My Taxi Service

DESIGN DOCUMENT

Julián David Gallego García

Politecnico di Milano December 4, 2015


POLITECNICO
MILANO

CONTENTS

1. Introduction

1.1. Purpose

This document is the Design Document for the system My Taxi Service project. This document has the purpose to present the design phase of the system, highlighting the decisions to reach an appropriate architecture for the system, taking into account the principles of software design and requirements for the system and justifications. A deeper explanation of the user interface design than in the RASD and it is presented the key algorithms of this system.

1.1. Scope

The goal of this project is to optimize the taxi service in a large city through a web application and a mobile app. These applications allow the passengers to request a taxi in an easy way and the taxi drivers to have a fair management of taxi queues. This document has the goal to present the system with more technical details, an will be focused on the architecture of the system, the interaction of the main components of the system, the interfaces of the system and the main algorithms of the program.

1.2. Definitions, acronyms, abbreviations Reference documents

1.2.1. Definitions

Application: A computer program designed for a specific task or use.

Architecture: <system> fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

Queue: In programing:
a. A sequence of stored data or programs awaiting processing.
b. A data structure from which the first item that can be retrieved is the one stored earliest.

Request: The act or an instance of asking for something.

Requirement: something that is needed or that must be done. Something that is necessary for something else to happen or be done.

System: A group of interacting, interrelated, or interdependent elements forming a complex whole, especially:

a. A network of structures and channels, as for communication, travel, or distribution.

b. A network of related computer software, hardware, and data transmission devices.

User: a person or thing that uses something.

Tier: a level or grade within the hierarchy of an organization or system.

1.2.2. Acronyms

GPS: Global Positioning System.
UML: Unified Modelling Language.
IEEE: Institute of Electrical and Electronics Engineers.
RASD: Requirements analysis and specifications Document.
EIS: Enterprise information systems.
JEE: Java enterprise edition.

1.2.3. Abbreviations

App: Application.

1.2.4. Reference documents

IEEE Std 1016-2009, IEEE Standard for Information Technology-Systems Design-Software Design Descriptions.

Specification document: Software Engineering 2 Project, Structure of the design document.

RASD: Julian Gallego, Requirements analysis and specification document for My taxi service.

IEEE Std 42010, IEEE standard on architectural descriptions.

1.3. Overview

The document is organized in:

- Section 1: brief description of the idea, goals and objective of the document.
- Section 2: Describes the architectural choices. From the overview of the hierarchy of that architecture, then the interactions between its components are explain and finally for each defined level or tier all its characteristics are presented.
- Section 3: The key algorithms of this system are explained and the main characteristics are point out. In particular, these algorithms are the ones which manage the taxi queues, the city zones.
- Section 4: Refers to the inner workings of all the graphical interfaces and the general experience of the system.

Section 5: Explain how the requirements defined in the RASD map into the design elements that are defined in this document.

2. Architectural Design

2.1. Overview

In this section it is provided detailed information of the product structure and the implementation of the general structure. In the following sections it is provided details of why it was decided the architecture for the system, what are the characteristics of the architecture that make it the chosen one

Then it continues talking about the general components of the architecture and the subcomponents of each component and their interactions. Following by information of all the functions that do each subcomponent.

After the primary information on how it is constituted the architecture, its explained how all these components work to make the system to work effetely.

The chapter finish with an explanation about the interface and how some components interact with it.

2.2. Architectural Style and Patterns

The system will be developed by using a general client-server 3-tier Architecture. This is because this is a service that its expect to have a lot of simultaneous request from visitors, passenger and Taxi drivers. Now it is present a general description of the architecture proposed for the system and furthermore, also a small description for each of the components.

The design of the architecture is based on a client-server with a fat client and fat server, this decision was made because with the actual capacities of the cellphones and computers, we want to take advantage of it and make part of the calculus and processing on the device of the costumer to reduce the time of respond of the application, improve the performance and to be able to respond all the simultaneous request in the minor time possible. For the system was decided a 3-tier distributed architecture to distribute the operations on visualization, logic and database, each tier is described as follows:

Client tier: This is an architectural layer whose job is translating the user actions and presenting the output of tasks and results into something the user can understand, in this case, in My taxi service is a web and mobile application.

Business Logic tier: This is an architectural layer whose job is coordinate the application, processes commands, perform calculations, make logical decisions and evaluations. It also moves and processes data between the client and EIS tiers.

EIS tier: This is an architectural layer whose job is to provide an abstract interface to information storage mechanism, is in charge of storing and retrieving information from the database.

