Delete a long-term reservation

- The system receives the alphanumeric code from the user.

- The system checks the DB of the users.

- The system deletes the found information.

- The system confirms to the user the correct elimination.

3.2.4 Locate the Users

- The system takes the address, inserted by the user.

- The system searches in which zone of the city is that address.

3.2.5 Managing Taxis location

- The system uses the GPS information.

- The system locates the position of all the taxis in the city.

- The system saves these information in the database and updates them every 2 minutes.

3.2.6 Managing Taxi queue

- Only in the case that a taxi is available, the system inserts its identifier in the zone queue, that is unique.

- If there is a new ride, the system chooses the first taxi in the queue, so, using a FIFO policy.

- If there is a new ride, but the interested zone queue is empty, the system checks the queues of the four adjacent zones; the system continues in this way as long as it finds an available taxi.

3.2.7 Allow Taxi to register in the system

- The system stores the data inserted by the Taxi Not Yet Registered.

- The system lets the Taxi to log in, using the vehicle plate and the password, inserted during the registration.

3.2.8 Notify the users about their booking, their long-term reservation modification or elimination and about possible delays of the taxi

- After a user's booking, the system sends to him, via SMS, a notification of the occurred booking.

- After a long-term reservation modification, made by the user, the system sends to him, via SMS, a notification of the occurred modification.

- After a long-term reservation elimination, made by the user, the system sends to him, via SMS, a notification of the occurred elimination.

- After receiving from a Taxi a delay notification, the system sends to the user, via SMS, a notification of the Taxi delay.

3.2.9 Notify Taxis about new available rides

- The system sends to the first Taxi in the zone queue a notification of the new ride.

- If the Taxi answers the system with a negative reply, it sends another notification to the next Taxi of the zone queue; the system continues in this way as long as it finds an available taxi.

3.3 The World and the Machine
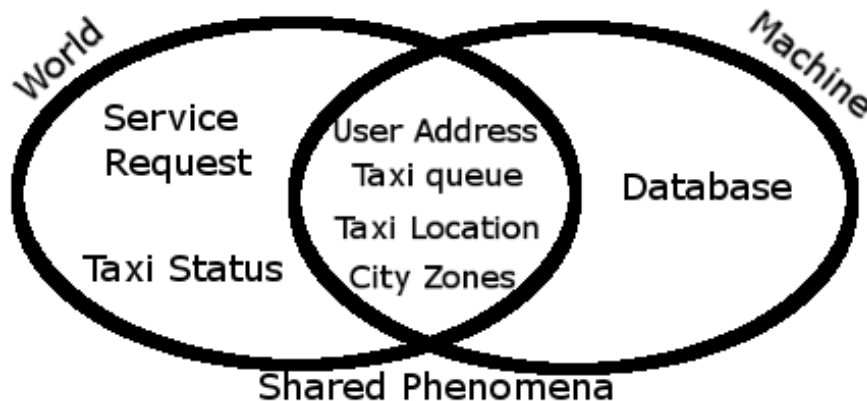
Service request and Taxi status are events that happen in the physical world; the machine then uses the data coming from the world to build a representation of it using taxi queue, taxi location and city zones. The database is only stored in the machine and contains all the data about users to send notification, long term reservations and data about taxis.

3.4 Scenarios

3.4.1 Scenario 1

Bob has a meeting in Los Angeles the day after tomorrow. Since he hasn't a car, he decides to use myTaxiService and therefore rent a taxi to go to the airport. He grabs his cell phone and, with the mobile application of the

service, he books a taxi. The app asks him his name, his surname, his phone number and the address where he wants to wait for the taxi. He inserts all the data requested and, after the server has stored these information in the database, with a SMS, it notifies Bob that the long-term reservation has been realized correctly. The chosen day, ten minutes before the appointment, the system calls the first available taxi, searching it in the taxis queue of the zone in which there is the address specified by Bob. Unfortunately the first taxi doesn't answer the call; so the system puts him at the end of the queue and it calls the second taxi of the queue, which accepts the assignment. Then, Bob receives from the system, via SMS, a notification in which it was clarified that a taxi was coming to drive him to his destination.

3.4.2 Scenario 2

Today Francois starts his new job as a taxi driver. Yesterday morning, he signed up himself in myTaxiService application. Through his smart phone, he inserted only four information: his name, his surname, his email address and finally a password which he will use to sign in. After that, the system assigned him an alphanumeric code, that was the register of the taxi, which he will drive. During this morning, Francois signs in, inserting the username, that is the register of his taxi, and the password that he choose yesterday, during the registration. Clicking on the "Confirm" button, he accesses in the homepage of the mobile application. Francois has many possibilities to choose: he can view information about the current ride; he can view all the rides that he did in the past; he can send notifications of delay to the system and finally he can log out, of course at the end of the business day. While he is browsing the mobile application, Francois receives from the system a notification of a new ride. After the opening of this notification, Francois confirms the service and his business day starts.

3.4.3 Scenario 3

Carl wants to go to visit one of his friends. Using the smart phone, he accesses to myTaxiService mobile application and he books a taxi, inserting name surname phone number and address, in the relative forms. The system stores these information in the database and then it checks in which zone there is the address specified by Carl. After that, the system notifies the first taxi in that zone's queue. The taxi receives a notification about that service by the system and he confirms the job, through the mobile application installed on his own smart phone. Unfortunately a taxi's tyre runs flat; so the taxi driver sends a delay notification to the system, informing it that he

can't complete the job. After the receive of this notification, therefore, the system puts that taxi at the end of the queue and it calls the second one that accepts the assignment. The system advises Carl, via SMS, that there will be a little delay.

