Politecnico di Milano

Department of Electronics, Information and Bioengineering

Master Degree course in Computer Science Engineering

Design Document (DD)

myTaxiService

Instructor: Prof. Elisabetta Di Nitto

Authors:	*Code:*
Luca Luciano Costanzo	789038
Simone Disabato	852863

November 26, 2015

Contents

1 Introduction **1**

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, Abbreviations 3

1.4 Document Structure 3

2 Architectural Design **4**

2.1 Distinctions between various kind of Users and Clients 5

2.2 High level components and their interaction 6

2.3 Component view 8

2.3.1 Data tier 8

2.3.2 Provider tier 11

2.3.3 Presentation tier 11

2.4 Deployment view 12

2.5 Runtime view 13

2.5.1 UX Diagram 14

2.5.2 Sequence Diagram 18

2.6 Component Interfaces 24

2.7 Selected architectural styles and patterns 24

3 User Interfaces Design **25**

3.1 Registration/Login 25

3.2 Personal Information Management 26

3.3 Driver functionalities 28

3.4 Zerotime Ride 29
3.5 Future Ride 31
3.6 Other User Functionalities 32

4 Other Info **35**
4.1 Working hours 35
4.2 Tools 36

List of Figures

2.1 High level architecture 6
2.2 Logical schema of the database 10
2.3 UX Diagram from the Starting Page. 14
2.4 UX Diagram for Profile Management. 15
2.5 UX Diagram for the Ride booking. 16
2.6 UX Diagram for Check the Reservations. 17
2.7 UX Diagram for the Driver functionalities. 17
2.8 Sequence Diagram for the Registration. 18
2.9 Sequence Diagram for the login. 19
2.10 Sequence Diagram for the Profile Management. 19
2.11 Sequence Diagram for Start Waiting Time. 20
2.12 Sequence Diagram for the Workshifts Management. 21
2.13 Sequence Diagram for Check the Reservations. 22
2.14 Sequence Diagram for the Zerotime Ride. 23
2.15 Sequence Diagram for the Future Ride. 24

3.1 Login page into website. 26
3.2 Login page into mobile application. 26
3.3 Registration page into website. 27
3.4 Registration page in mobile application. 27
3.5 Personal Information Management page into website. 27
3.6 Personal Information Management page in mobile application. . 27
3.7 The Start Waiting Time page. 28
3.8 The request for a ride to a driver page. 28
3.9 The workshifts management page. 29

3.10 Zerotime ride request part1 into mobile application. 30
3.11 Zerotime ride request part2 into mobile application. 30
3.12 Zerotime ride request part3 into mobile application. 30
3.13 Zerotime ride request into website. 30
3.14 Future ride request part1 into mobile application. 31
3.15 Future ride request part2 into mobile application. 31
3.16 Future ride request part3 into mobile application. 32
3.17 Future ride request into website. 32
3.18 View reservation into mobile application. 33
3.19 Modify reservations into mobile application. 33
3.20 Check reservation into website. 33
3.21 Read the alerts into mobile application. 34
3.22 Read the alerts into website. 34

Chapter 1

Introduction

This chapter provides a short description about the purposes and the scope of this document. After that, a glossary is given to help the readers to understand the meaning of each word or acronym used in this document. At last, the main structure of the document is shown.

1.1 Purpose

The design document of myTaxiService aims to describe all the aspects concerning the architecture of the system. The introduced tiers or levels into that architecture are described and studied more in details, explaining the reasons for the single choices and the interactions between them.

After that, the key algorithms of this system are shown in pseudo-code to suggest and describe the real implementation of the code. Finally, the last purpose is to give the readers the idea of final applications (both MA and WS) using mockups.

1.2 Scope

Users, once registered, are able to ask for an immediate ride or to book one of them.

The system provides the user with a complete map of the city and its suburbs within the taxi service is available. The current position of the user is obtained by localization services of the user's smartphone if it's possible, otherwise the user notifies his position directly on the map with a marker or by a searching box. The destination is also chosen either graphically or by a research. The user can view the suggested path and then he must confirm the request.

When a user asks for a ride, the system checks the availability of a taxi driver near the current position, by splitting the city in several areas and using a FIFO (First In First Out) policy to manage the assignment of the ride's driver. The selected driver can accept or decline the ride. In the former case the system informs the user about waiting time, estimated travelling time, prices and cab car-code.

The system gives also the possibility to book a ride with at least two hours in advance. As the user does when he asks for a ride, he selects the desired starting venue and the destination. Afterwards, the system gives a calendar where the customer can choose the date (at most 30 days in advance) and the starting hour. Ten minutes before the meeting time the system starts all the operations described before in order to assign a taxi-driver.

A reservation from the app or the website can be undone until the system confirmation of the availability of a taxi, while a booking can be cancelled at most fifteen minutes before the meeting hour.

After those deadlines the ride is considered bought by the customers and an eventual absence on the established venue forbids other possibilities to book or to take a ride.

1.3 Definitions, Acronyms, Abbreviations

MA
: acronym for the mobile application.

Mockup
: a simple graphical representation.

WS
: acronym for the website of the system.

1.4 Document Structure

The chapter 2, called Architectural Design, describes all architectural choices. First, the high hierarchy of that architecture is shown and the interactions between its components are explain. Then, for each defined level or tier a standalone paragraph is dedicate to present all its characteristics.

The **??**, called Algorithm Design, points out the key algorithms of this system. In particular, these algorithms are the ones which manage the cabs' queues and the city areas and the ones which manage the special case that happen when no taxi are available in the desired area.

The chapter 3, called User Interface Design, describes all the graphical interfaces by using mockups.

Finally, the **??** is dedicated to point out the links between requirements presented in the RASD document and the decisions taken and shown in this document.

Chapter 2

Architectural Design

In this chapter the complete architecture of myTaxiService is shown with various levels of description. In the section 2.2 there is a global view and the interactions between all the components are described.
The data tier is illustrated in section 2.3 with all related policies and entities. Then the other tiers are characterized using different diagrams.
In the section 2.4 the deployment of each components is illustrated (for instance the data component is sit in a different place with respect to the other component? It is replaced twice or more? And similar question will have an answer).
In section 2.5 the view level is defined. The interactions between all kinds of user and the system are described using UX diagrams and sequence diagrams that display the order in which each screen is visualized. Besides, the mockups of these screens are shown in chapter 3.
A standalone paragraph, the section 2.6, is dedicated to list all interfaces, both internal (between two components) and external.
Finally, in the section 2.7 the design patterns used to develop myTaxiService are described first in general case. After that, all the changes needed to adapt this design patterns to our system are characterized.

2.1 Distinctions between various kind of Users and Clients

The "visible architecture" of myTaxiService is very varied. The term visible is referred to various user interfaces, so what the users can see when they are using the system.
On the other hand, we have said that myTaxiService is varied because it has two principal version (MA and WS) and for both of them there are a few levels of specialization, according to the kind of the user. All of them are explained in this paragraph.

The WS version is shown into a browser, so there is no client application that can be used. Hence, all the pages are loaded into the server and then they are sent to the client browser.
Instead, the MA is a client application and it has different ways to communicate with the server. All the aspects concerning these differences are explained better during the descriptions of the architecture's components that handle the clients.

The cab company is the special user who administrates the service. Obviously, it has a command center at its headquarters where it can control both the system and the service situation. Hence its special functions are not implemented neither in the MA nor in the WS, but they can directly access the server using private keys and reserved terminals.
A customers can use both the MA and the WS to enjoy his functionalities. No particular cases or restriction are reserved to them.
When a driver logs into the service, we suppose that it is working, so his special functionalities are developed and implemented only for the MA. In fact no driver carries a computer with an internet connection on his taxi and uses it. On the other hand, since the driver is also an user, it can use the WS, but here he has only the user functionalities due to the reason shown above.

2.2 High level components and their interaction

The main structure of myTaxiService can be described as a Service-Oriented architecture (see section 2.7 to have a detailed description of this design pattern). In addition to this idea the Model-View-Controller paradigm is applied in order to develop a good-programmed, reusable and easy-maintainable system.

