

Politecnico di Milano

A.A. 2015-2016

Software Engineering 2: “myTaxiService”

Design Document

Luiza Bentivoglio, Michele Cantarutti

4 December 2015

Summary

**1. INTRODUCTION ............................................................................................................5**

1.1 PURPOSE ............................................................................................................................................5

1.2 SCOPE .................................................................................................................................................5

1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS ...................................................................................6

1.4 REFERENCE DOCUMENTS .................................................................................................................6

1.5 DOCUMENT STRUCTURE ………………………………………………………………………………………………………...6

**2. ARCHITECTURAL DESIGN ..........................................................................................8**

2.1 OVERVIEW ......................................................................................................................................11

2.2 HIGH LEVEL COMPONENTS AND THEIR INTERACTION .............................................................12

2.3 COMPONENT VIEW ........................................................................................................................11

2.4 DEPLOYMENT VIEW ......................................................................................................................12

2.5 RUNTIME VIEW ..............................................................................................................................11

*2.5.1 Runtime units .............................................................................................................................33*

*2.5.2 Request a taxi .............................................................................................................................33*

*2.5.3 Reserve a taxi ..............................................................................................................................34*

*2.5.4 Cancel a reservation ................................................................................................................35*

*2.5.5 Modify a reservation ................................................................................................................36*

*2.5.6 Inform about the availability ...............................................................................................37*

2.6 COMPONENT INTERFACES ............................................................................................................12

2.7 SELECTED ARCHITECTURAL STYLES AND PATTERNS ...............................................................11

2.8 OTHER DESIGN DECISIONS ...........................................................................................................12

**3. ALGORITHM DESIGN..................................................................................................18**

**4. USER INTERFACE DESIGN........................................................................................22**

**5. REQUIREMENTS TRACEABILITY...........................................................................22**

**6. REFERENCES …………………........................................................................................22**

**1. Introduction**

* 1. **Purpose**

The purpose of this document is to give a functional description of the system.

In order to do so, we’ll analyze the architectural design, by showing which components will make up our system and how they will interact with one another. We will also show the algorithm design, focusing on the most relevant algorithmic part of the project, which is the way the system manages the requests and how it assigns them to taxi drivers. Lastly, this document will provide an overview on how the user interfaces will look like.

**1.2 Scope**

The scope of this document is to show our choices with regard to architecture of our system. Most of the choices in style and design will be discussed at the end of chapter 2.

**1.3 Definitions, acronyms, abbreviations**

**1.4 Reference documents**

Documenti della prof + roba su google + nostro rasd

**1.5 Document structure**

**2. Architectural design**

**2.1 Overview**

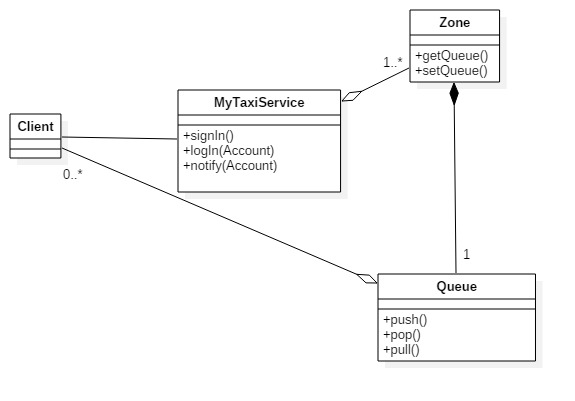
In the next sections we will show the architectural design of our project. At first we will only introduce the high level components of our architecture and describe the main interaction between them, without getting into details.

After this introduction, we will refine what we’ve show and identify sub-components that make up our components. Along with this, we will show what kind of interaction there is between components, by showing who offers a certain service and who uses it. We will also show what executable will be running and on which device they’ll be running. Lastly, we’ll talk about the architectural styles and patterns we decided to use in our project.

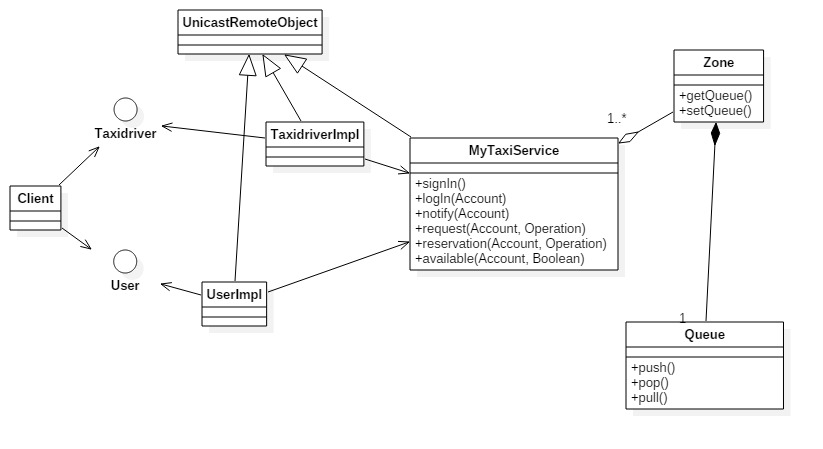
**2.2 High level components and their interaction**

Here we will introduce the high level components of our architecture. This diagram will provide an overview of the entire system, identifying the main components that will be developed for the product. We tried to use possibly nontechnical terms so that it could be understandable to the administrators of the system, too.