3.4.4 Scenario 4

Thanks to one of his closest friends, Tizio has known a service, called My-TaxiService, that lets to book a taxi, through the web application or the user-friendly mobile one. Whereas his computer was broken, using his smart phone, first of all, Tizio has inserted his personal data, that were name, surname, phone number and home address. In that moment, the system has received the request of Tizio and it has checked in what zone of the city Tizio was asking for the service. In the taxis queue of that area, the system could have verify that there were no taxis. So it has become to check, starting from the zones near the area in which Tizio has requested a taxi, all the lists of available taxis. Finally the system has found the only one taxi which, in that moment, was available. So the system has sent to Caio the notification of the new route. Besides Caio was standing, with his taxi, in a zone far from the place in which there was Tizio, he has confirmed the service, using the mobile application of his smart phone. After receiving the confirmation, the system has warned Tizio that the taxi would be arrived with a delay. About one hour later, Tizio is get into the Caio car and so, in a short time, he is arrived in the place where he would have meet his girlfriend Sempronia, relaxing himself, discovering that she was late more than him.

3.4.5 Scenario 5

Ann receives a call from one of her friend that tells her that her best friend has given birth to her son, Jimmy. Ann, due to the trains' strike decides to call myTaxiService to take her to the hospital to visit the newborn. So from her pc, she opens the web site of myTaxiService, that is www.mytaxiservice.com, clicks on the "Book" button to rent a taxi, after she has inserted all the needed data, that were name, surname, phone number and address, the system checks the taxis queue of the zone in which there is the address of Ann. Discovering that there is no available taxi in that zone in this specific time, the system searches for an available taxi in the queue of the adjacent zones. After it finds a free taxi in a nearby zone, the system notifies Ann, via SMS, that her reservation has been realized correctly but the taxi will arrive to Ann's address with a little delay.

3.4.6 Scenario 6

Finally, today, Rossana would have taken the airplane. The flight was booked three days ago, but, because of bad weather, it was been deleted. So, she has accessed to the website of MyTaxiService to modify her reservation. She has used her notebook and then she has gone to the web page, concerning the modification/elimination of reservations. In the dedicated form, she has inserted the code that the system gave her to do this stuff, if she would have needed, but she could do it only until 15 minutes before the stipulated time. So she has postponed the reservation, without any problems. One hour before the meeting with the taxi, she has received a call from her boyfriend Heric, who wasn't really enthusiastic about her decision to go to USA for the Erasmus. When he has known that Rossana hasn't left yet, Heric has decided to give her a ride to the airport. So the girl, for that moment using her smart phone (the computer was in the luggage), has accessed to the mobile application of MyTaxiService to delete definitely her reservation. Going again in the dedicated page and then inserting her code in the specific form (but now choosing the "Elimination"), she has confirmed the cancellation of the reservation.

3.5 UML Models

3.5.1 Use Case

3.5.1.1 Registration of a new taxi driver

Name	Registration
Actors	Taxi not yet registered
Entry conditions	No entry conditions

Event flow	1. The new taxi driver not yet registered
	
	    • opens the myTaxiService mobile application;
	    • clicks on "Sign In" link;
	    • inserts name, surname, email address and a password for the login, after the opening of a new window;
	    • clicks on "Confirm" link;
	
	2. the system assigns an identifier, that is the taxi plate of the new taxi and which will be used by the new taxi driver as his username for the login.

Exit Condition	The system adds the new taxi driver in the database and it grants him access to the application.
Exceptions	Password inserted wrongly.

3.5.1.2 User side

3.5.1.2.1 Booking a ride

Name	Book a short-term reservation
Actors	User, Taxi
Entry conditions	No entry conditions
Event flow	1. The user accesses to the web site or the mobile application. 2. The user clicks on "Book a ride". 3. The user inserts name, surname, phone number and address. 4. The user clicks on the "Confirm" button. 5. The system inserts the data of the user in the database. 6. The system identifies the area in which the address specified by the user is. 7. The system selects the first available taxi from the taxis queue. 8. The system sends the notification to the taxi. 9. The system sends a confirmation via SMS to the user.
Exit Condition	The system adds the user information in the database.
Exceptions	Address inserted wrongly.

Name	Book a long-term reservation
Actors	User, Taxi
Entry conditions	No entry conditions
Event flow	1. The user accesses to the web site or the mobile application. 2. The user clicks on "Book a ride". 3. The user inserts name, surname, phone number and address. 4. The user specifies the date and the hour of the long-term reservation. 5. The user clicks on the "Confirm" button. 6. If the date is the actual one, the system checks if the current time is at least two hours before the long-term reservation time. 7. The system inserts the data of the user in the database. 8. The system sends a confirmation of the long-term reservation via SMS to the user, with an alphanumeric code, that the user can use to modify and/or delete this long-term reservation. 9. Ten minutes before the meeting time with the user, the system searches in the taxis queue of that area the first available taxi. 10. The system sends a notification to that taxi. 11. If the taxi accepts the service, the system sends a confirmation to the user. 12. Otherwise, the system searches the next available taxi.
Exit Condition	No exit conditions
Exceptions	• Address inserted wrongly; • Data and/or hour not valid.

3.5.1.2.2   Service Information

Name	Service Information
Actors	User
Entry conditions	No entry conditions
Event flow	1. The user accesses to the myTaxiService web site or the mobile application. 2. The user clicks on "Service Information" link. 3. The user can learn about the myTaxiService.
Exit Condition	No exit conditions
Exceptions	No exceptions.