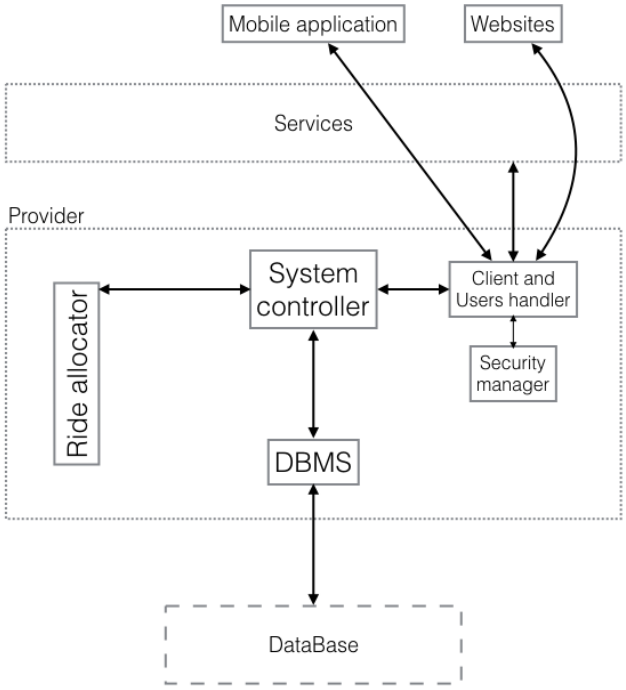

Figure 2.1: High level architecture

In the figure 2.1 is shown the architecture. At the bottom of the figure there is the Data tier which contains only the database (a component that stores all data about the customers, the rides and the system).

The central part of the picture contains the provider, the big component that gives the service. At this level of description we can look in the provider. This component is split into five parts. The DataBase Managment System (DMBS) manages all communications with the Data tier. The Ride Allocator has only

one role, to assign and to handle the ride, both future and zerotime. The system controller is the "heart"of the provider: it involves the ride allocator when is needed, it gives all the services of the system, it provides the DBMS with the data to be stored (or asks him some data to be found) and communicates with the Client and Users Handler. The Client and User Handler, as its name said, handles the communications with the clients and administrates the Users, so which services can be shown and selected by them.

The third tier is a presentation level. In the Service-Oriented pattern this level represents the available services. Here, the clients can see the services and select the desired one.
Finally, at the top of the image, there is a representation of the two kind of clients, the WS and the MA.

All the tiers are also distinguished into the three parts of MVC-paradigm. The Data tier is the model[1]. The provider is also the controller: even if the controller can be seen as the component that manages the interaction between the view and the model, in this application we consider other "handler"as part of controller because they encapsulate rules and action codes. The view is the union between the presentation tier (here the services are only shown to client, the "decisions"are taken by the provider) and the clients.

Up to now, we have introduced all the main components or tier of our system, without focusing on the communications between them, so we'll dedicate a little part of this paragraph on this argument.
All the communications between the model and the provider are managed by the DBMS that has store policies and finds the required data. The system controller is the central node in communication because it is the only way to all the other components of the provider. The Client and User Handler is the link between the provider and the clients. It checks the type of the current user and, as

[1] see the Alloy section or the UML class diagram into the RASD to see it. However, in section 2.3.1 it is studied again.

consequence, the available services that he can uses. When a user asks a service, he binds the request asking for the related function to the System Controller. Finally, it checks the type of client, because they have two ways of communication. When a user is connected with the WS, the Client and User Handler load the pages into itself and then, using the HTTPS protocol sent them to the client. Besides, the communications with the MA is implemented using sockets. The implementation of these ways of communication with the clients is realized by a common interface (see the section 2.6 for further information).

2.3 Component view

2.3.1 Data tier

This paragraph shows the Logical schema of the database in figure num for the application.
The underlined attributes are the primary keys of the tuples, while the bold attributes are the reference keys.

- rides (rideID, **passenger**, **driver**, **departure**, **destination**, departureTime, arrivalTime, creationDate, isFuture) :
  it contains both the zerotime rides and future rides, the field isFuture has value 1 when it's a future ride, 0 when zerotime ride.
- users (userID, email, password, name, surname, taxcode, birthday, cityOfResidence, isDriver, registrationDate, activationCode, activated) :
  it contains the general and fundamental information of all the users. The field isDriver has value 1 when the user is a driver, 0 otherwise. The field activated has value 1 if the user has completed the registration process by confirming on the link with his activation code sent to his e-mail address, 0 otherwise.
- drivers (driverID, **userProfile** ,cabCarCode, **cabCompany**) :
  it contains all the information of the drivers. The field userProfile is the reference key to the record in the users table which contains all the other general information of the driver.

- cabCompanies (cabCompanyID, name) :
  it contains all the cab companies that use myTaxyService.
- positions (positionID, gpsCoordinates, **address**, civicNumber) :
  it contains the stored positions for the rides (departure and destination) with the reference to the corresponding address in the table addresses and optionally the gps coordinates and the civic number.
- addresses (addressID, city, street, area) :
  it contains all the existing streets and cities covered by myTaxyService, with the reference to the area which they belong to.
- areas (areaID, name) :
  it contains all the areas covered by myTaxyService.
- driversWaiting (**area, driver**, driverAddedTime) :
  it contains all the available drivers waiting for a ride request. The tuple "area,driver" is unique because a driver cannot be available in more than one area at time. The field driverAddedTime contains the date and time when the driver is added to the list of available drivers for a certain area, it is useful to sort the list with the desired order (a FIFO queue in our application).
- workShifts (workShiftID, **driver**, weekDayNumber, startingTime, endingTime) :
  it contains all the work shifts of a driver. weekDayNumber is the number of the day of the week (between 1 and 7, 1 is Monday, 7 Sunday). startingTime and endingTime represent a continuous time range of a work shift of a driver in a certain day of the week.
- alerts (alertID, **user**, receptionDate, message) :
  it contains all the alerts that a user has received. "user" field is the reference to the receiver of the alert.

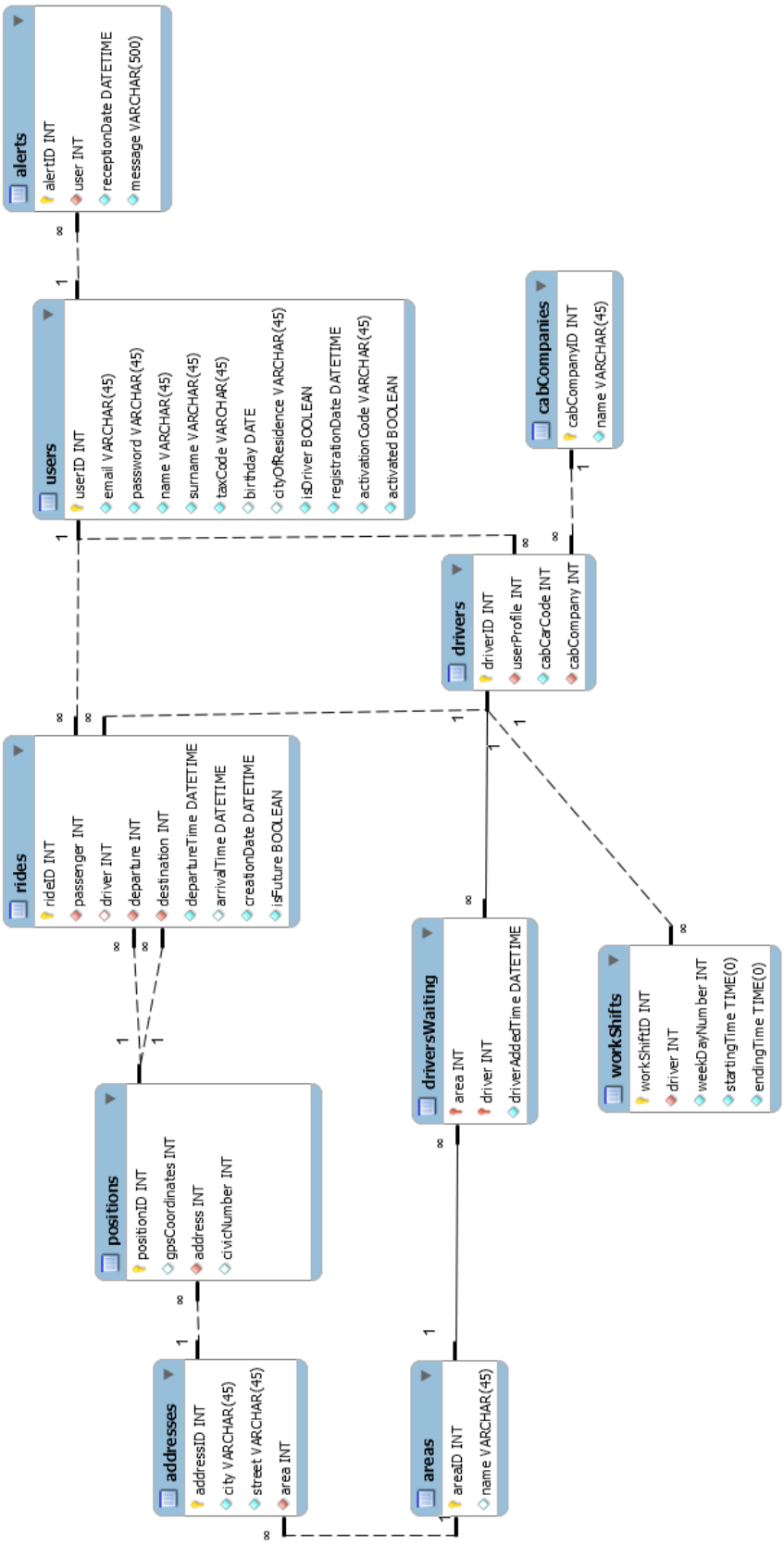

Figure 2.2: Logical schema of the database

2.3.2 Provider tier

DO IT DO IT DO IT

is a driver or a cab company administrator other special credentials will be asked to increase security).

