Politecnico di Milano

Software Engineering 2 Project
a.y. 2015-16

MyTaxiService
Design Document
Version 1.0

Casati Fabrizio, 853195
Castelli Valerio, 853992

Referent professor: Di Nitto Elisabetta

November 24, 2015

Contents

Chapter 1

Introduction

1.1 Purpose

This document represents the Software Design Description (SDD), also simply called Design Document, for myTaxiService.

The purpose of this document is to give all the interested parties a closer look at how myTaxiService is designed and architected, with a particular emphasis on what design decisions the development team has made and the rationale behind them.

1.2 Scope

1.3 Definitions, Acronyms, Abbreviations

1.4 Reference Documents

1.5 Document Structure

In order to let stakeholders easily navigate through the document and find all the relevant information, we will now briefly discuss its structure and give a short description of each section.

In the Architectural Design section, we will go in depth describing how the system is designed from different point of views. In particular, we will provide:

- A general overview of how the system is architected from a high level point of view
- A description of the main components that make up the system, their inner structure and the relations between them

- A view of how the components of the system are actually deployed on the physical infrastructure
- A detailed view of the normal runtime conditions in which the system operates, with sequence diagrams of the main tasks
- A list of the significant architectural styles and patterns that have been chosen to implement the system

In the Algorithm Design section, we will focus on defining the most relevant and critical algorithms that drive the system operations. In particular, for each of them we will provide a short pseudo-code representation outlining their key steps.

In the User Interface Design, we will mainly reference the existing UI sketches that were already defined in the RASD and further refine them.

Finally, the Requirements Traceability section will explain how our software architecture fulfills the requirements that were identified in the requirement analysis phase and how those requirements have influenced our design decisions.

Chapter 2

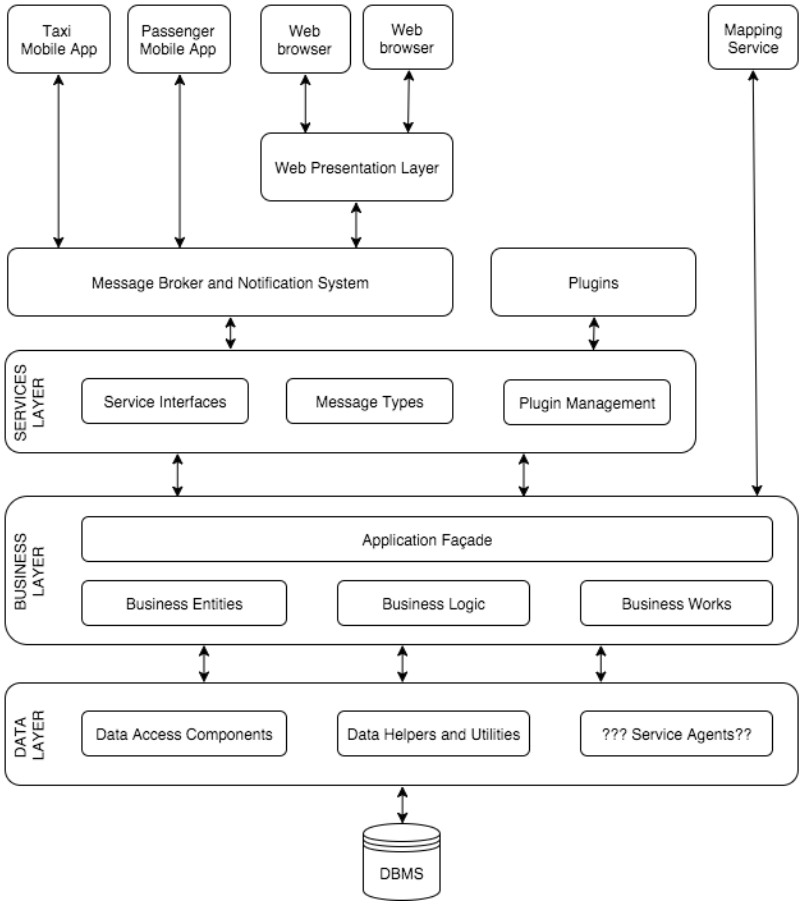
Architectural Design

2.1 Overview

In the following paragraphs we present a general overview of how the system is architected, with specific focus on the distinction between logically separated layers.

Specifically, myTaxiService has been envisioned from the ground up to be fully scalable and easily deployable on any number of servers. This characteristic is not only desirable, but actually fundamental for the system to fully accomplish the tasks it's been designed for. In order to guarantee this level of flexibility and modularity, we have settled for a system architecture that inherently enables a wide range of hardware configurations for all kinds of workloads.

The system architecture is thus logically organized in a set of interplaying software layers, whose detailed description is hereby presented.



The lowest layer of the architecture is composed by the relational **Database Management System (DBMS)** that supports all data storage operations. This component fully supports ACID distributed transactions and is meant to be run inside a Demilitarized Zone (DMZ) which is separated from the rest of the system for security reasons. (N.B. questo va messo qui?) In order to provide a higher level of abstraction to all components that need access to data and to be as platform agnostic as possible, the DBMS interface is not directly exposed to the classes that implements the business logic of the system. Instead, an intermediate **Data Layer** is responsible of performing queries on the DBMS while exposing a more flexible, customized interface to the upper layers.

The **Business Layer** implements all core functionalities of the system. In particular, all operations related to handling taxi requests and reservations, taxi availability and zone management are performed by components of this layer. Data is stored and retrieved using the APIs exposed by the Data Layer. Core functionalities are exposed to clients through a unified Application Faade that allows fine-grained tuning of which operations can

be invoked from outside the central system.

The Business Layer also depends on an external **Mapping Service** for the implementation of reverse geocoding and waiting time estimation operations. This external service is directly invoked by classes of the Business Layer by using a public API provided by the Mapping Service itself.

However, not every functionality exposed by the Application Faade may actually be made publicly available to every client. In principle, several levels of permissions could be offered to third parties while maintaining control over private APIs which should only be used by "official clients". Read-only reporting functionalities, for example, could be made available to any requesting party, while more critical operations could be offered only to selected developers after having verified certain requirements are satisfied. Furthermore, specific sets of APIs could be made available only to approved plugins and not be offered to remote services. For this reason, the **Services Layer** provides a comprehensive interface to all kinds of third party services and plugins by carefully defining which methods are available for remote and local invocation, what protocols should be followed or invoking them and what kind of messages can be exchanged between a remote component and the central system.

While locally invokable APIs are made available only to plugins, remotely invokable APIs are also offered to components living outside the perimeter of the central system. The **Message Broker and Notification System** is precisely concerned with guaranteeing an efficient and reliable communication channel with these remote entities by supporting message queues, publish/subscribe communications and dynamic, asynchronous event notifications.

The **Web Presentation Layer** is responsible for the implementation of the web application. It generates the dynamic web pages, offers them to the client via a web server, accepts requests and forwards them to the business layer by means of the communication layer and of the services layer.

Finally, the **Mobile Applications** let users access the functionalities offered by the system on their smartphones and tables in a native way. As for the web application, they interact with the central system using the communication layer and the services layer.

2.2 High level components and their interaction

2.3 Component view

2.4 Deployment view

2.5 Runtime view

2.6 Component interfaces

2.7 Selected architectural styles and patterns

2.8 Other design decisions

Chapter 3

Algorithm Design

Chapter 4

User Interface Design

Chapter 5

Requirements Traceability

Chapter 6

References

Appendix A

Hours of work