3.5.1.2.3 Modify long-term reservation

Name	Modify long-term reservation
Actors	User
Entry conditions	No entry conditions
Event flow	1. The user • accesses to the web site or the mobile application; • clicks on "Modify/Delete long-term reservation"; • inserts the alphanumeric code; • clicks on the "Confirm" button; • chooses "Modification"; • clicks on "Continue" link; • modifies the date and/or the hour; • clicks on "Confirm" button; 2. the system • sends a confirmation via SMS to the user; • modifies the changed data of the long-term reservation from the database.
Exit Condition	No exit conditions
Exceptions	• Alphanumeric code inserted wrongly; • data and/or hour not valid.

3.5.1.2.4 Delete long-term reservation

Name	Delete long-term reservation
Actors	User
Entry conditions	No entry conditions

Event flow	1. The user
	
	    • accesses to the web site or the mobile application;
	
	    • clicks on "Modify/Delete long-term reservation";
	
	    • inserts the alphanumeric code;
	
	    • clicks on the "Confirm" button;
	
	    • chooses "Elimination";
	
	    • clicks on "Continue" link;
	
	2. the system
	
	    • sends a confirmation via SMS to the user;
	
	    • deletes the data of the user and the ones of the long-term reservation from the database.

Exit Condition	No exit conditions
Exceptions	Alphanumeric code inserted wrongly.

3.5.1.2.5   View long-term reservation

Name	View long-term reservation
Actors	User
Entry conditions	No entry conditions
Event flow	1. The user - accesses to the web site or the mobile application; - clicks on "View booking"; - inserts the alphanumeric code; 2. the system loads from the database the information about the long-term reservation.
Exit Condition	No exit conditions
Exceptions	Alphanumeric code inserted wrongly.

3.5.1.3 Taxi side
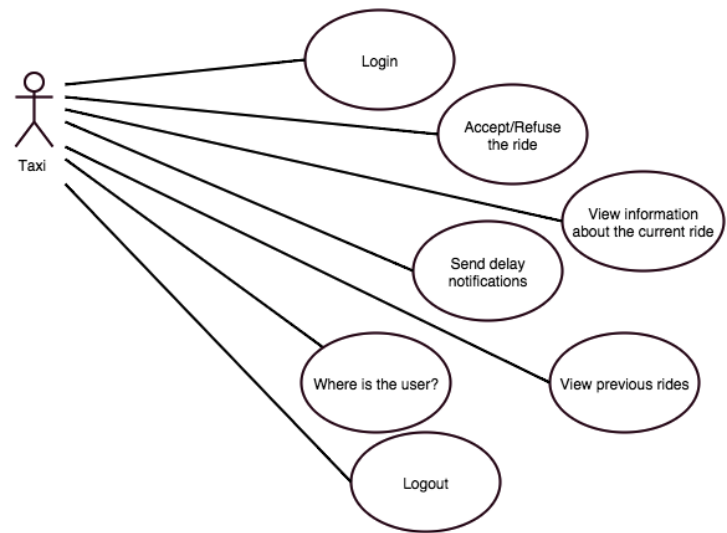
3.5.1.3.1 Login

Name	Login
Actors	Taxi
Entry conditions	No entry conditions

Event flow	1. The taxi
	
	    • opens the myTaxiService mobile application;
	
	    • clicks on "Log in" link;
	
	    • inserts username and password, after the opening of a new window;
	
	    • clicks on "Confirm" link;
	
	2. the system accepts the data inserted and the main page is shown to the taxi driver.

Exit Condition	No exit conditions
Exceptions	If the username and/or the password inserted don't exist in the database, an error message will be shown.

3.5.1.3.2 Confirmation/Refusal of the service

Name	Confirm/Refuse the service
Actors	Taxi
Entry conditions	Login successful
Event flow	1. The taxi • clicks on the received notification; • clicks on "Confirm" or "Refuse"; 2. the system • sends a notification to the user, if the taxi confirms the service; • otherwise searches the next available taxi in the queue.
Exit Condition	No exit conditions
Exceptions	No exceptions.

3.5.1.3.3 View information about the current ride

Name	View information about the current ride
Actors	Taxi
Entry conditions	Login successful
Event flow	1. The taxi clicks on "View information"; 2. the system loads from the database the information of the current ride.
Exit Condition	No exit conditions
Exceptions	No exceptions.

3.5.1.3.4 Send delay notifications

Name	Send delay notification
Actors	Taxi
Entry conditions	Login successful
Event flow	1. The taxi clicks on "Send delay notification"; 2. the system advises the user about the delay.
Exit Condition	No exit conditions
Exceptions	No exceptions.

3.5.1.3.5 View previous rides

Name	View information about the previous rides
Actors	Taxi
Entry conditions	Login successful
Event flow	1. The taxi clicks on "View history"; 2. the system loads from the database the information of all the previous rides.
Exit Condition	No exit conditions
Exceptions	No exceptions.

3.5.1.3.6 Where is the user?

Name	Where is the user?
Actors	Taxi
Entry conditions	Login successful
Event flow	1. The taxi clicks on the button "Where is the user?"; 2. the system sends an SMS to the user to tell him/her that the taxi is arrived.
Exit Condition	No exit conditions
Exceptions	No exceptions.