- *Ride services.* This macro section contains all the functionalities related to the rides, so the characteristic ones of myTaxiService. The zerotime and future rides' reservation, the possibility to view or to modify the personal rides and the reading of user's alerts belong to this area and they are all shown together.
  The Client and User Handler shows these services to all logged users (except for the administrators).

- *Driver services.* Driver services is a section reserved to the drivers logged into the MA, for the reason presented in section 2.1. The functionalities give here are the management of work shifts, the notifications of availability and the reading of reserved alerts (for instance, the request for a ride or the confirmation of a ride).

- *Administration services.* This section, for security reasons, can be accessed only with administrator credentials (they consists in a couple name and password and in the use of an electronic card) into the company's headquarters on the WS version. The Client and User Handler shows into this section the following functionalities: create or remove a driver profile, notification for all the users (usually, for general information or problems with the service) and checks some service information (for instance anomaly values of taxi queues).

2.4 Deployment view

cosa scrivere qui? c'Ã¨ un diagramma che mostra i vari componenti software/hardware
e quindi dobbiamo fare un diagramma contenente le classi principali del controller

2.5 Runtime view

In this paragraph we'll show the main features on the runtime view, which means the UX diagrams and the sequence diagrams.
The UX diagrams display the structure of the myTaxiService's screens (input forms, available functions and shown data) and the effects of each function (the next screens viewed). The sequence diagrams show the sequence of functions's calling and screens to benefit of a functionality.

An important note is the following: the functionality Read The Alerts[3] is not shown here. It is performed as a simple screen on the Homepage that redirect to another screen with only the list of alerts (composed by a title and a written text).

Finally, the two version of myTaxiService (WS and MA) have no differences on the visualization of the screen. In fact, the screen has the same name and the same purpose, but they are adapted to the particular version where they are displayed.

[3] see the RASD for further information.

2.5.1 UX Diagram

Starting Page

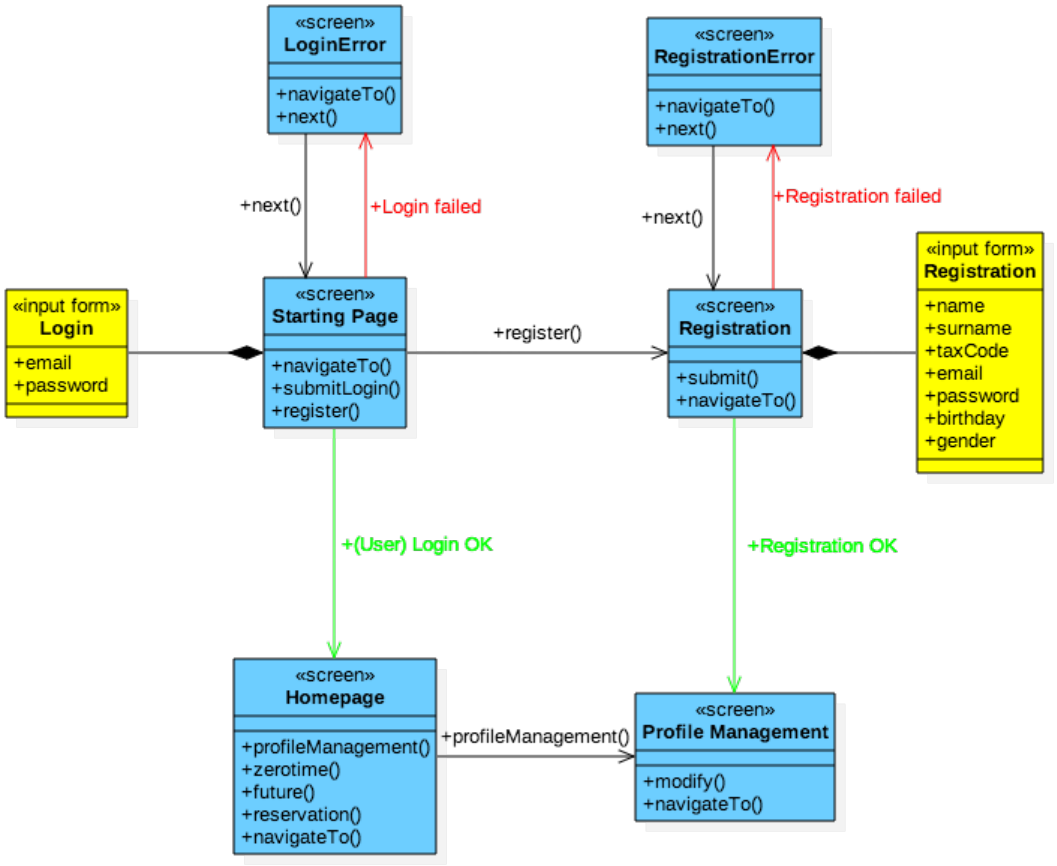

Figure 2.3: The UX diagram from the Starting Page. The main functionalities available from this screen are displayed here.

Profile Management

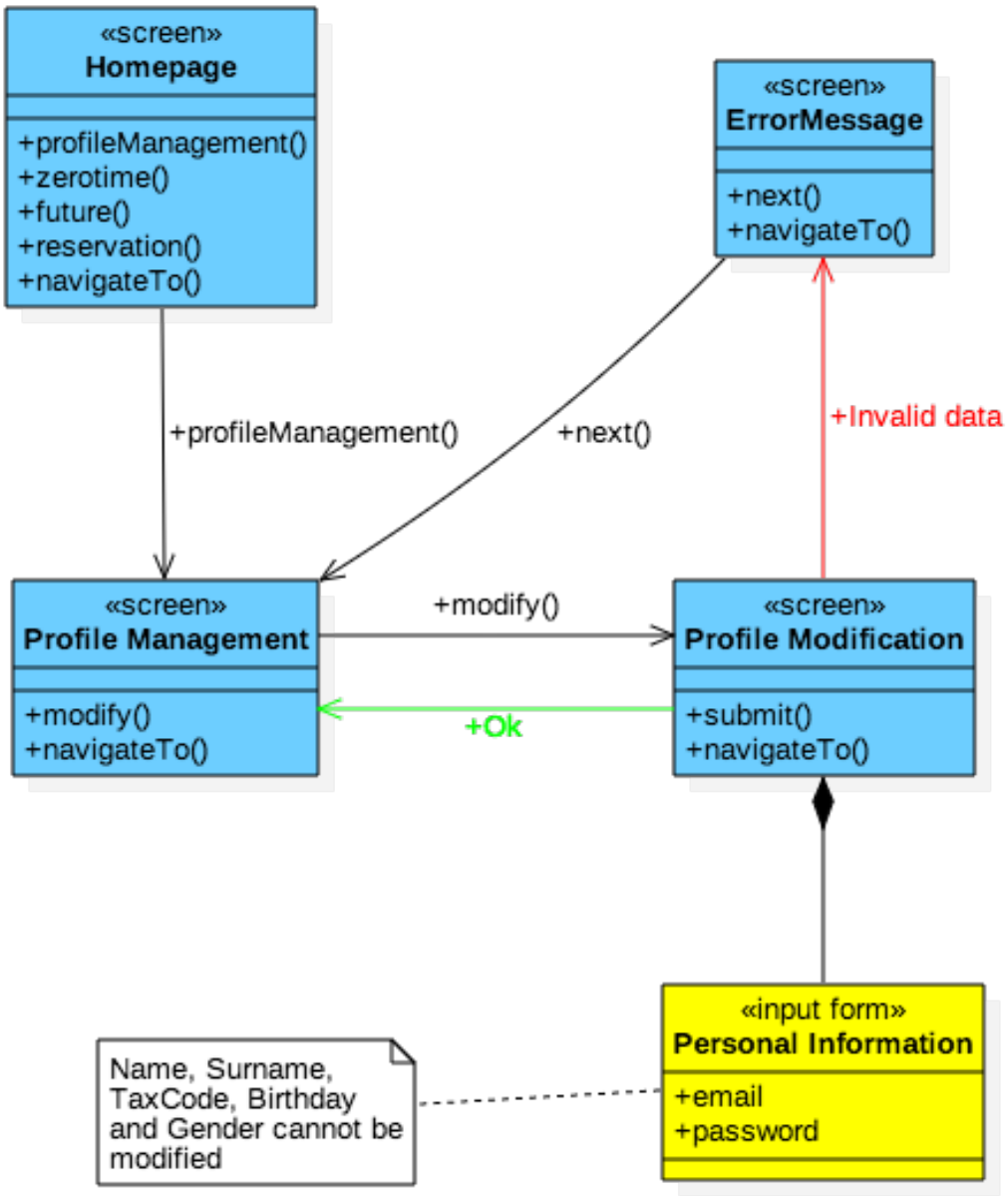

Figure 2.4: The UX diagram for the management of the personal profile. All the possible operations are displayed here.