At the core of our system we decided to use a component called Controller which has the most important role, because it has to manage services that are asked by the clients. Services could be simple features like signing up and logging in or making a taxi request (if the client is a passenger) and notifying the system about the availability of a taxi driver (if the client is a taxi driver). We decided to use a component for taxi requests. It should be noted that there are two different functions associated with requests: MakeARequest can be asked by a client and the request will be managed by the Controller, but the actual creation of a request is only made afterwards through CreateRequest, because, for example, it wouldn’t make sense to create an entity for a request without verifying if there is an available taxi driver who will take care of it, because the request will simply be dismissed at once. It is also necessary that the Controller is able to communicate with the Queue component because it must gather information about which taxis are available when a certain request is received.

****

**2.3 Component view**

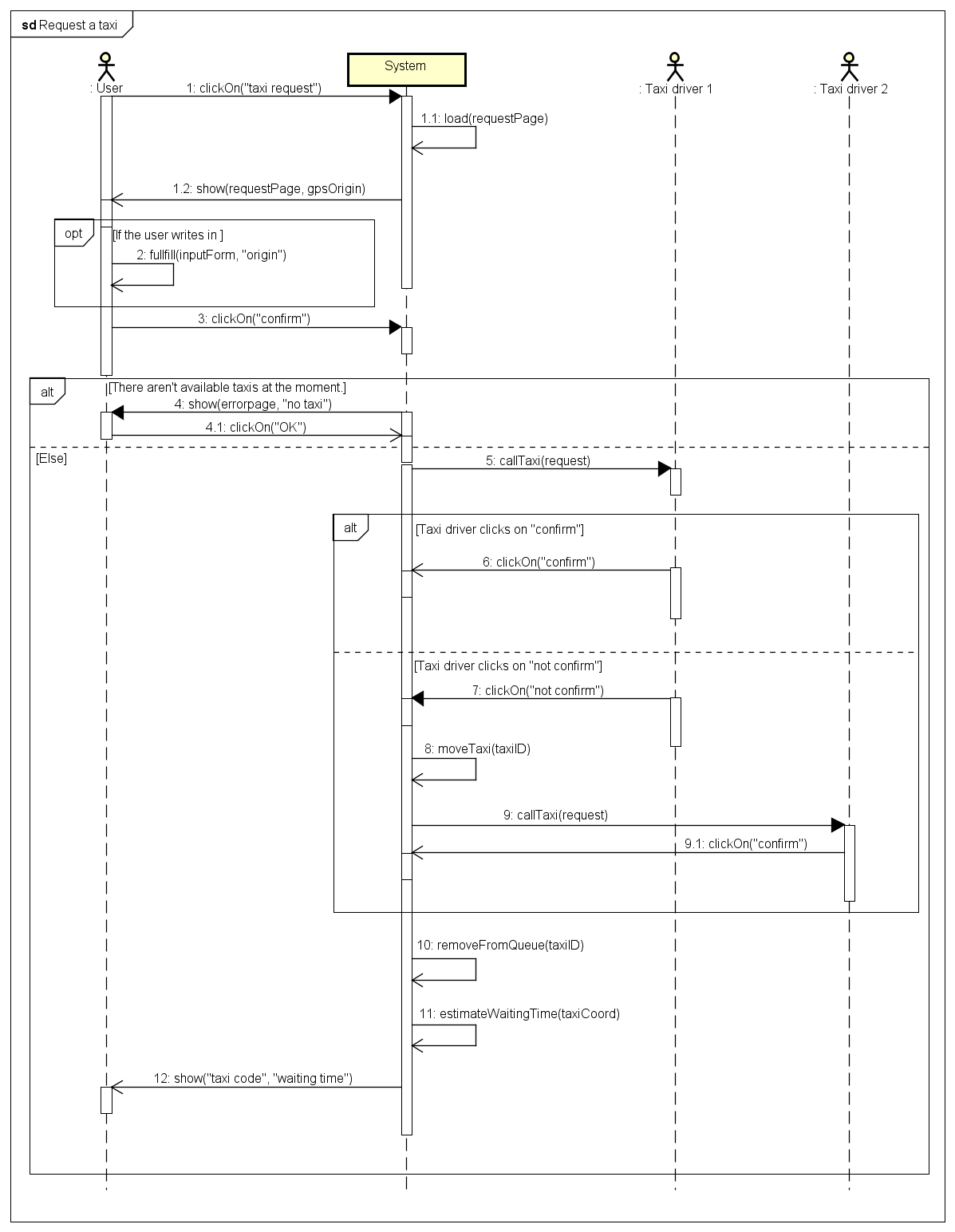
****

**2.4 Deployment view**

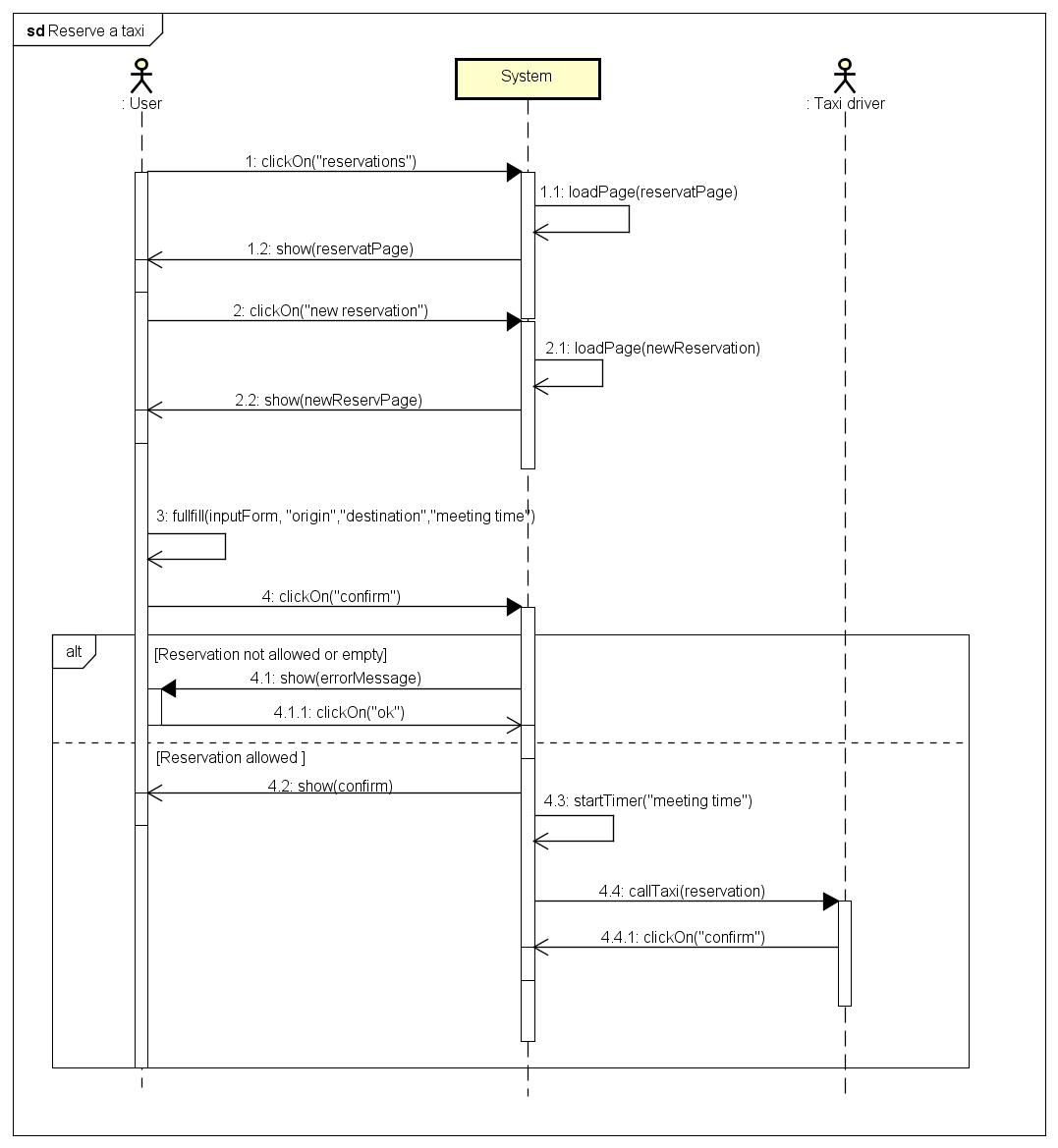
**2.5 Runtime view**

**2.5.1 Runtime units …qui cosa va messo?**

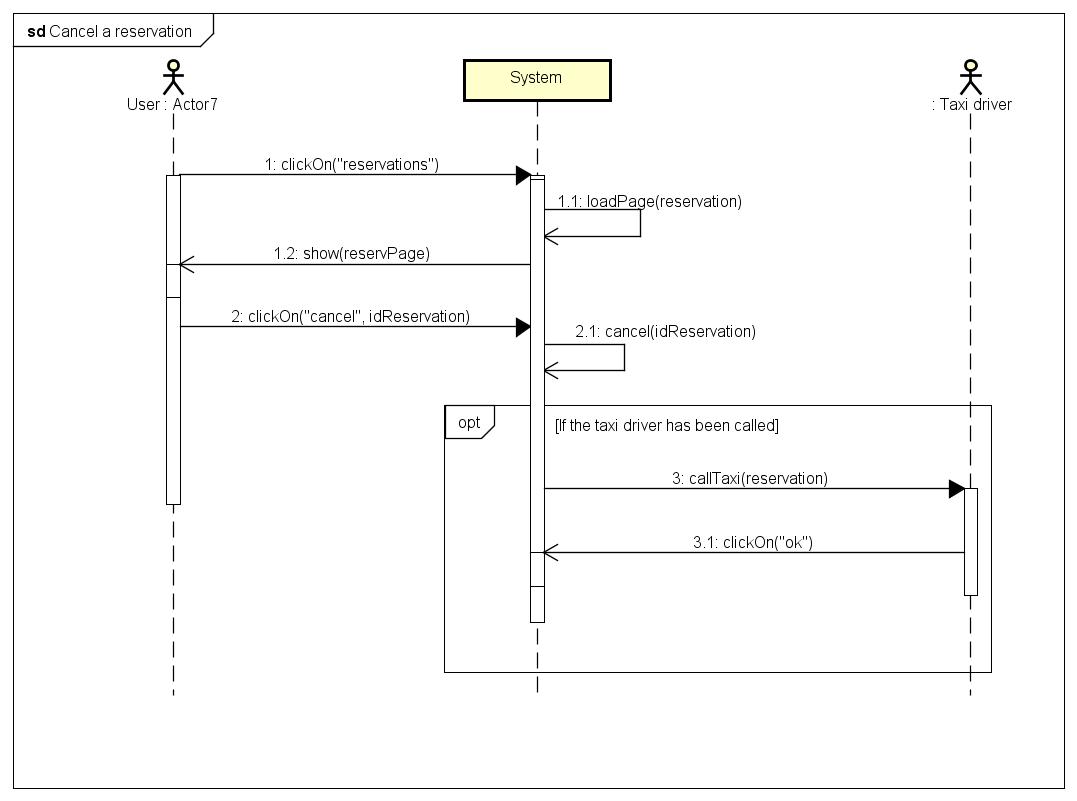