3.5.2 Sequence Diagrams

Some sequence diagrams of the system's most interesting aspects
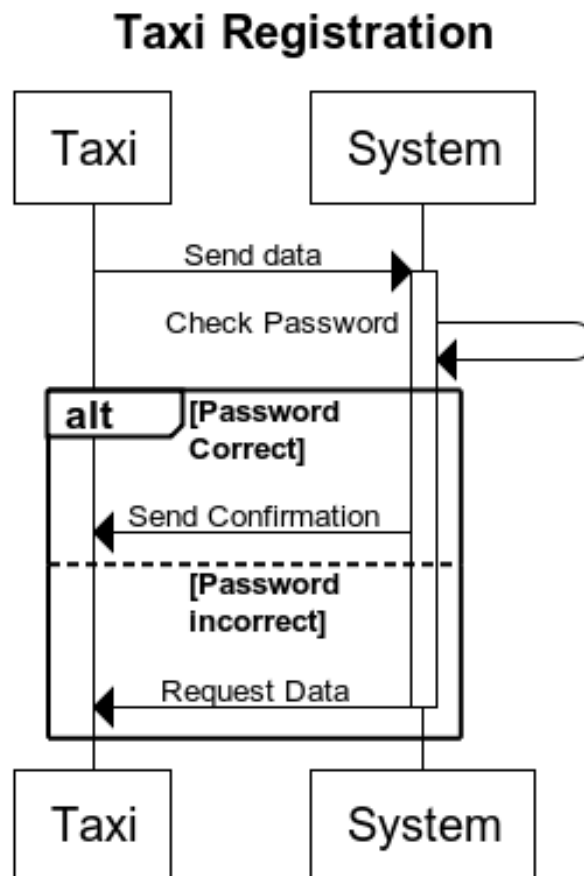
In order to register a taxi must send his identifier and a password to the system that localize him using the gps and inserts him into the appropriate queue. After this the system confirms the registration.

Short Term Reservation
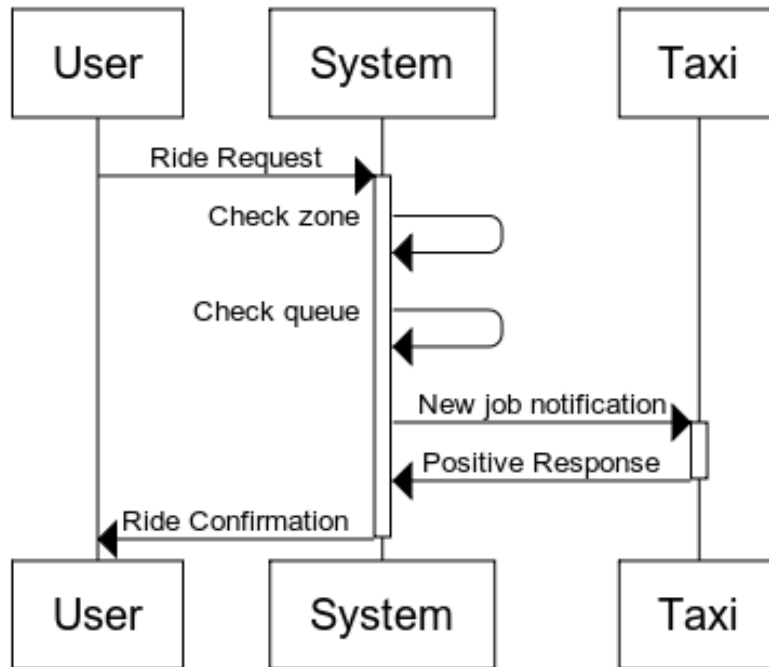
In order to request a short term reservation ride a user must send his address; date and hours must be omitted in this case. The system check where the address is located and the queue associated to that zone; then sends a notification to the first taxi in the queue and after the taxi confirmation, the system confirms the reservation to the user.

Short Term Reservation with multiple requests to taxis
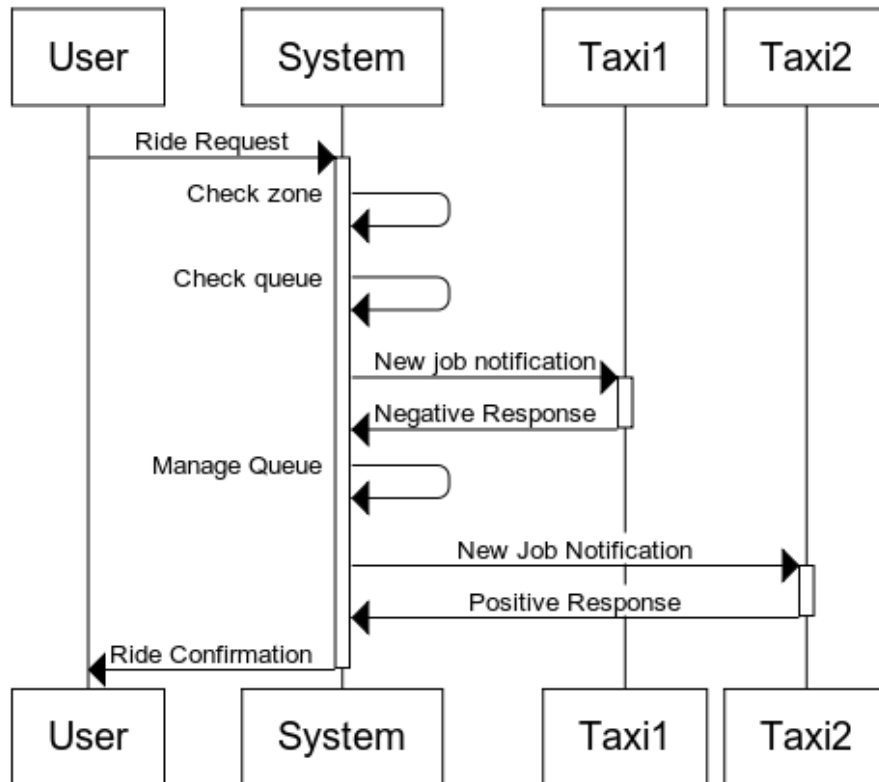
In order to request a short term reservation ride a user must send his address; date and hours must be omitted in this case. The system check where the address is located and the queue associated to that zone; then sends a notification to the first taxi in the queue and since the first taxi answers with a negative response the system puts the taxi at the end of the queue and asks to the second taxi in the queue. After the taxi confirmation, the system confirms the reservation to the user.