Zerotime and Future Rides

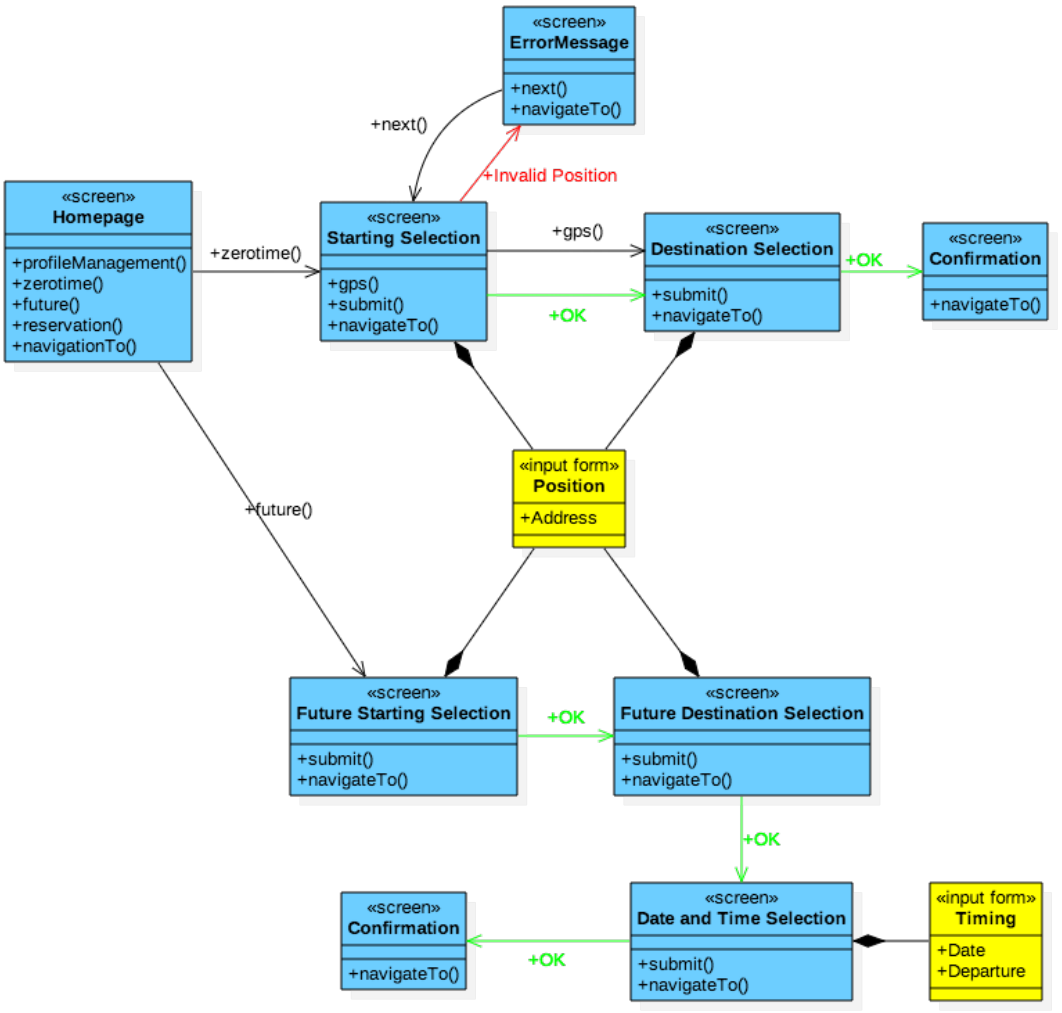

Figure 2.5: The UX diagram for both the booking of a future ride and the asking for a zerotime ride, starting from the homepage.
In order to make the diagram easy to understand, not all the error screens are inserted into the diagram: only the first one is shown.The other ones are similar and where there is an input form.

Check The Reservations

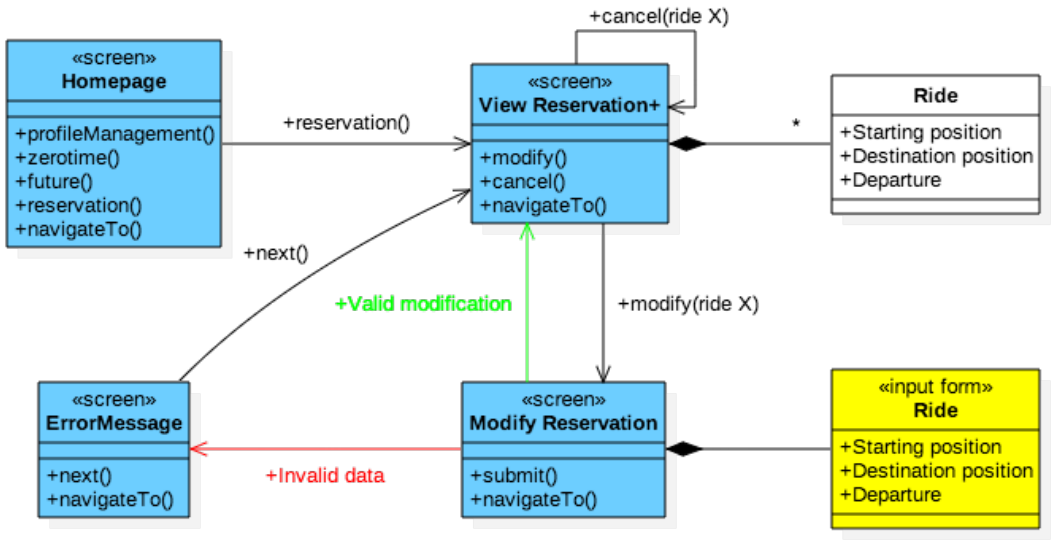

Figure 2.6: The UX diagram for the management of the personal reservations.

Driver Functionalities

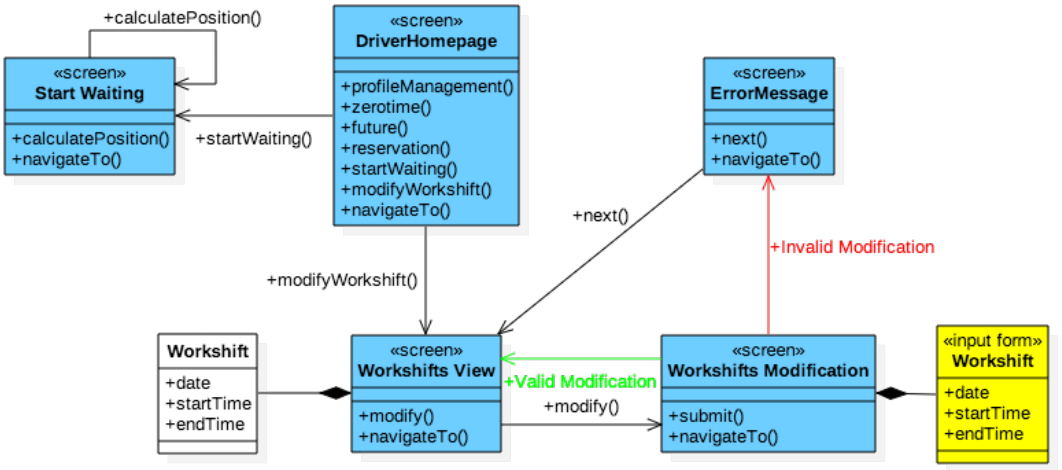

Figure 2.7: The Ux diagram for the Driver functionalities.

2.5.2 Sequence Diagram

In this section are displayed the sequence diagram of myTaxiService run-view. In each diagram where the actor is an user and not a visitor of the website there is always an edge with “Login Procedure”that can be misunderstood: the login to the websites it is required to perform the desired action, but if it is already done (for instance when the user is on his homepage after another operation) the login it is not requested again.

