

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:

Luca Luciano Costanzo

Simone Disabato

Code:

789038

852863

November 21, 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
2.4	Deployment view	8
2.5	Runtime view	8
2.6	Component Interfaces	9
2.7	Selected architectural styles and patterns	9
3	Other Info	10
3.1	Working hours	10
3.2	Tools	10

List of Figures

2.1	High level architecture	6
-----	-----------------------------------	---

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 ??, called User Interface Design, describes all the graphical interfaces, 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

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.

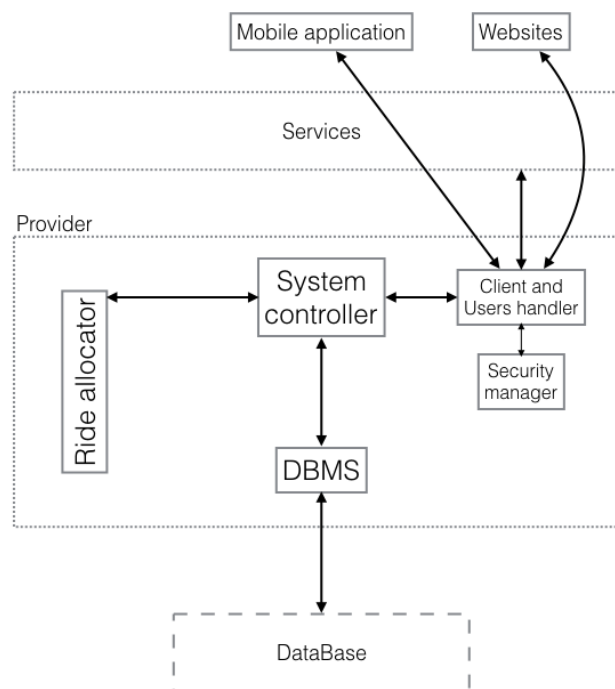


Figure 2.1: High level architecture

In the ?? 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 model¹. 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

¹see 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 use. 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

TESTO NON UFFICIALE

qui si descrivono nel dettaglio tutti i vari componenti, specialmente il data tier e il server tier.

2.4 Deployment view

TESTO NON UFFICIALE

Distribuzione dei vari componenti

Ci sono più macchine per il server? Ci sono altri componenti come firewall? ecc

2.5 Runtime view

TESTO NON UFFICIALE

qui si pone particolare attenzione alle screen e quindi al client tier.

Tutti i sequence diagram e gli UX saranno messi qui

2.6 Component Interfaces

TESTO NON UFFICIALE

descrizione di tutte le interfacce tra i vari component e verso l'esterno.
Forse si metteranno anche i principali metodi?

2.7 Selected architectural styles and patterns

EVENTBASED per ora

Chapter 3

Other Info

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

3.1 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
Total	6.30h	7h
Global	35.30h	36h

3.2 Tools

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

- L^AT_EX and TexStudio editor
- starUML