Short Term Reservation with taxi timeout
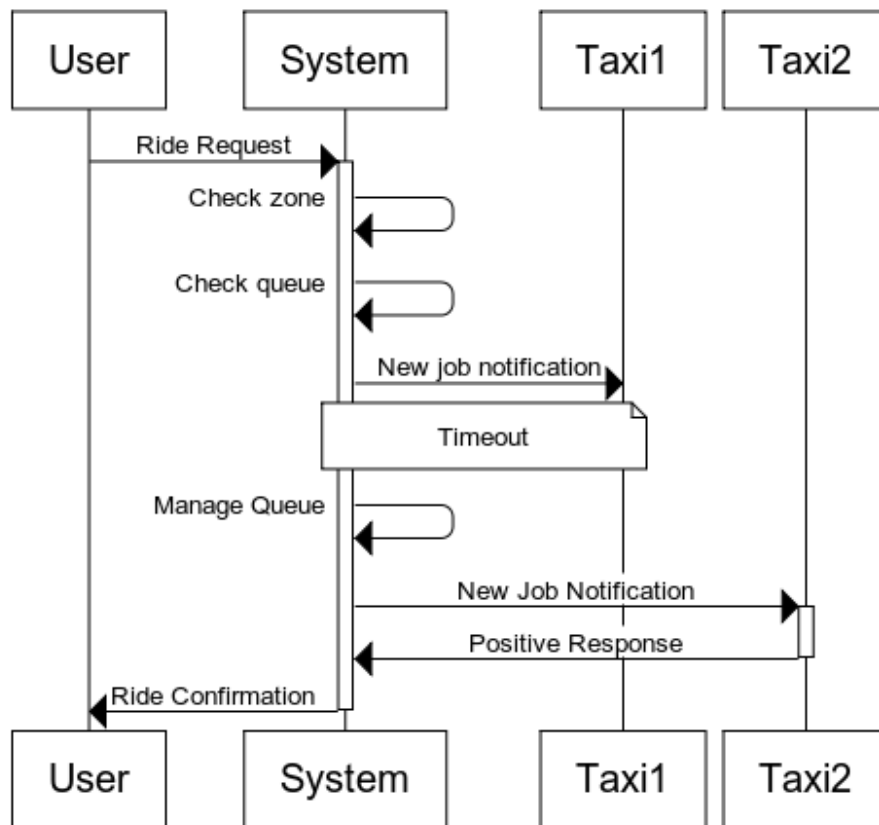
In order to request a short term reservation ride a user must send his address; date and hours must be omitted in this case. The system check where the address is located and the queue associated to that zone; then sends a notification to the first taxi in the queue and since the first taxi doesn't answers the system, after a timeout, puts the taxi at the end of the queue and asks to the second taxi in the queue. After the taxi confirmation, the system confirms the reservation to the user.

Long Term Reservation
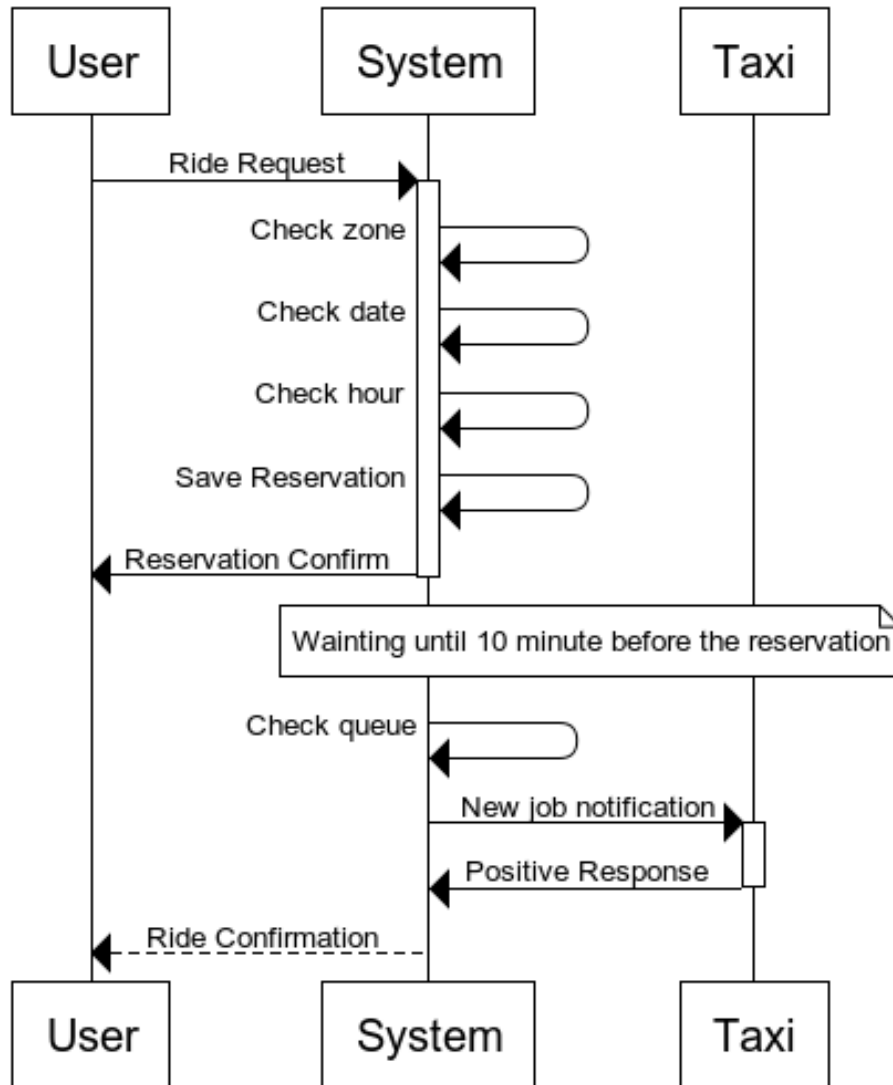
In order to request a long term reservation ride a user must send his address, the date and the hours of the reservation. Te system check where the address is located and stores the reservation i the database and confirms the user his reservation. Ten minutes before the ride the system check the queue associated with the address zone and send a notification to the taxi. After the positive answer by the taxi the system confirms the user his ride.