Registration

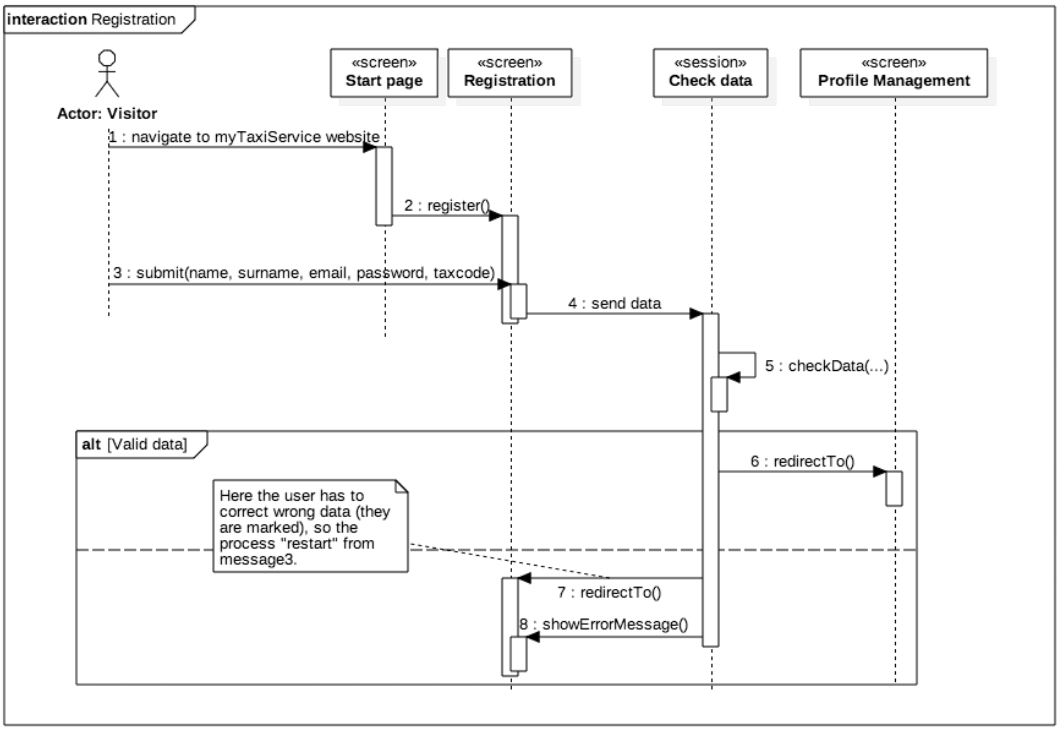

Figure 2.8: The sequence diagram for the Registration.

Login

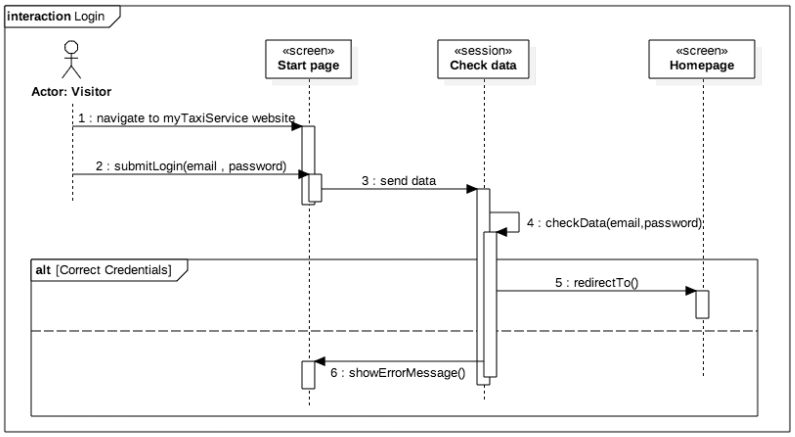

Figure 2.9: The sequence diagram for the Login.

Profile Management

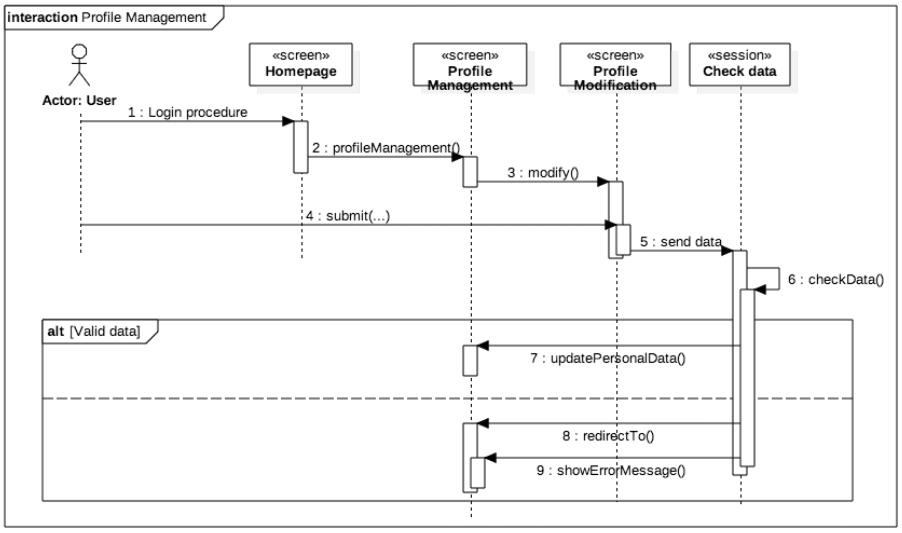

Figure 2.10: The sequence diagram for the management of the personal profile.
The only data that can be modified are the email and the password used to log into the system, while name, surname, tax code, gender and birthday are not modifiable by definition.

Start Waiting Time

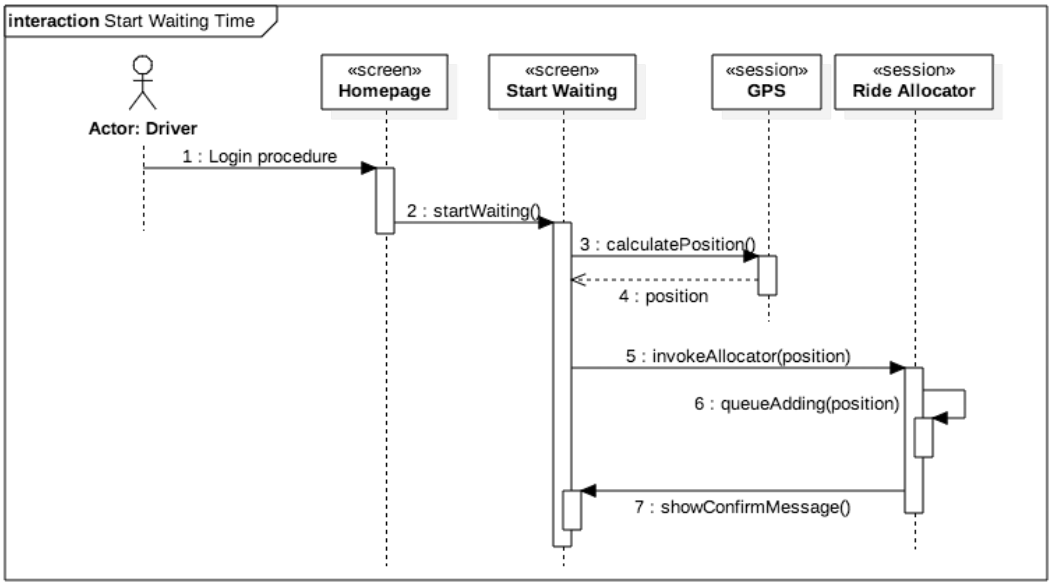

Figure 2.11: The sequence diagram for the notification of waiting time's start.

Workshifts Management

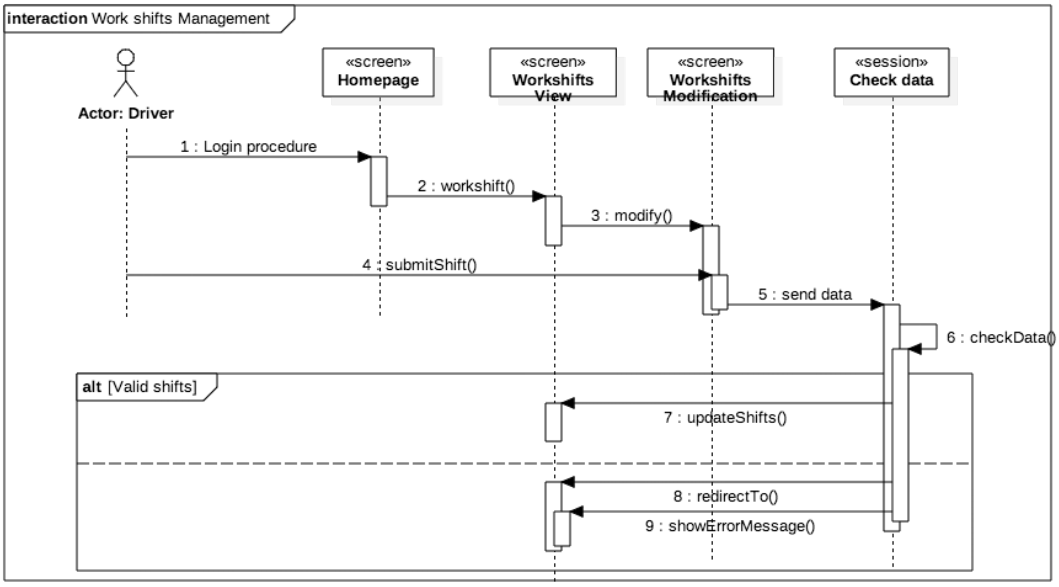

Figure 2.12: The sequence diagram for the management of the driver's workshifts.