**2.5.2 Request a taxi**

****

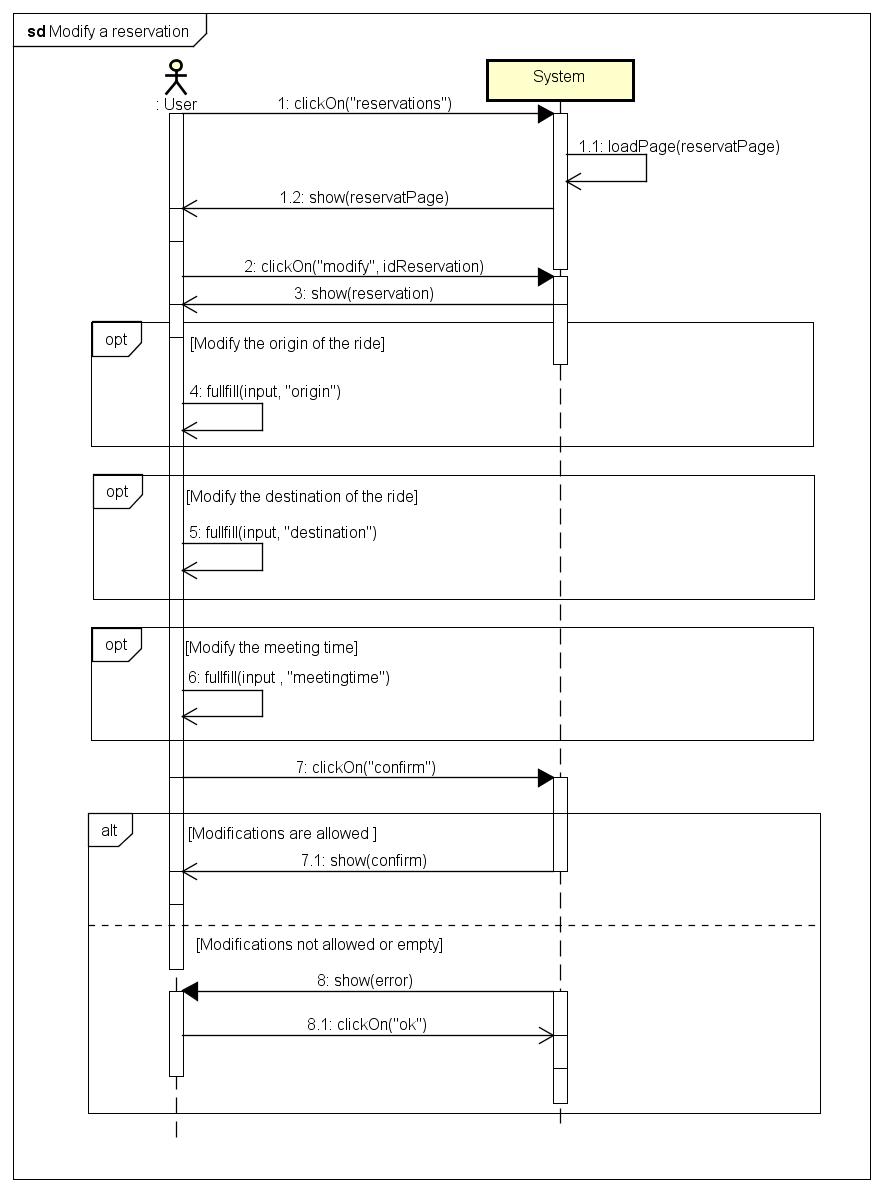
**2.5.3 Reserve a taxi**

****

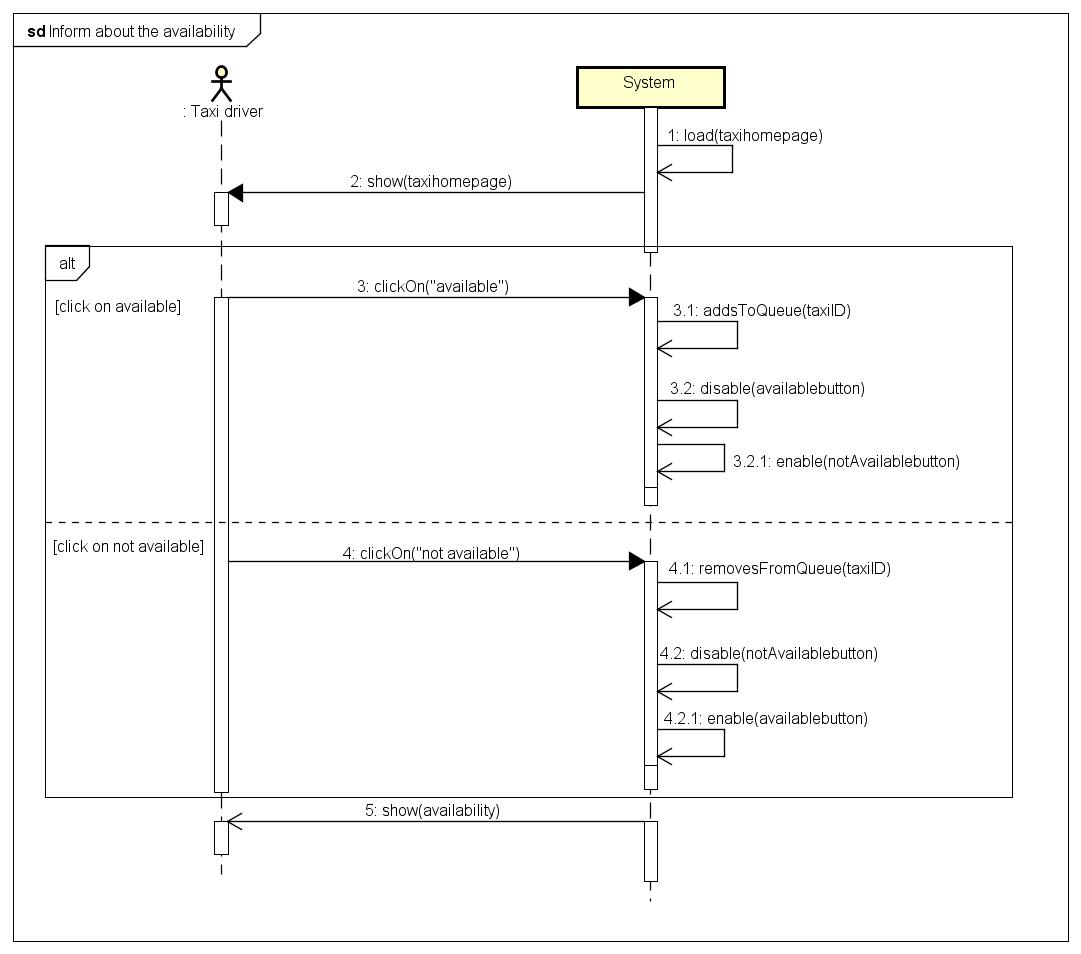
**2.5.4 Cancel a reservation**

****

**2.5.5 Modify a reservation**

****

**2.5.6 Inform about the availability**

****

**2.6 Component interfaces**

**2.7 Selected architectural styles and patterns**

**2.8 Other design decisions**

**3. Algorithm design**

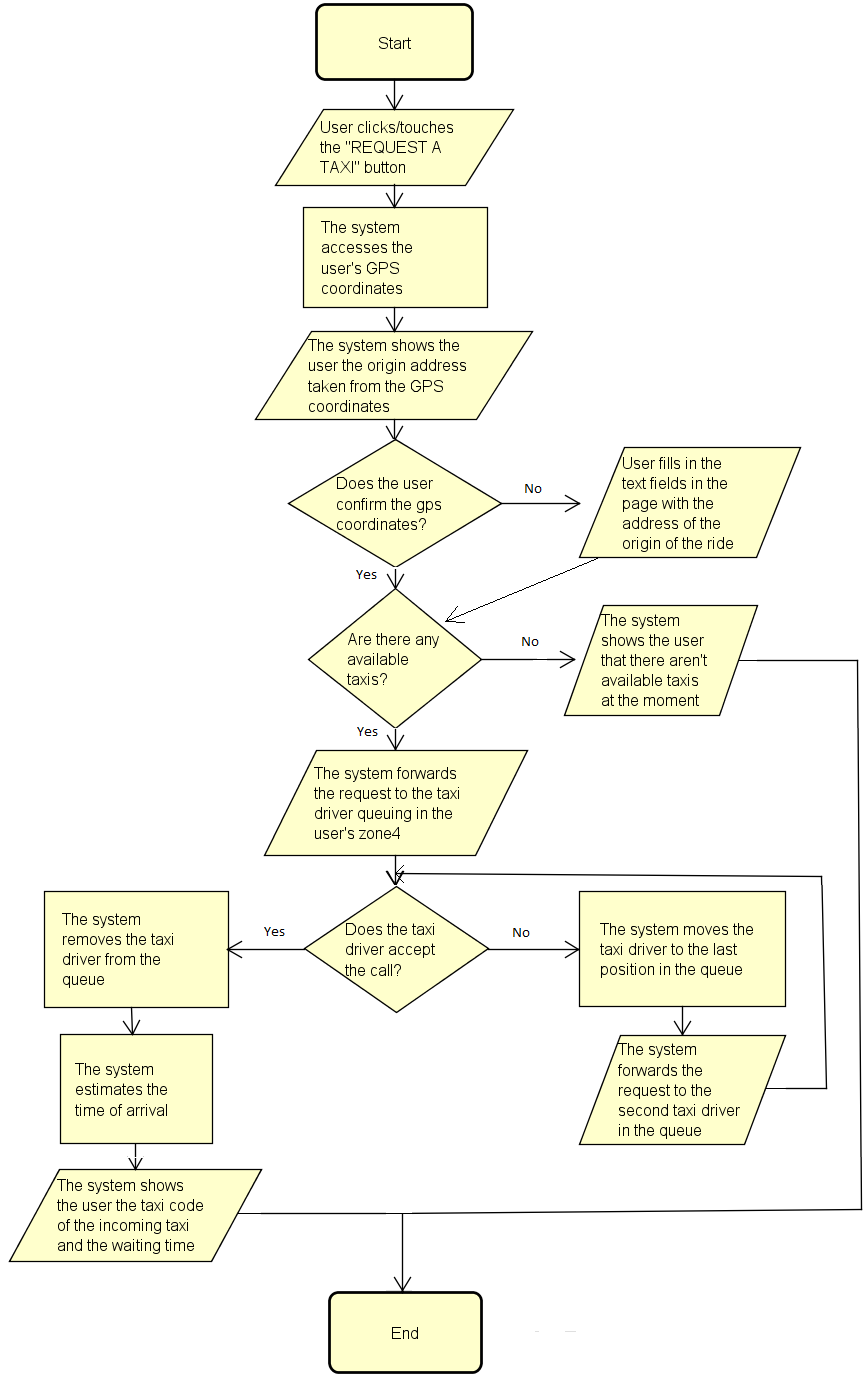
(Nel flow chart aggiungerei le parole YES/NO e collegherei il box “User fills in the text fields…etc” con un altro box)