3.5.3 Class Diagram
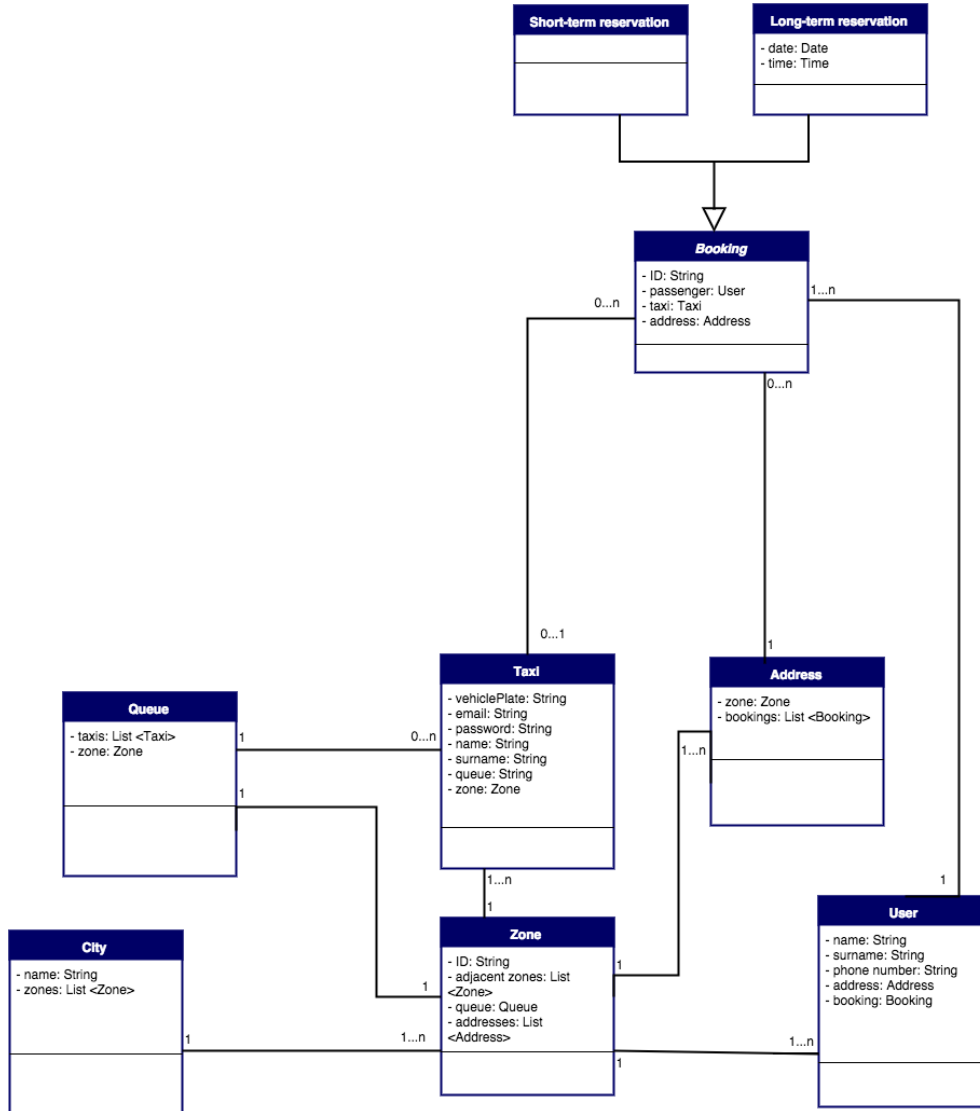
Starting from the definition of the "City" class, we are been able to define the other classes, specifying the relations between them. Only the class "Booking" is an abstract one and it is extended by two other classes, "Short-term Reservation" and "Long-term Reservation". We have written for the classes some private attributes, that are relevant, in our opinion. These attributes connect a class to another, with reference to the relations, defined before. We have omitted the classes methods because this Class Diagram lets us to have a first overview and because we will not implement this system.

3.6 Alloy

3.6.1 Signatures

```
//********************************
//          SIGNATURES
//********************************
sig City{
        contains: some Zone
}

sig Zone{
        address: some Address,
        uses: one Queue,
        encloses: some Taxi
}

sig Address{
        locates: set Booking
}

sig Taxi{
        rides: set Booking
}

sig Queue{
        include: set Taxi
}

sig User{
        booksShort: lone ShortReservation,
        booksLong: set LongReservation,
}

abstract sig Booking{}

sig ShortReservation extends Booking{}

sig LongReservation extends Booking{}
```