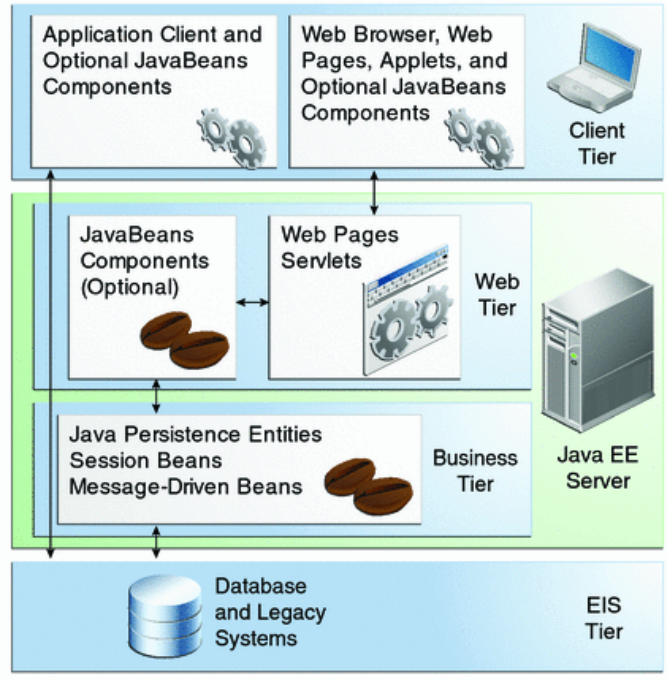

Figure 1. System architecture [1].

2.3. High level components and their interaction

As described before, the system has 3 high level components that are the tiers of the architecture:

-The client.
-Business logic.
-EIS.

These components manage all the operations made on the system asynchronously, this are started by a request (signal) from the tier that wants to start an operation to the the tiers that needs to accomplish the operation.

The first component, the client, has the possibility to start communication directly with the two other tiers, with logic when is trying to pull a request for a taxi/Search a costumer or when the driver wants to change his status, and with the EIS, when the client wants to manages his account information.

The second component, logic, manages all the queues of the city, checks the position of every taxi using the GPS of each driver mobile. When the client component sends a signal of status change from a driver, it also is in charge to add or remove the taxi to the queue and send a signal to the EIS to save the changes. Keeps the drivers in the correct queue depending of the position. When the client component sends signals of request a taxi, assigns a taxi to each passenger and sends them the current location of the taxi that accepted the ride when they are waiting the arrival, this is possible by sending a signal to the Client component to displayed on the user interface.

The EIS tier is the component that has access and manages all the information about the passengers and drivers, including also the queues information, this tier is entirely reactive from the signals from the logic or client components.

2.4. Component view

From the high level components or tiers, now its explained the subcomponents that are equally important, they are those who carry out the various operations necessary to all the functionalities of the system works. The subcomponents and their functions are:

2.4.1. Client tier subcomponents

Status/information manager: This subcomponent is in charge to generate a signal when the driver wants to change of status, when a visitor wants to create an account or when a user wants to see or change his personal information. It also is in charge to verify the information when the user fills a form, being in the correct format, to send always a standard message to the other components.

User interface: This subcomponent has the responsibility of show messages, web pages or interfaces depending of the signal that receive from the other subcomponents.

Request manager: When a driver searches a passenger or a passenger request a taxi, this component is the one that create a signal to fulfill this actions.

2.4.2. Logic tier subcomponents

Queues manager: This subcomponent is the one to has the information about the queues like a cache of the most recent information, receive information to update drivers from the position manager, send information to the dispatcher to propose to the first driver

of each zone a possible ride, finally is in charge of update most of the data related to the queues sending the data to the database.

Position manager: This subcomponent is the one that controls the positions of the drivers, receive the information from the GPS and send the information to the notification to give information to the passenger, also with this position, controls the zone in witch are the taxi drivers, changing them from the queue that are if they go away from the area of the zone that where assigned before.

Status manager: Is the responsible to update the status of the drivers, when the status/information manager send that signal, it has to send a signal to the queue manager to fulfill the change, then get the information relevant and update it to the database.

Dispatcher: This subcomponent has the responsibility to propose a ride to the drivers on the top of each zone with the information received from the queue manager, if a taxi driver accept a ride, this component computes then waiting time and generate a code to the ride, this information is send to the notification component, it also send a signal to the queues manager to update the queues. If a taxi denies a ride, this component sends the information to the queue manager to put the driver at the end of the queue.

Notification: Is the one in charge to compute the waiting time and update the time and the position to the driver and the passenger, when the driver accepts a ride.

2.4.3. EIS tier subcomponents

Database is the only subcomponent of this tier of the system, receive all the signals from the other components, update the data recorded or send data asked from a component.

2.5. Deployment View

The following diagram is the deployment view of the system, it shows the relations that are between the components and subcomponents, giving the idea of the subcomponents that are need it to do a task on the system, dependencies and the route through the subcomponents that follows actions in the system.