Having to decide which algorithmic part of our project to describe, we decided to focus on the part that manages requests, because we feel like it’s the very core of our project.

As it’s shown in the flow chart, the whole process begins when a passenger requests a taxi by either clicking on the “request a taxi” button, or touching it if he/she is using the mobile version of the application. In order to make things faster for the passenger, the system will automatically gather information about the current location of the passenger through the GPS, so that he/she won’t have to type it down, unless the displayed address is wrong, in which case the passenger will have to write it by himself/herself.

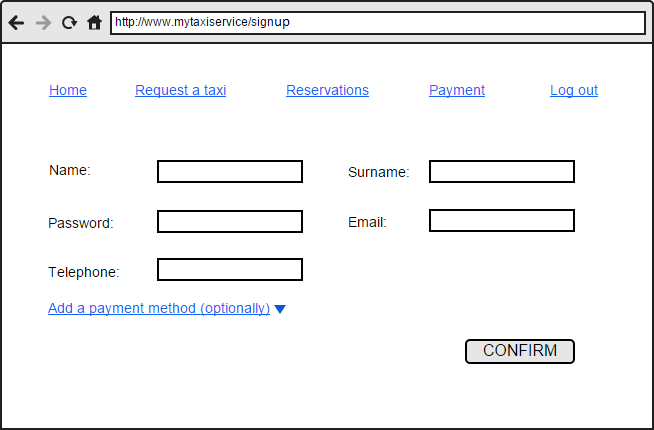
The system will check whether there are any available taxis in the zone where the passenger currently is and if it can’t find at least one, it will notify the passenger that no one will be able to come pick him/her up. Otherwise, the system will forward the request to the first available taxi driver. If the taxi driver doesn’t accept the call, he/she will be moved to the last position in the queue and the request is forwarded to the second taxi driver.

When a taxi driver accepts the call, he/she is removed from the queue. Then, the time of arrival is estimated and shown to the passenger, along with the code of the incoming taxi.