3.6.2 Facts

```
//********************************
//            FACT
//********************************
//ALL ZONES BELONGS TO A CITY
fact ZonesBelongToCity{
        all z: Zone, c, c2: City | (z in
           c.contains and z in c2.contains)
           implies c=c2
        all z: Zone | z in City.contains
}



//ALL ADDRESSES BELONG TO A ZONE
fact AddressBelongsToZones{
        all a: Address  | a in Zone.address
}



//ADDRESSES ARE DISJOINT
fact AddressDisjoint{
        all a: Address | one z: Zone | a in
           z.address
}



//A QUEUE CAN BE ONLY IN ONE ZONE
fact QueueOnlyInOneZone{
        all q: Queue | one z:Zone| q in z.uses
}

//IF A TAXI IS NOT AVAILABLE IN NOT IN A QUEUE
fact TaxiNotAvailable{
        all s: ShortReservation, t: Taxi, q:
           Queue | (s in t.rides) implies t not
           in q.include
}

//IN A TAXI IS AVAILABLE IS IN A QUEUE
fact TaxiAvailable{
```

```
        all t: Taxi, q: Queue | (t in q.include)
            implies (no s: ShortReservation  | s
            in t.rides)
}

//ONLY ONE USER CAN BELONG TO A SHORT RESERVATION
fact ShortReservationForOnlyOneUser{
        all s: ShortReservation, u, u2: User | (s
            in u.booksShort and s in
            u2.booksShort) implies u=u2
        all s: ShortReservation | s in
            User.booksShort
}

//ONLY ONE USER CAN BELONG TO A SHORT RESERVATION
fact LongReservationForOnlyOneUser{
        all l: LongReservation, u, u2: User | (l
            in u.booksLong and l in u2.booksLong)
            implies u=u2
        all l: LongReservation | l in
            User.booksLong
}

//A SHORT RESERVATIONS IS BOUND TO ONLY ONE TAXI
fact ShortReservationForOnlyOneTaxi{
        all s: ShortReservation, t, t2: Taxi | (s
            in t.rides and s in t2.rides) implies
            t=t2
        all s: ShortReservation | s in Taxi.rides
}

//A TAXI CAN SERVE ONLY A SHORT RESERVATION
fact TaxiRidesOnlyOneShortReservation{
        all s, s2: ShortReservation, t: Taxi |
            (s!=s2 and s in t.rides) implies s2
            not in t.rides
}

//A USER EXISTS ONLY IF HAS BOOKED SOMETHING
fact UserIfBookedSomething{
```

```
              all u: User | (u in
                  booksShort.ShortReservation) or  (u in
                  booksLong.LongReservation)
}

//A TAXI CAN BE ONLY IN ONE QUEUE
fact TaxisOnlyInOneQueue{
          all t: Taxi, q , q1: Queue | (t in
              q.include and t in q1.include) implies
              q=q1
}
 //A TAXI CAN BE IN ONLY ONE ZONE
fact TaxiOnlyOneZone{
          all t: Taxi, z,z2: Zone | (t in
              z.encloses and t in z2.encloses)
              implies z=z2
          all t: Taxi | t in Zone.encloses
}

//TO ONE BOOKING CORRESPONDS ONLY ONE ADDRESS
fact BookingOnlyOneAddress{
          all b: Booking, a, a2: Address | ( b in
              a.locates and b in a2.locates) implies
              a=a2
          all b: Booking | b in Address.locates
}

//A TAXI THAT IS IN A QUEUE IS IN THE ZONE
   RELATED TO THE QUEUE
fact ZoneTaxiQueueAreRelate{
          all t: Taxi, q: Queue, z: Zone | (t in
              q.include and q in z.uses) implies (t
              in z.encloses)
}

//NO TAXI IS BOUND TO A LONG RESERVATION
fact LongReservationsHaveNoTaxi{
          all l: LongReservation, t: Taxi | l not
              in t.rides
}
```

3.6.3 Assertions

```
//********************************
//              ASSERT
//********************************

//THIS ASSERTION VERIFIES THAT AN ADDRESS BELONGS
   TO ONLY ONE ZONE
assert AddressInOnlyOneZone{
        all a: Address, z, z2: Zone | (a in
           z.address and a in z2.address) implies
           z=z2
}

//THIS ASSERTION VERIFIES THAT A BOOKING HAS ONLY
   ONE ADDRESS
assert BookingOnlyOneAddress{
        all a, a2: Address | no b: Booking | (b
           in a.locates and b in a2.locates and
           a!=a2)
}

//THIS ASSERTION VERIFIES THAT A CITY MUST EXIST
   IF THERE ARE ZONES, TAXIS AND QUEUES
assert ExistsCity{
        all z: Zone, t: Taxi, q: Queue | ((z in
           City.contains) implies some City) and
           ((t in q.include) implies some Queue)
}

//THIS ASSERTION VERIFIES THAT A USER HAS BOOKED
   SOMETHING
assert  UsersHaveBookedSomething{
        all u: User | (u in
           booksShort.ShortReservation) or  (u in
           booksLong.LongReservation)
}

//THIS ASSERTION VERIFIES THAT A TAXI BELONGS TO
   ONLY ONE QUEUE
assert TaxiInOnlyOneQueue{
```

```
        all q , q1: Queue | no t: Taxi | (t in
            q.include and t in q1.include and
            q!=q1)
}

//THIS ASSERTION VERIFIES THAT A TAXI IS LOCATED
    TO ONLY ONE ZONE
assert TaxiInOnlyOneZone{
        all z, z2: Zone | no t: Taxi | (t in
            z.encloses and t in z2.encloses and
            z!=z2)
}

//THIS ASSERTION VERIFIES THAT A SHORT
    RESERVATION IS DONE BY ONLY ONE TAXI
assert ShortReservationForOnlyOneTaxi{
        all t, t2: Taxi | no s: ShortReservation
            | (s in t.rides and s in t2.rides and
            t!=t2)
}

//THIS ASSERTION VERIFIES THAT A LONG RESERVATION
    IS BOOKES BY ONLY ONE USER
assert LongReservationForOnlyOneUser{
        all u, u2: User | no l: LongReservation |
            (l in u.booksLong and l in
            u2.booksLong and u!=u2)
}

//THIS ASSERTION VERIFIES THAT A SHORT
    RESERVATION IS BOOKES BY ONLY ONE USER
assert ShortReservationForOnlyOneUser{
        all u, u2: User | no s: ShortReservation
            | (s in u.booksShort and s in
            u2.booksShort and u!=u2)
}
```