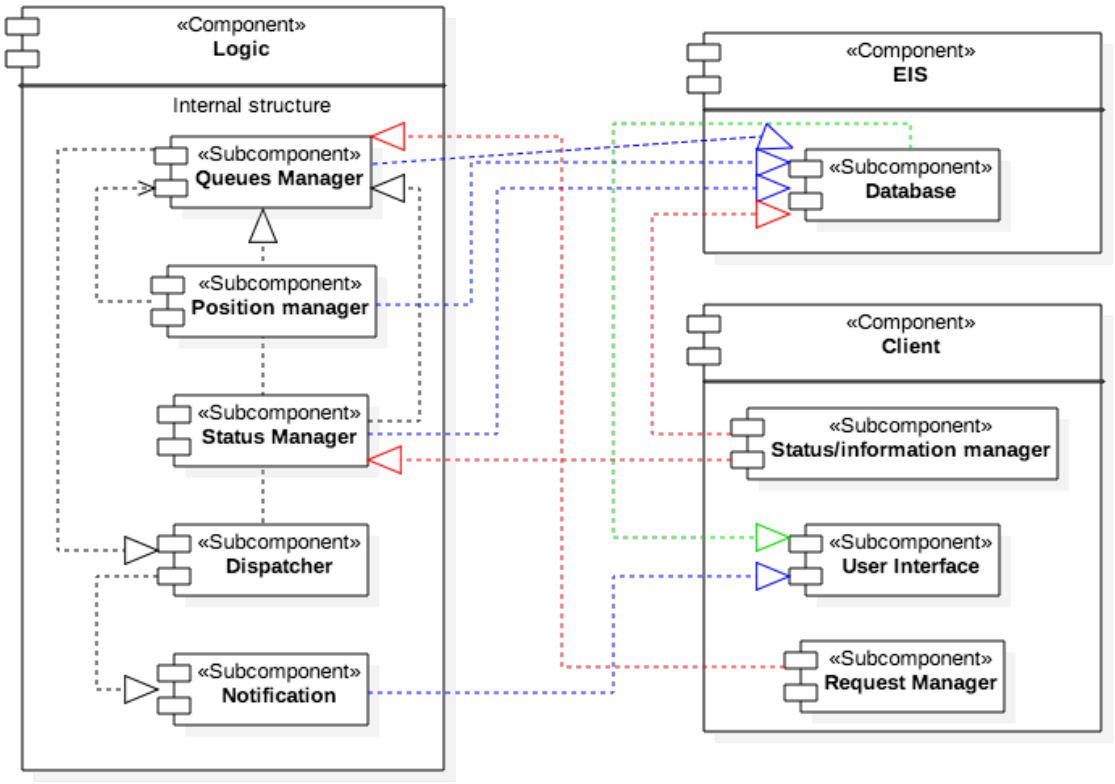

Figure 2. deployment View.

2.6. Runtime view

In the following figure is shown the possible actions that can be done in the system. With the actions or functionalities that can be done in the system shown in the image and based on the deployment view, its possible to understand more about the architecture of the system with the help of the sequence diagrams that explain how functionalities are accomplish by the combination of work of the subcomponents. In the following numerals are explained by sequence diagrams each one of these functionalities.