Check The Reservations

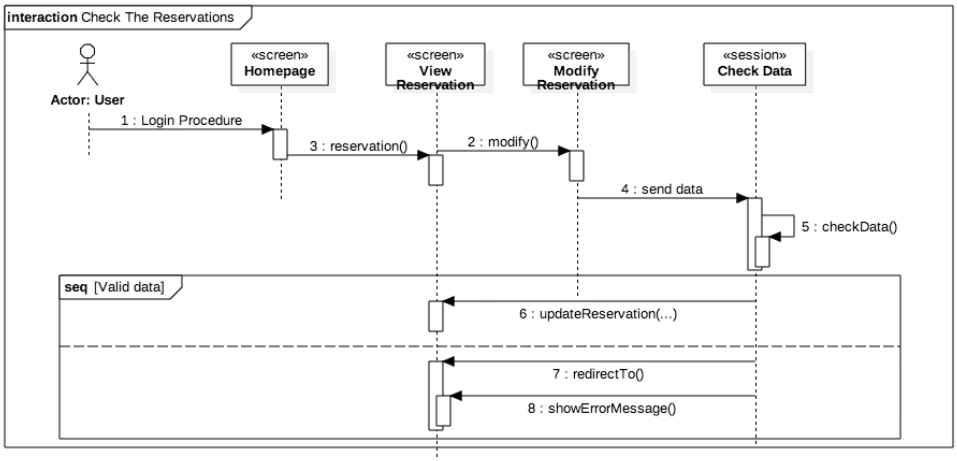

Figure 2.13: The sequence diagram for the management of personal booking.

Zerotime Ride

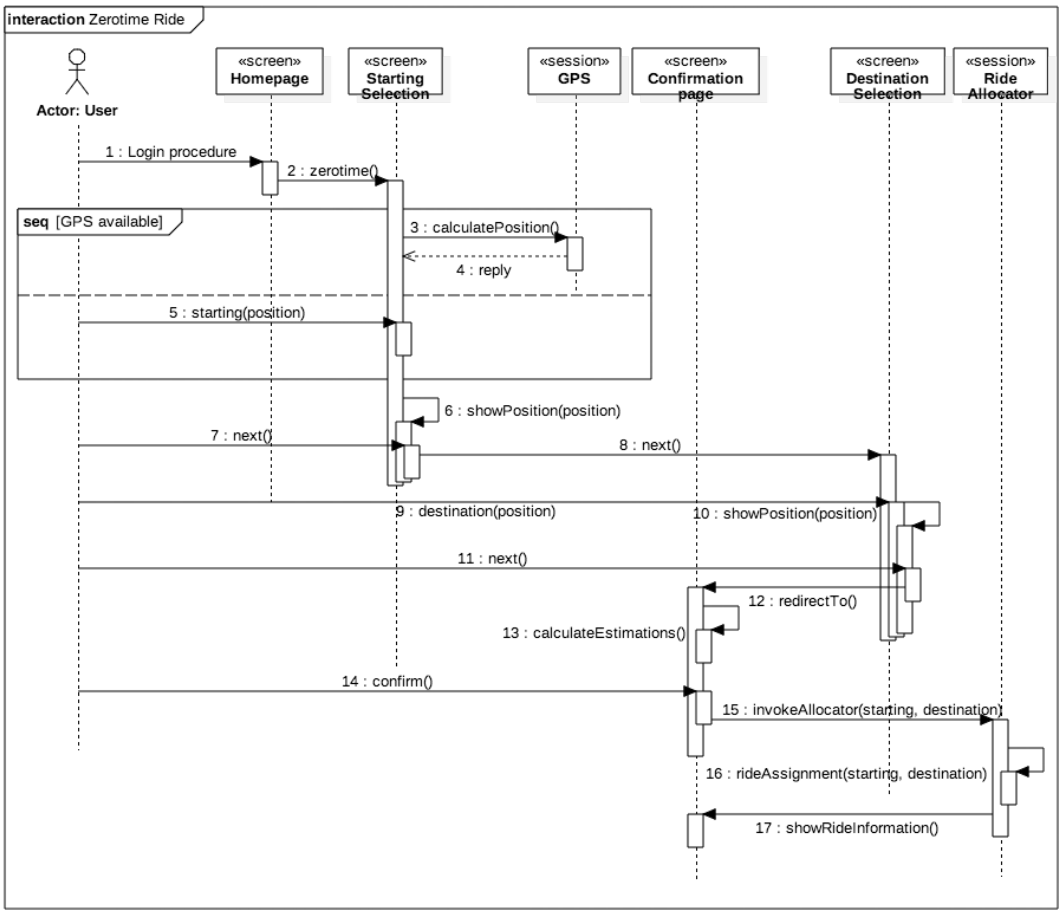

Figure 2.14: The sequence diagram for the asking for a zerotime ride.
In order to make the diagram easy to understand, the errors on inserting the positions are not modelled.

Future Ride Booking

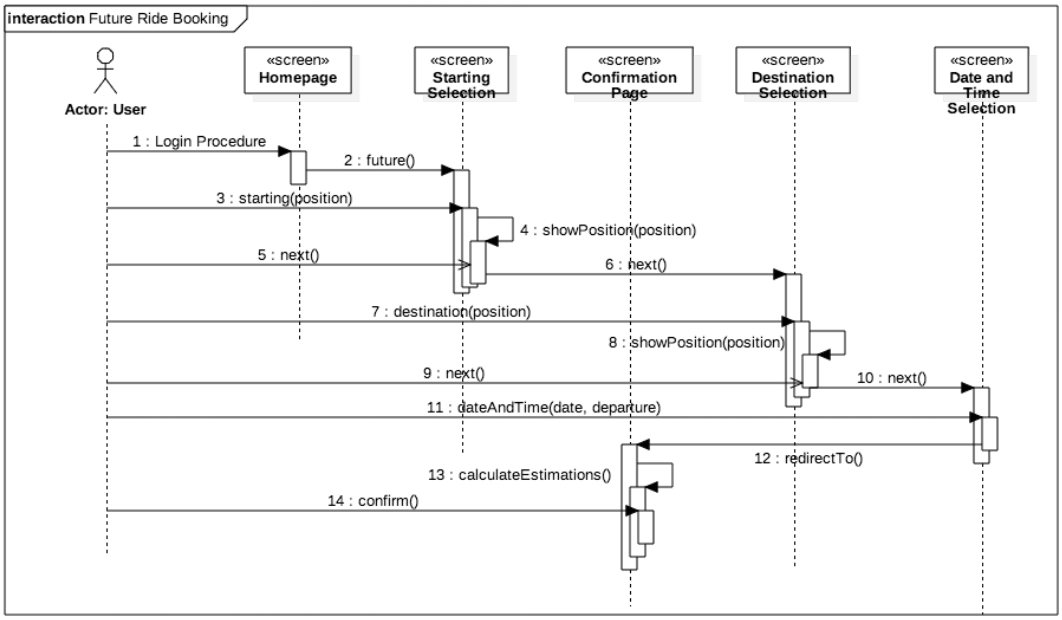

Figure 2.15: The sequence diagram for the booking of a future ride.
In order to make the diagram easy to understand, the errors on inserting either the positions or the departure are not modelled.

2.6 Component Interfaces

2.7 Selected architectural styles and patterns

EVENTBASED per ora

Chapter 3

User Interfaces Design

In this chapter are shown the main mockups of myTaxiService, due to present the graphical user interfaces of the system for each functionality. All the mockups are displayed with an accurate description to prevent misunderstanding and to recall the functionality.

3.1 Registration/Login

When an user connects to the WS or opens the MA, the first screens is the login page. For both the versions, there is a button to redirect to the registration page, if the user is not registered yet.
The figure 3.1 is the WS version where the registration button is made by a link on the phrase “Not Registered yet? Click here”. At the same time the user can insert his credentials into the specific boxes.
The figure 3.2 is the corresponding mockup for the mobile application: at the application opening the user can inserts his credentials or clicks on the registration button.