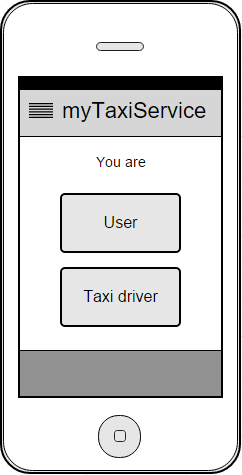
**4. User interface design**

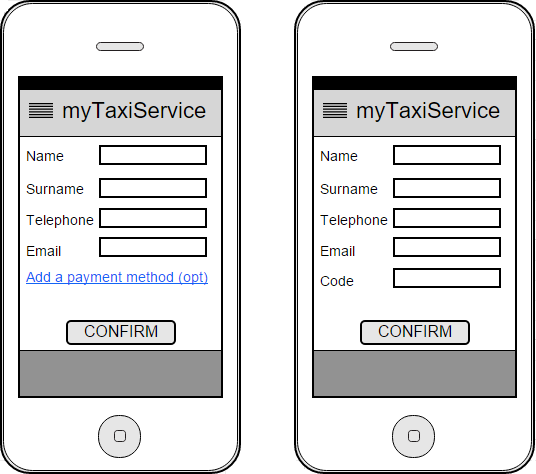
This is the page the user sees when he/she goes on the myTaxiService page to sign up for the service. This page is shown on the web application, which means the user is accessing it through a computer and therefore we can assume the user is a passenger, because taxi drivers can only access it through a mobile. Because of this, the user is only asked for basic data like name, surname, password, email and telephone number but a registration code is not required, unlike for taxi drivers.

****

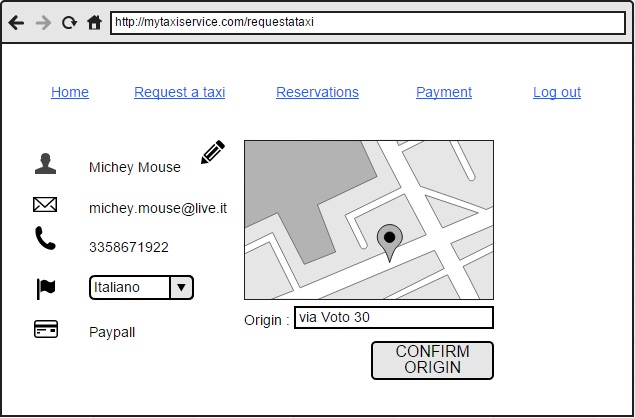
As we can see in the pictures, the mobile version shows a couple of minor differences. First of all, at the very beginning, the user is asked whether he/she is a passenger or a taxi driver. Depending on what they have selected, the following page will be slightly different. If the user selects “Passenger” then the following page will ask for basic data, plus, optionally, what additional payment method they would like to use. Otherwise, if the user selects Taxi Driver, they will be asked for basic data along with the registration code, which has been given to the taxi driver by the administrator of the system (which is the government of the city in the case of this project).

(nel disegno User va cambiato in Passenger)

****

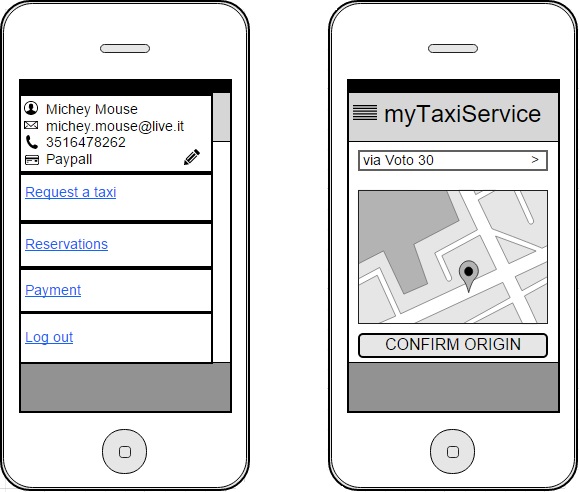
****

This is the page that is shown to the user when he/she is making a request. To make things faster and easier for the passenger, the system will get the current location of the passenger through geolocation, but he/she is able to modify if he/she wants to. By clicking “confirm origin” the passenger will send the request and then it will be managed by the system.

****

In this case, the pictures represent the same situation as before, but they’re on the mobile application instead. The picture on the left shows the basic menu that the passengers can see from their mobile application, which allows them to make requests, reservations, manage payments or log out.

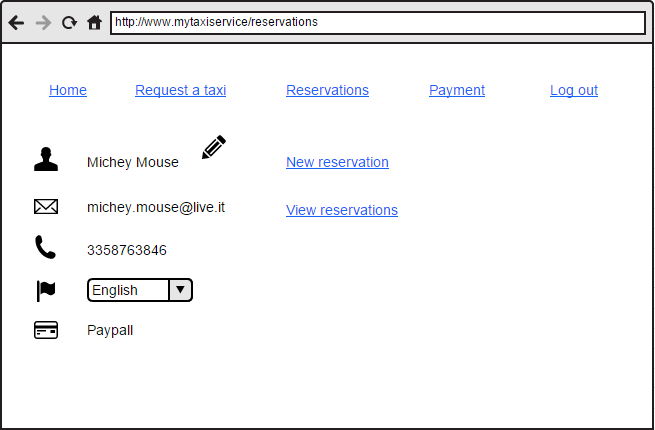
As show in the picture on the right, just like before, instead of asking the origin of the ride, the system will automatically suggest the current location of the passenger as the origin of the ride (the location is known to the system thanks to the GPS). Either way, if the address obtained through the GPS is wrong or if the passenger would like to be picked up somewhere else, he/she can simply type down an address by himself/herself.