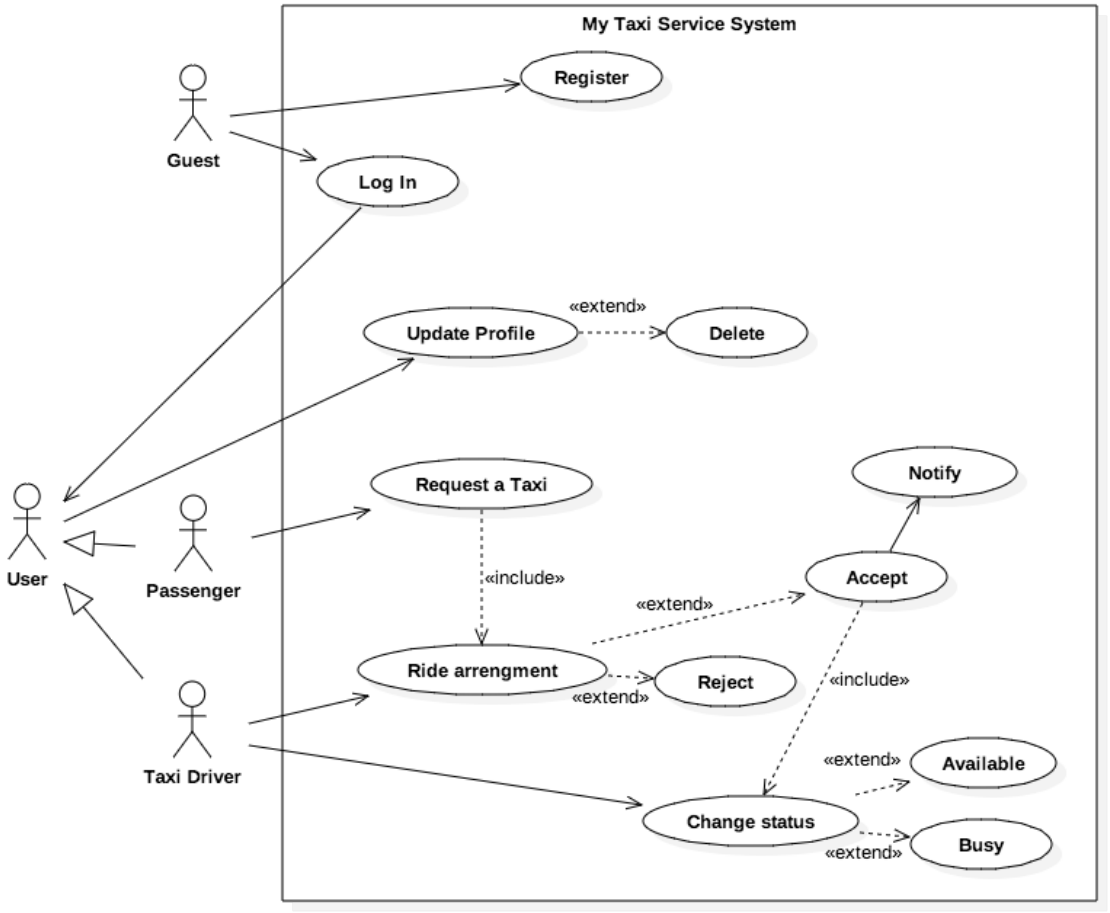

Figure 3. User interactions with the system.

2.6.1. Users registration

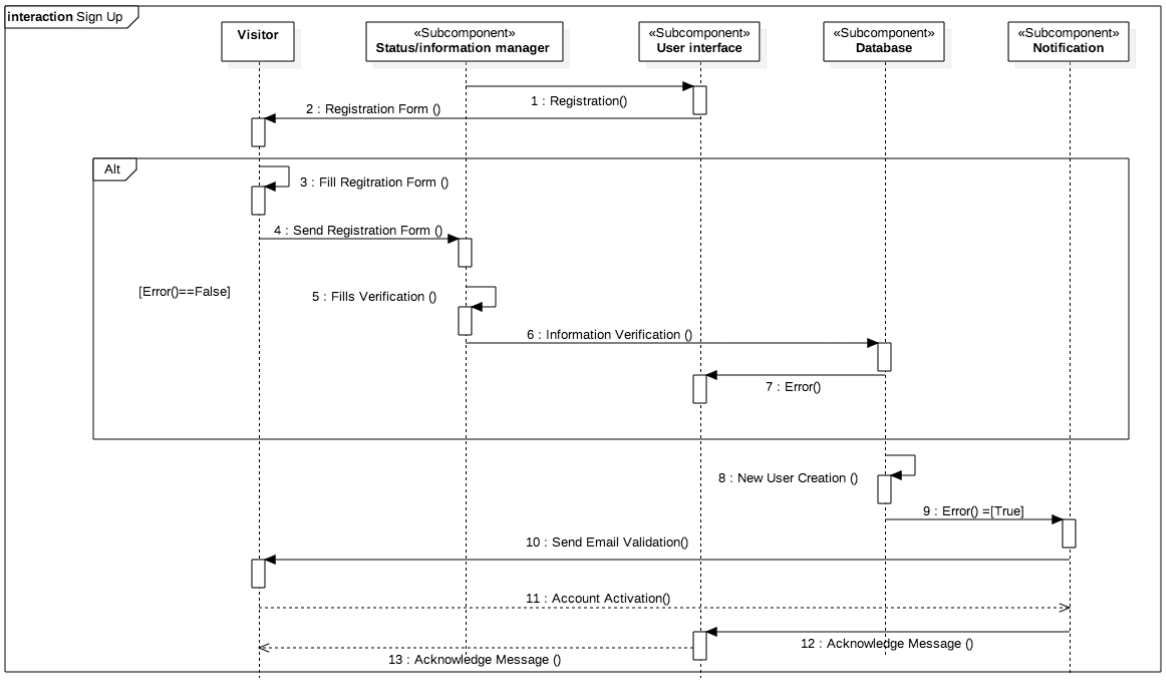

Figure 4. Sing up sequence diagram.

2.6.2. Log In

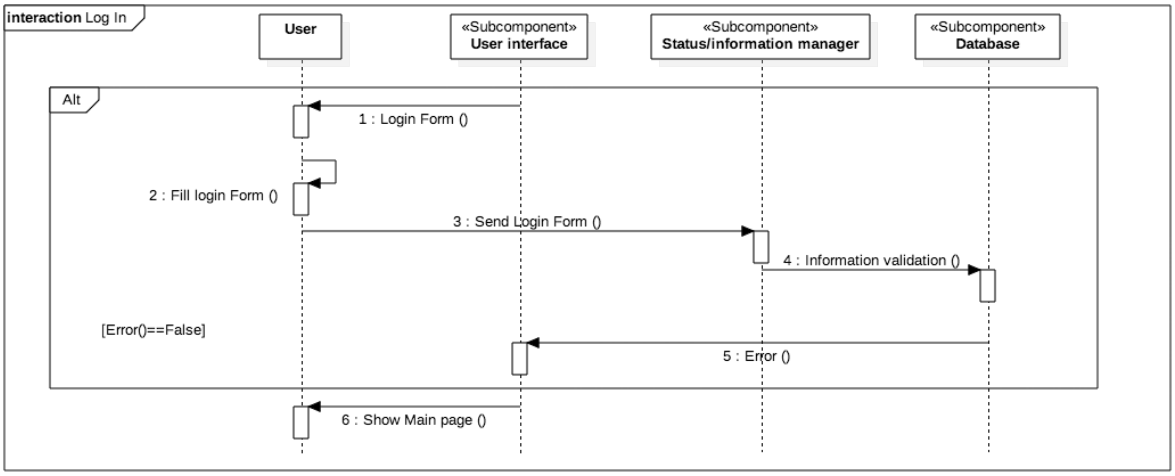

Figure 5. Log in sequence diagram.