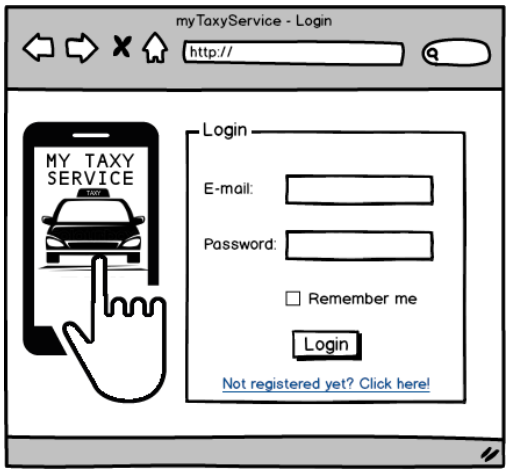

Figure 3.1: Login page into website.

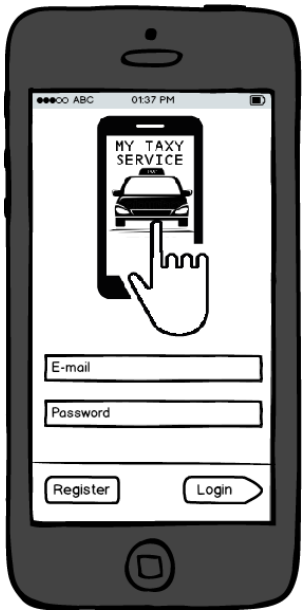

Figure 3.2: Login page into mobile application.

A non registered user has to enrol into the system, so he clicks on the corresponding button and he is redirected to that page. The figures 3.3 and 3.4 shows the mockups for this: into the MA the box labels are written into them and disappears when the user fills in; into the WS the labels are shown near the corresponding one.

3.2 Personal Information Management

The figures 3.5 and 3.6 show the mockups for the management of personal information.
The only information that can be modified are the email, the password and the city of residence. In both the myTaxiService's versions the other information are shown into a non-modifiable boxes (so they have a different color, typically grey, to mark this property).

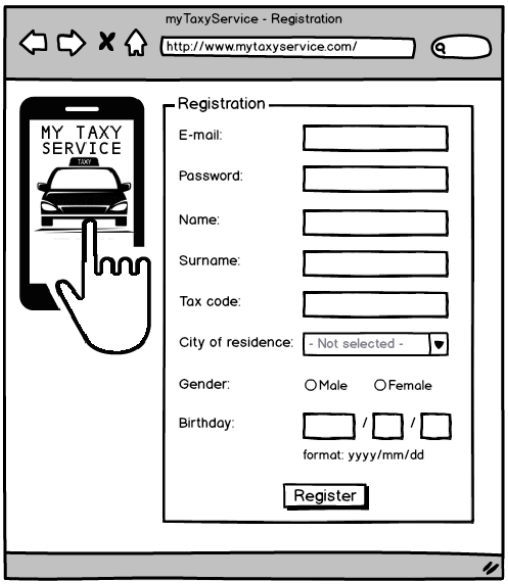

Figure 3.3: Registration page into website.

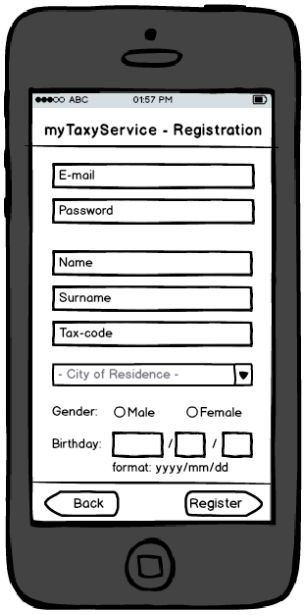

Figure 3.4: Registration page in mobile application.

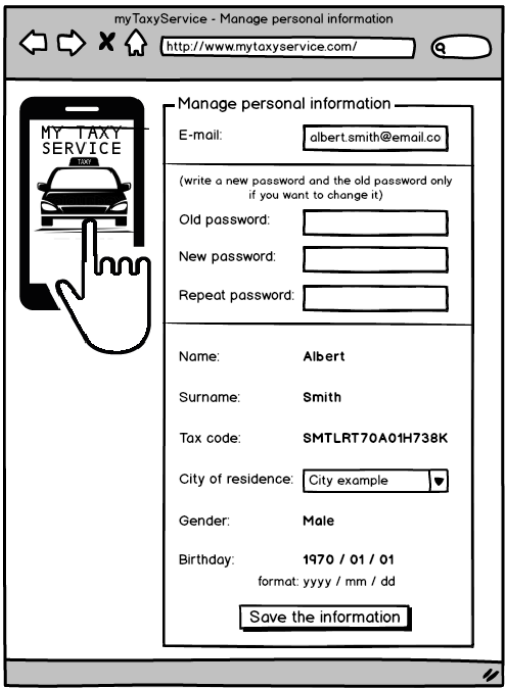

Figure 3.5: Personal Information Management page into website.

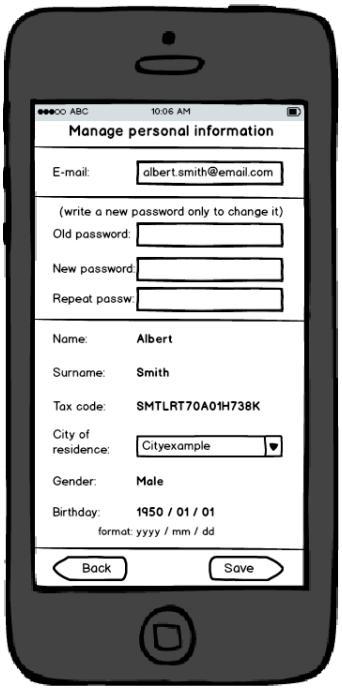

Figure 3.6: Personal Information Management page in mobile application.

3.3 Driver functionalities

The functionalities reserved for the cabmen are developed only for the MA version of myTaxiService: the reasons are explained in section 2.1. When a cabman has to notifies the system that it is starting his workshift or he has ended a ride, so he is now waiting for a new ride, he uses the Start Waiting Time functionality that is shown in figure 3.7: the displayed position is calculated by GPS.

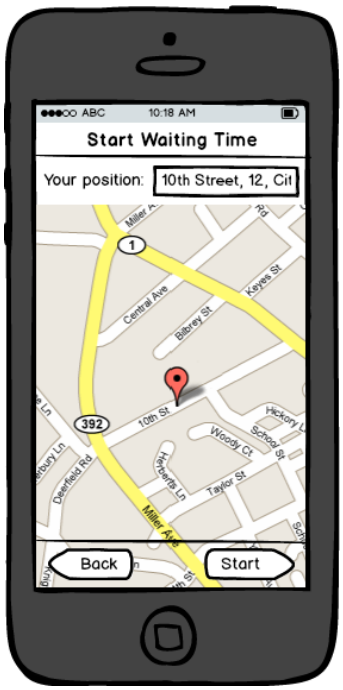

Figure 3.7: The Start Waiting Time page.

Figure 3.8: The request for a ride page.

During this period, the system can asks the driver to make a ride, so in figure 3.8 it is shown the screen for this request.
At last, in figure 3.9 there is a mockup for the page where the driver can modify his workshifts. He can add, modify or remove them. The page where the driver can see his workshifts is not displayed, but it is similar to the shown one, without the modification symbols.

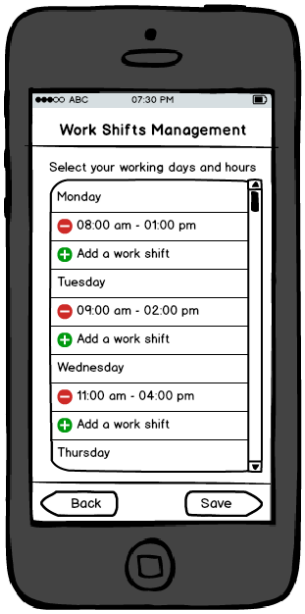

Figure 3.9: The workshifts management page.

3.4 Zerotime Ride

When a user asks for a zerotime ride, he has to insert the starting and the destination positions (the first one can be calculated by the GPS) and then to confirm the request.
In the MA this sequence of operations is performed with three screens (mock-ups in figures 3.10, 3.11 and 3.12), while into the WS it is used only one screen split into the correspondent sections, shown in figure 3.13.