3.6.4 Checks and Preds

```
//******************************
//     CHECKS AND PREDS
//******************************
pred show(){
#ShortReservation > 1
#LongReservation > 1
}

check AddressInOnlyOneZone

check BookingOnlyOneAddress

check ExistsCity

check UsersHaveBookedSomething

check TaxiInOnlyOneQueue

check TaxiInOnlyOneZone

check ShortReservationForOnlyOneTaxi

check LongReservationForOnlyOneUser

run show for 5 but 7 Address
```

```
#1: No counterexample found. AddressInOnlyOneZone may be valid.
#2: No counterexample found. BookingOnlyOneAddress may be valid.
#3: No counterexample found. ExistsCity may be valid.
#4: No counterexample found. UsersHaveBookedSomething may be valid.
#5: No counterexample found. TaxiInOnlyOneQueue may be valid.
#6: No counterexample found. TaxiInOnlyOneZone may be valid.
#7: No counterexample found. ShortReservationForOnlyOneTaxi may be valid.
#8: No counterexample found. LongReservationForOnlyOneUser may be valid.
#9: Instance found. show is consistent.
```

3.6.5 Graph
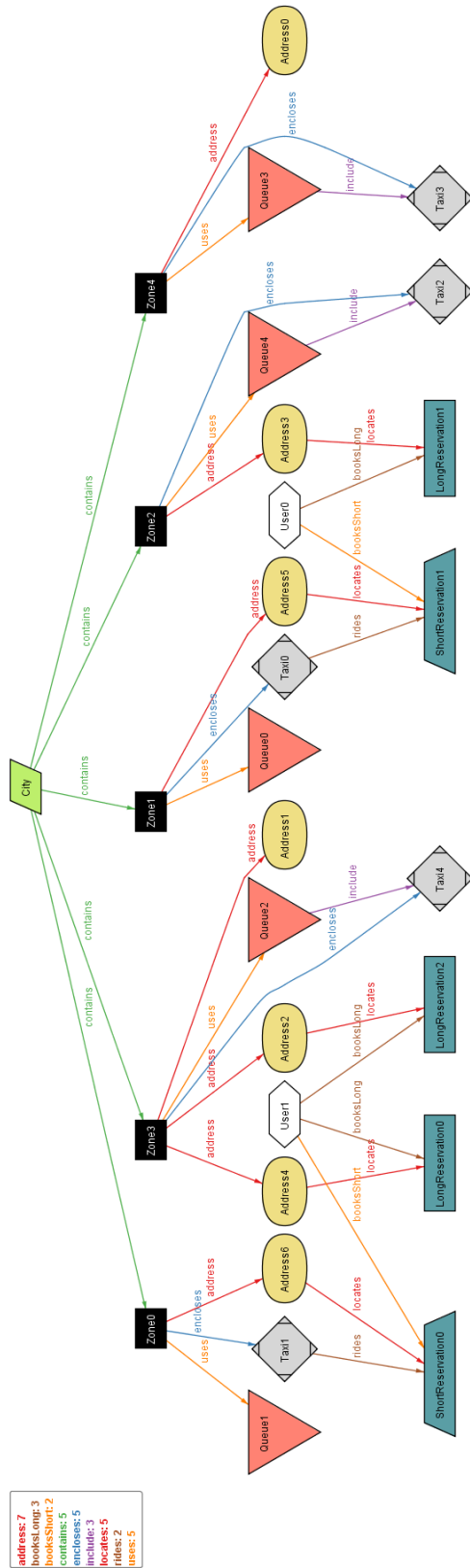
You can find the full-resolution version in the repository.

3.7 Non-Functional Requirements

- The system will provide a secure access to their personal page for all the taxi drivers, using a username and a password.

- The system should be working 24/24 7/7, with an availability of at least 98% (mean down time in a year: 7.30 days).

- The UI of both mobile and web applications has to be very responsive, and it should allow to execute actions on the system spending less time as possible.

4 Appendix

4.0.1 Software and Tools Used

- ShareLatex `http://www.sharelatex.com`, TeXstudio `http://www.texstudio.org/`, MiKTex `http://www.miktex.org/` to redact this document;

- Draw.io `http://www.draw.io` to draw use case diagrams, mobile app mockup and logo;

- Pencil `http://pencil.evolus.vn/` to draw website mockup;

- Web Sequence Diagram `http://www.websequencediagrams.com/` to draw the sequence diagrams.

4.0.2 Hours of Work

In order to write this document, we have done the following hours of work:

- Paolo Paterna: 40 Hrs;

- Lara Premi: 40 Hrs.

5 Revision

After some considerations, we have modified some stuff to the RASD document.

- 13/01/2016

  - We have added the last assumption, located in the list of the subsection **Assumptions**.
  - We have added the abbreviations of the goals and we have specified them in the corresponding list of the subsection **Abbreviations**.
  - We have added a use case table, in particular the last one named "Where is the user?".
  - We have added the link called "Where is the user?" in a screen of the taxi smarthphone.
  - We have added the use case named "Where is the user?" in the use case diagram of the taxi.

- 07/02/2016

  - We have modified the section "Software Interfaces", eliminating PHP.