2.6.3. Manage personal information

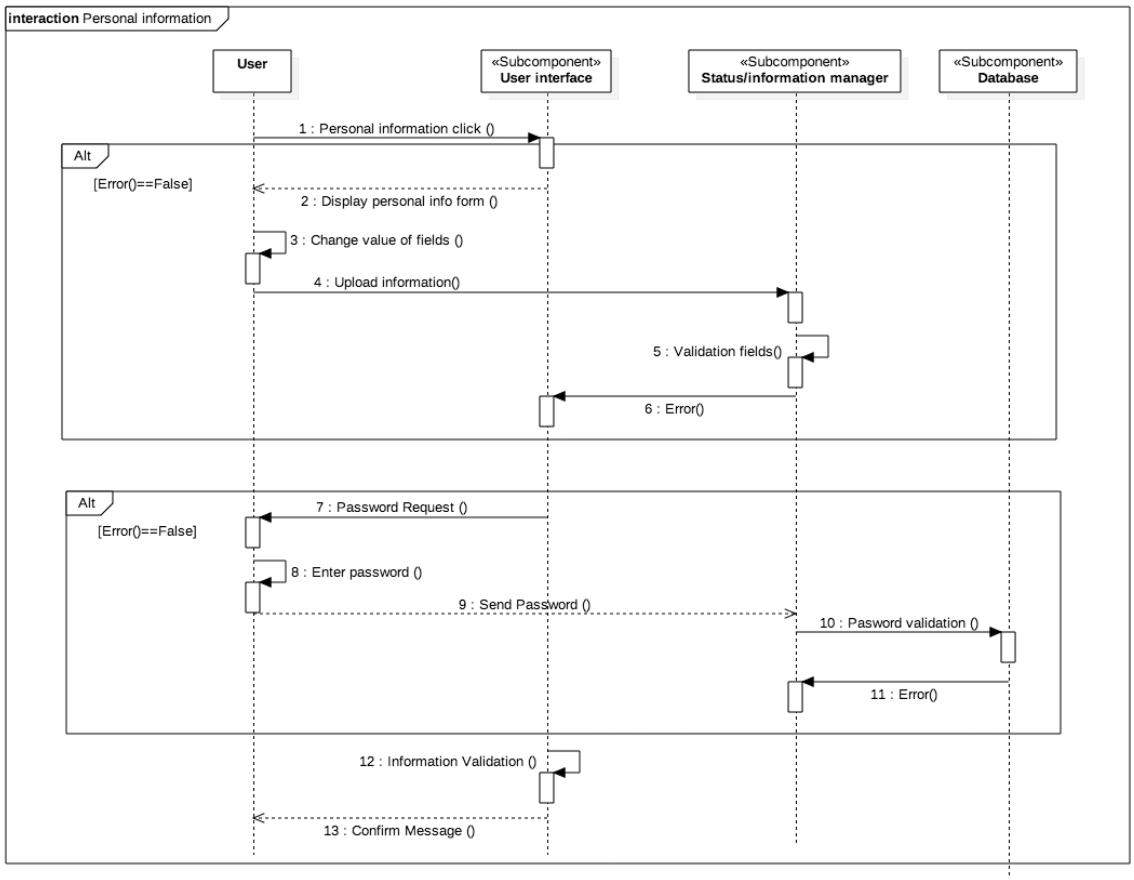

Figure 6. Manage personal information sequence diagram.

2.6.4. Change driver status

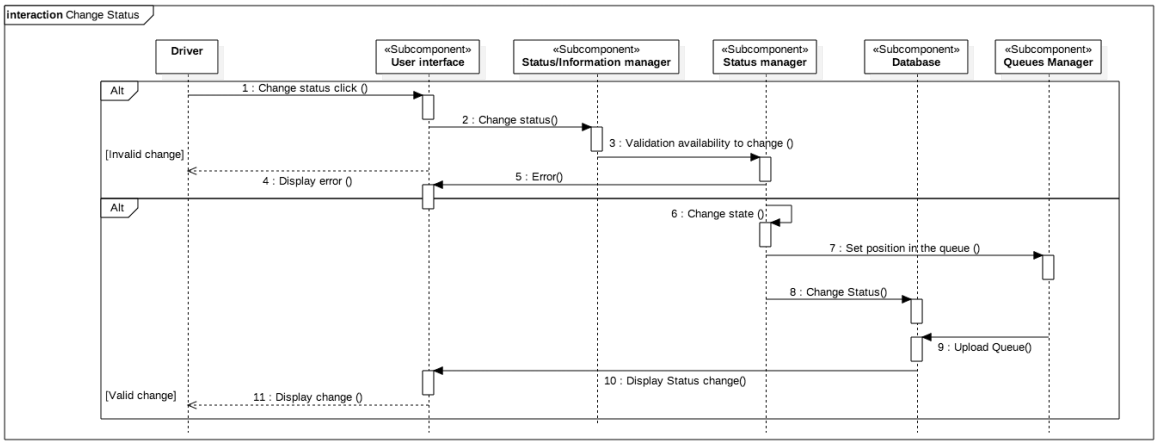

Figure 7. Change status sequence diagram.