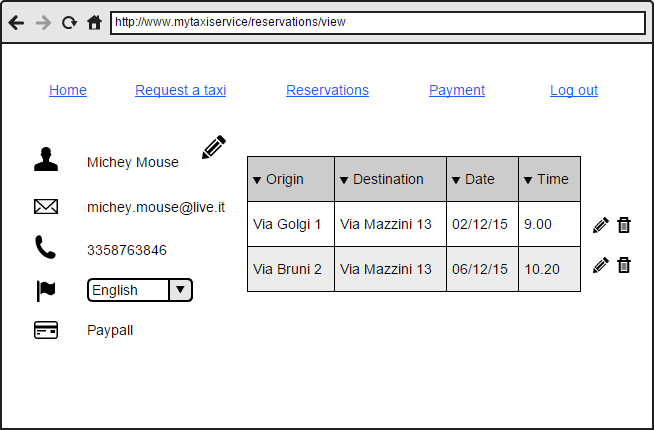
****

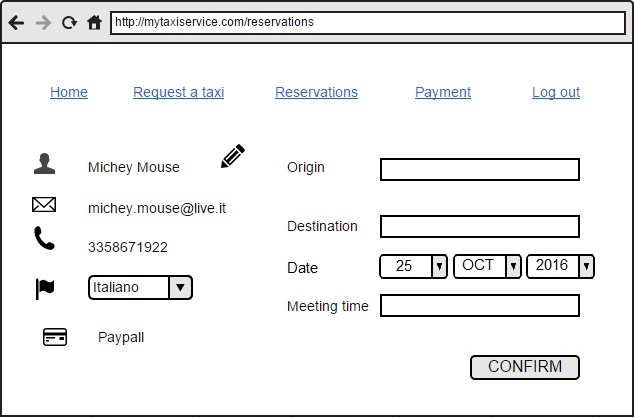
The following pictures show what happens when passengers decide to manage their reservations when they are on their computer. First of all, they can either choose if they want to make a new reservation or choose if they want to view their past and future reservations.

When they click “View reservations” they’re shown all their reservations, both the past and the future ones. Each reservation is shown with its corresponding details, such as the date, time, origine and destination of the ride. Next to each reservation are two buttons which allow passengers to delete or modify a certain reservation.

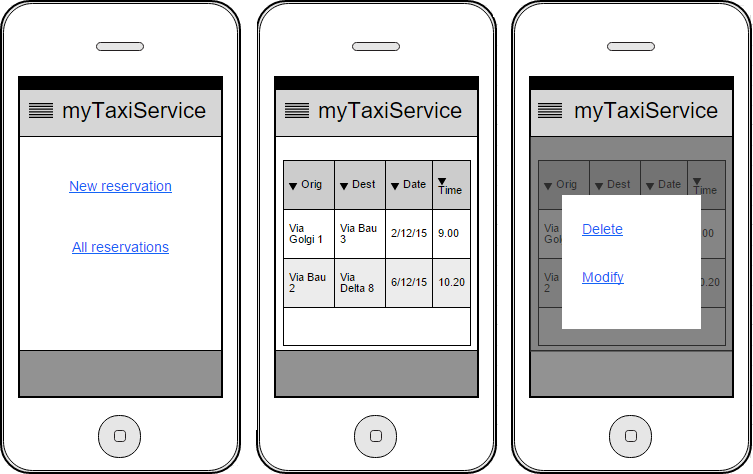
Lastly, the third picture shows what passengers see when they click “New reservation”. The interface is very easy and it simply asks passengers to fill in the fields with all necessary data, such as the date, meeting time, origin and destination of the ride.

****

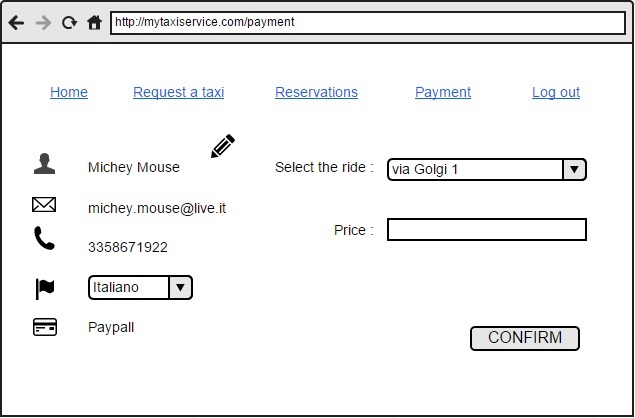
****

****

The next four pictures show the same situations as before, but they’re on the mobile application instead. As we can see, there are no major differences and passengers are allowed to choose whether they want to make a new reservation or view the ones they’ve already made, along with the possibility of deleting and modifying them.

****

****

****

****

In the left picture we can see what the taxi drivers can see when they want to change their status. The interface is very simple: when they’re available, their only option is to change their status to unavailable if and when they want to, and when they’re unavailable, their only option in to change their status to available.

The picture in the middle shows what the taxi drivers can see when they receive a call. Some details are shown, like the name of the passenger, the phone number and the address so that they know where they have to go to pick up their customer. At the bottom of the page they’re shown an option, to decide whether they want to accept the call or not.

The right picture shows what the taxi drivers sees when a reservation has been cancelled by a passenger. This is basically a message to notify the taxi driver that he/she doesn’t have to take care of the call because the passenger has cancelled it.

****

**5. Requirements traceability**

**6. References**