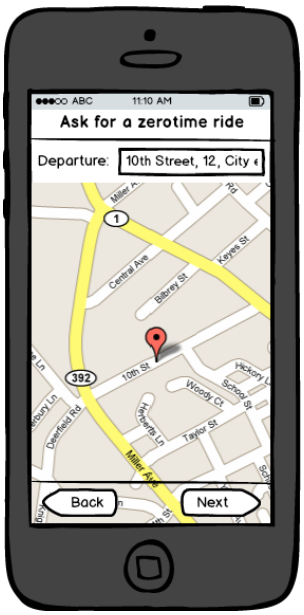

Figure 3.10: Zerotime ride request into mobile application: starting position insert (manually or by GPS) page.

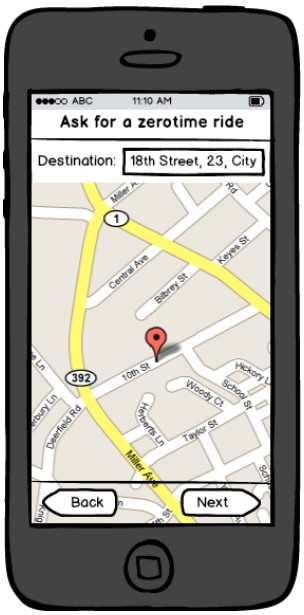

Figure 3.11: Zerotime ride request into mobile application: destination position insert page.

Figure 3.12: Zerotime ride request into mobile application: confirmation page.

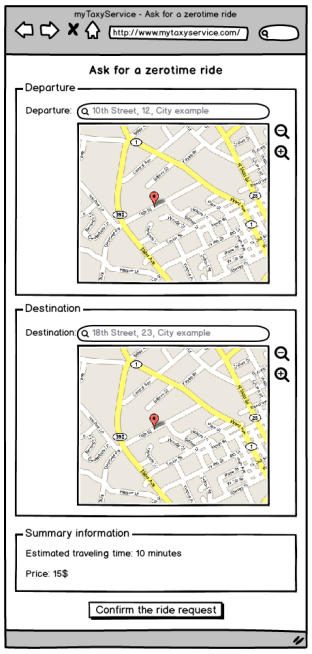

Figure 3.13: Zerotime ride request into website page.

3.5 Future Ride

When a user desires to book a future ride, he has to insert the starting and the destination positions (the first one can be calculated by the GPS), the departure (both the date and the time) and then to confirm the request.
In the MA this sequence of operations is performed with three screens (mockups in figures 3.14, 3.15 and 3.16), while into the WS it is used only one screen split into the correspondent sections, shown in figure 3.17.
In both the MA and the WS, the departure inserting is made with a popup that uses the apposite operative system functionalities to insert the dates (for example a computer OS can use a calendar for dates and a box for time, while a mobile OS can use a dropdown list or something that slides following the finger's touch).

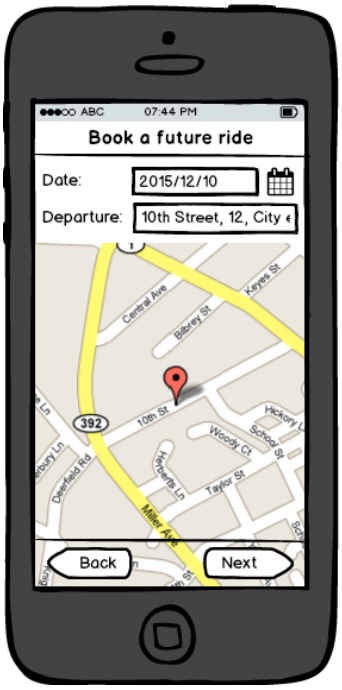

Figure 3.14: Future ride request into mobile application: departure starting position insert (manually or by GPS) page.

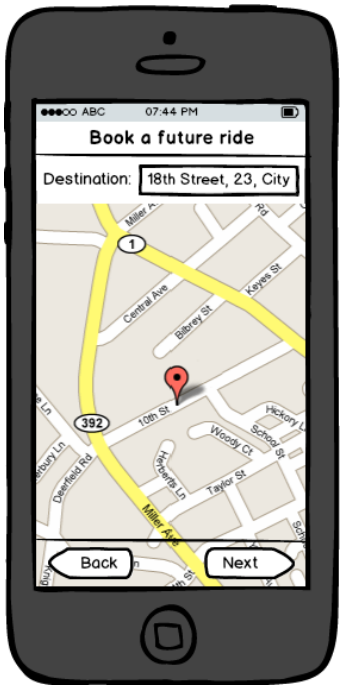

Figure 3.15: Future ride request into mobile application: destination position insert page.

Figure 3.16: Future ride request into mobile application: confirmation page.

Figure 3.17: Future ride request into website page.

3.6 Other User Functionalities

myTaxiService gives other useful functionalities to the users.
In MA there are two screens: in figure 3.18 the users can see all his reservation split into two sections, one for the future reservation (they can be edited) and one, at the bottom, for a list of the past rides; in figure 3.19 the user can edit a specific reservation (either the departure or the positions can be modified) that it was selected in the previous page by a click on the “edit, delete”button.

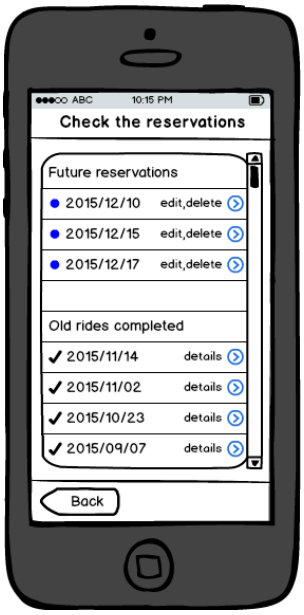

Figure 3.18: The view reservations page into mobile application.

Figure 3.19: The page where a reservation can be modified or be cancelled into mobile application.

In WS the same operations can be performed into the same page, displayed in figure 3.20.

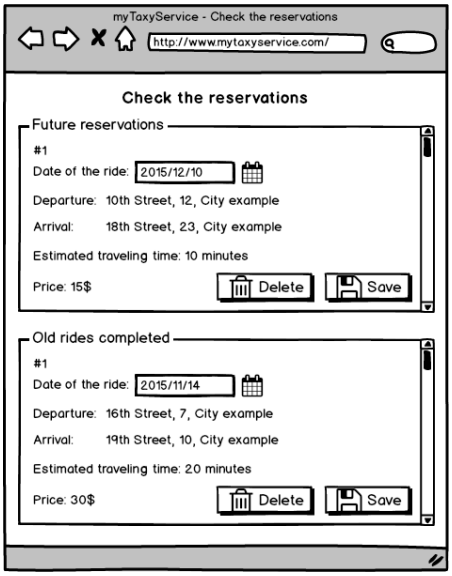

Figure 3.20: The check reservation page into page.

Finally, the user can received some alerts regarding, for instance, the assignment of the ride for which he asked for. The alerts can also be a general information, related to some problems of the service or to a strike.
The corresponding screen for the MA and the WS are shown, respectively, in figures 3.21 and 3.22.

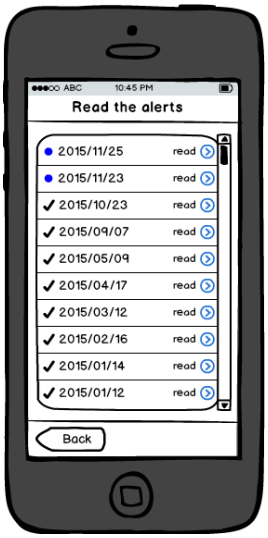

Figure 3.21: The alerts page into mobile application.

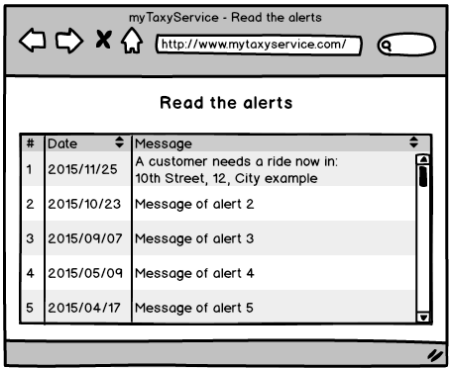

Figure 3.22: The alerts page into website.

Chapter 4

Other Info

This chapter contains information about the used tools and the hours of work by the members of the working group.

4.1 Working hours

Date	Costanzo's hours	Disabato's hours
2015/11/16	1h	1.30h
2015/11/17	2h	2h
2015/11/18	1h	1h
2015/11/20	2.30h	2.30h
2015/11/21	1h	2h
2015/11/22	1.30h	-
2015/11/23	1h	-
2015/11/24	4h	5h
2015/11/25	-	1.30h
Total DD	14h	15.30h
Global	43h	44.30h

4.2 Tools

In this first requirements study phase the following tools were used:

- LaTeX and TexStudio editor
- starUML