2.6.5. Request a Taxi

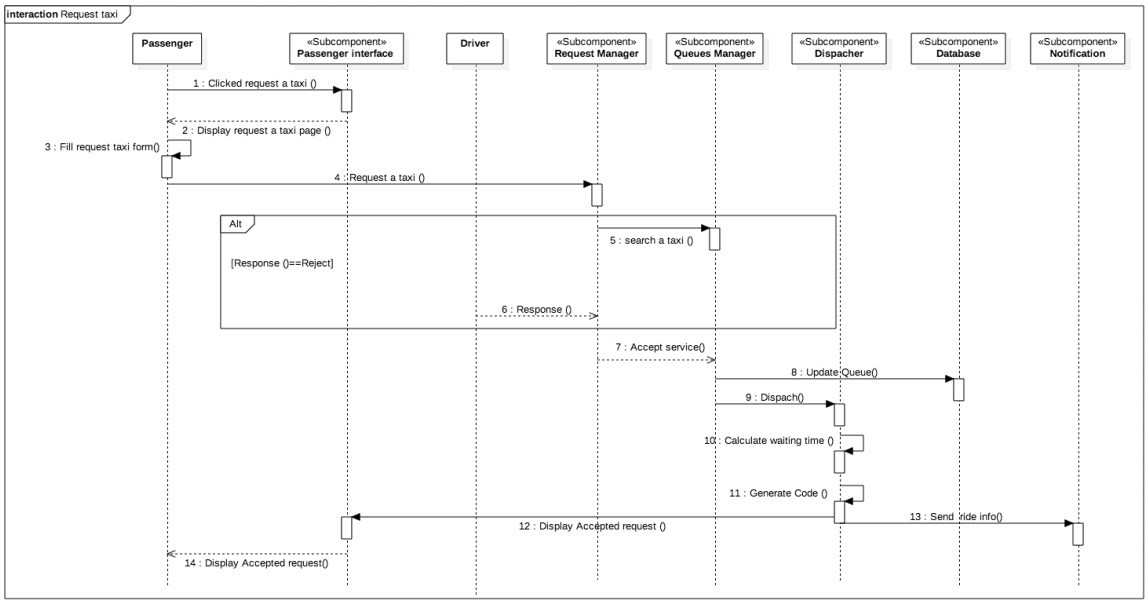

Figure 8. Request a taxi sequence diagram.

2.6.6. Searching passenger

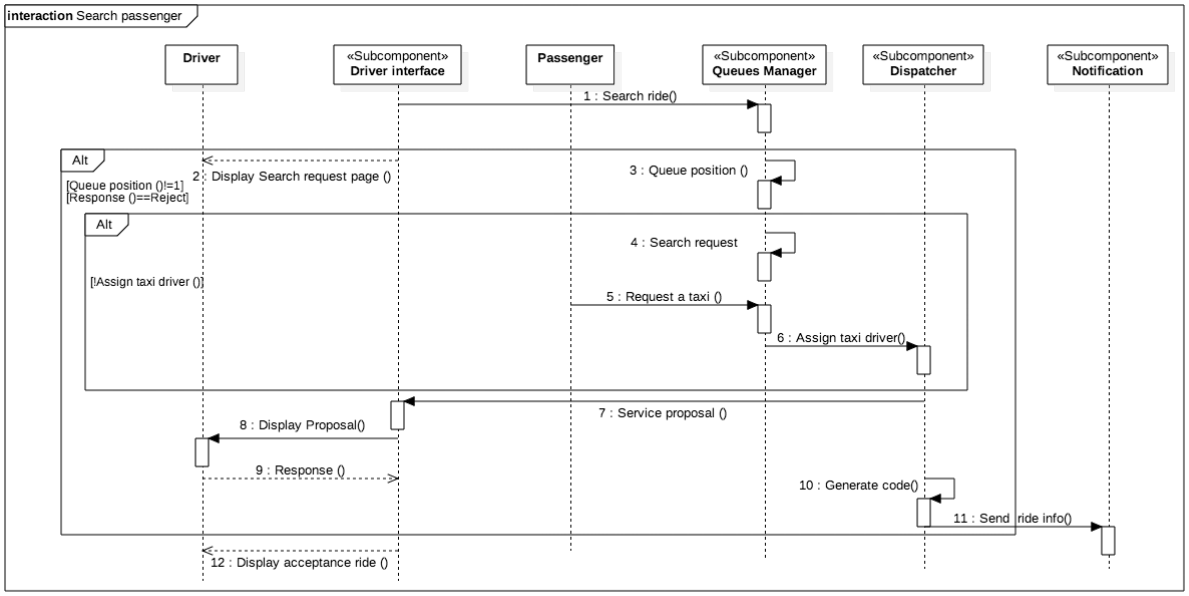

Figure 9. Search passenger sequence diagram.

2.6.7. Notification about ride

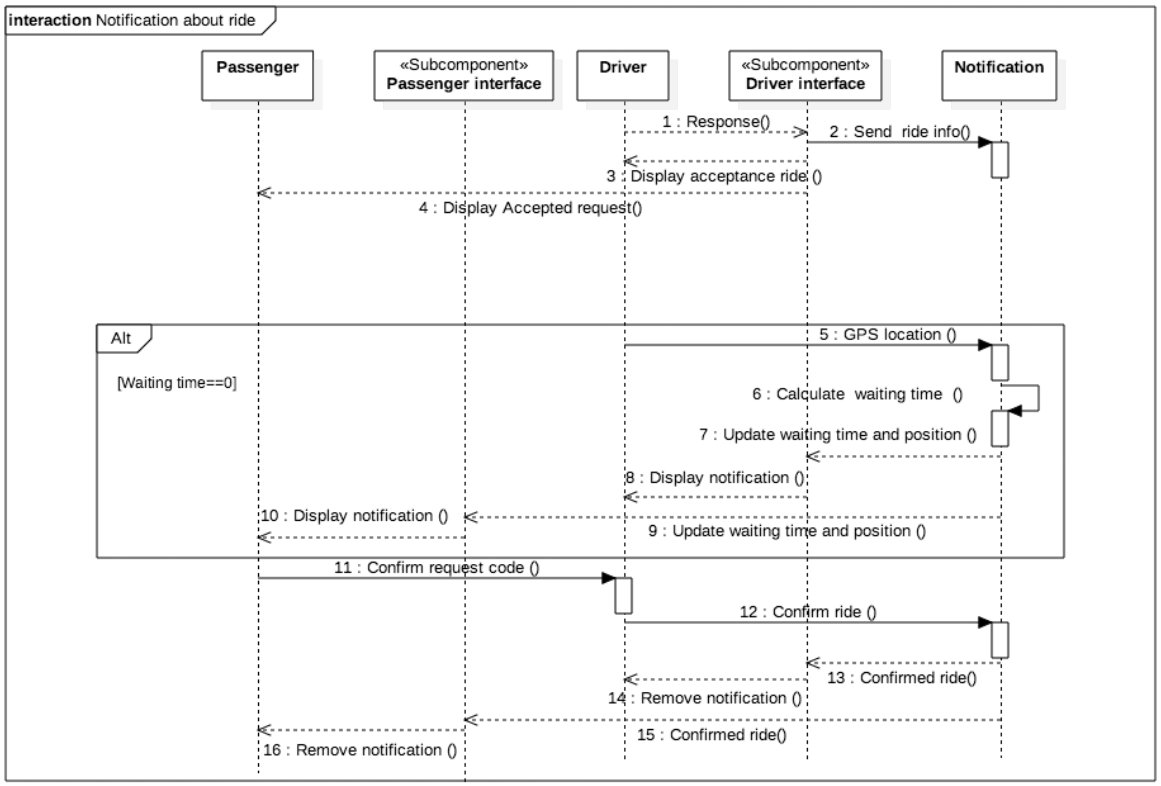

Figure 10. Notification of ride sequence diagram.

2.7. Component interfaces

Client Tier: Is composed of the XHTML pages that the normal user will see or cellphone interfaces. This is strictly related to the logic Tier, that is the one that provides all the info displayed here.

Logic Tier: Includes Web servlets and the web beans. This tier receives the requests of the client tier and has some beans which listen to these events and display data regarding the user requests. They retrieve the information. This tier also has all the logic underlying our application; it is responsible of communication with Client Tier and EIS Tier.

EIS Tier: Is composed of the entity beans and database of the system that I call “database” in general, the beans represent the connection to the database, which connects to the Database, to insert, update, delete, select. The database itself is composed of tables composing the database we generated from the needs of the project.

3. Algorithm Design

3.1. The city zones

Having the vectors for the central position Clat[Max_zones] (latitude) and Clong[Max_zones] (longitude)in a xy position format of each zone of the city, with this algorithm every time the position of the taxi changes it can calculate the number of the zone for the new position.

```
Char Taxi_zone_assign  ( float latitude_taxi, float longitude_taxi){
float posx=0,posy=0;
char zone=0;
        Convert_gps_to_xyposition(latitude_taxi, longitude_taxi, posx*,posy*);

        For(i=1;i<=Max_zone;i++){
        If(posx<=Clat[i]+2) && (posy <=Clong[i]+2 || posy >=Clong[i]-2  ){
        zone=i;
        return zone;

        }
        else If(posx>=Clat[i]-2 && (posy <=Clong[i]+2 || posy >=Clong[i]-2  ){
        zone=i;
        return zone;

        }
        return 0;
}
```

4. User interface Design

To see all the interfaces of the system, refer to the RASD document section 3.1. Where are shown and explained all the interfaces for the passengers and the taxi drivers, presenting the Web version and the one for the mobile app.

In the next figure it's seen the general operation of the system, for any visitor, the idea with this is to give a general idea of the flow that can be done in the system, from the time that a visitor uses the system for first time until when the user erases his account, passing by the request of a taxi or the modifying of his personal information.

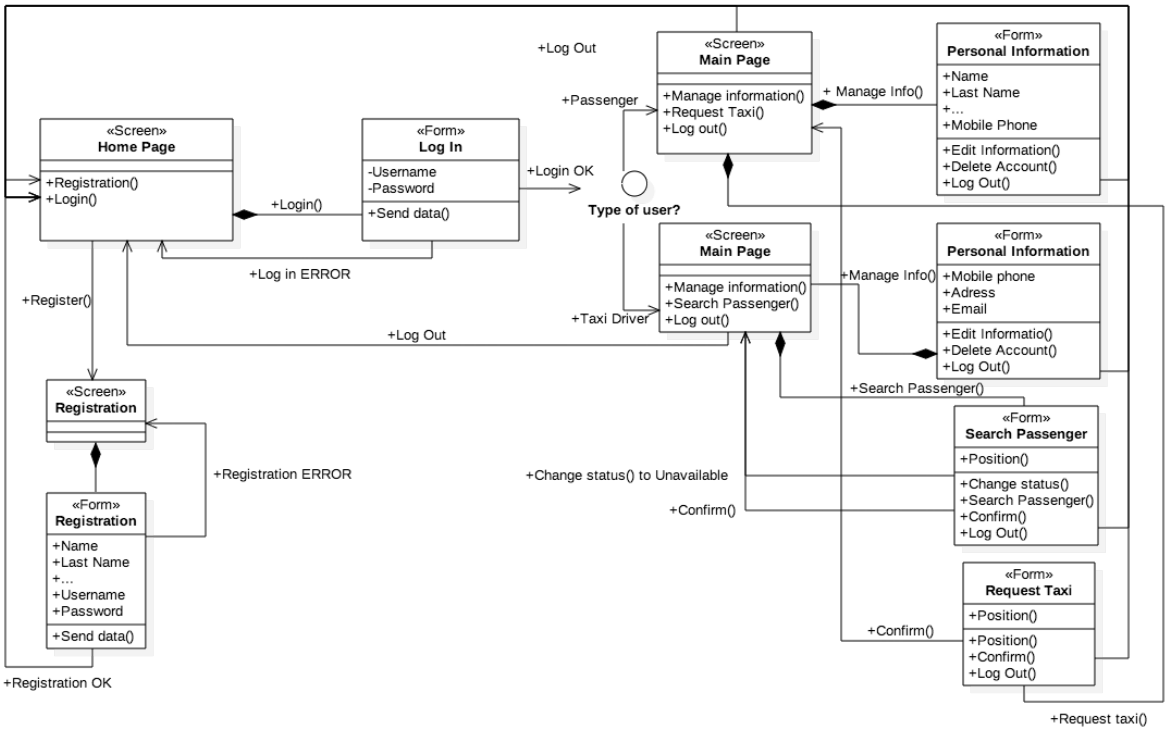

Figure 11. General user experience Diagram.

5. Requirements Traceability

5.1. Software System Attributes

5.1.1. Reliability

The system will be developed with redundancy, periodical backups, and all the functions must pass several test to be implemented in the program, to avoid algorithm problems as possible, for that reason the information is save in the cache of the logic tier and often saved also in the database.

5.1.2. Availability

My taxi service will be accessible anytime. All the service will be hosted in the cache of the logic tier primary, but is also will have the information on the database as a backup plan in case the servers goes down, the database will recover the information to start working at any time they would be need it, the system when starts send a message to the users and restart the services as soon as possible.

5.1.3. Security

The app is password-protected, there is a method to authenticate the user identity and specially the driver's identity. The system will guarantee the safety of the user's data according with the agreement accepted at the registration.

5.1.4. Maintainability

The system will be very maintainable thanks to a modular structure. The application will be documented in a way that will truly help the future developers, explaining how the application works and how it has been developed, in addition, the code will be periodically examined and improved, with a proper documentation.

5.1.5. Usability

The application is meant to be as easy and intuitive to use as possible. The drivers have capacitation for improve their performance with the app. All the users can provide feedbacks to improve the app.

5.1.6. Portability

The software for the system will be developed using the Java language and related dependent technologies because Java is designed to have as few implementation dependencies as possible. Meaning that code that runs on one platform does not need to be changed to run on another.

6. Software used

The following software were used to redact and do the content of this document:

- StarUML: to create Use Cases Diagrams, Sequence Diagrams and Class Diagrams.
- Microsoft Word: this editor was used to redact the document.
- Mockups.com: to create interface mockups.

7. References

[1] Distributed Multitiered Applications - The Java EE 6 Tutorial
https://docs.oracle.com/javaee/6/tutorial/doc/bnaay.html, Web. 1 Dec. 2015.