­



Politecnico di Milano

A.A. 2015-2016

Software Engineering 2: “MyTaxi”

Design Document

Manzi Giuseppe (mat. 854470) &

Nicolini Alessandro (mat. 858858)

CONTENTS

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

2. Architectural design 4

2.1 Overview 4

2.2 High level components and their interaction 4

2.3 Component view 4

2.4 Deployment view 4

2.5 Runtime view 4

2.6 Component interfaces 5

2.6 Selected architectural styles and patterns 5

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to provide the design and the architecture of the application, describing and justifying the reason of our choices. A design is a conceptualization of the system that embodies its essential characteristics, demonstrates a means to fulfil its requirements, serves as a basis for analysis and evaluation and can be used to guide its implementation.

This document also shows the architecture of the system we are implementing, so its fundamental concepts and properties in its environment embodied in its elements, relationships, and in the principles of its design and evolution.  
The document is intended for both developers, who will implement the software, and manager, who need understand the high-level structure of the system.

## 1.2 Scope

The scope of the design model of the system that we will describe in this document is to fulfil the requirements described in the RASD. Critical points are:

* The system must be web-based
* The system must handle with many users at the same time.
* The software we will develop will be a quite large-scale application that can become even larger if future implementations will be realized.
* The application must provide efficient algorithms to manage queues.

# 2. Architectural design

## 2.1 Overview

[…]

## 2.2 High level components and their interaction

* **PagesCreator** is the component in which creates pages for the browser interface and receives requests and data submitted form the browser.
* **Authentication** is the component that is called by the PagesCreator or the MobileApp when username and password are provided by the client, checks the user’s category through the Accounting in the DBMS, if all the field are correct, gives the new page and the AUTHORIZATION to the client.
* **RequestsManager** is called by the PagesCreator or by the MobileApp and creates instances of the Request class (that generalize GuestRequest, RegisteredRequest and Reservation) to be enqueued calling the RequestQueueManager. It also makes the association between Requests and Taxis, thanks to the connection with the TaxiQueuesManager. It communicates with PagesCreator and MobileApp to notify users about the association.
* **TaxiQueuesManager** creates a queue for each ZONE, in which all the taxis that are in that zone are enqueues. If a taxi exceeds the limit of the zone his id is moved in the new zone and eliminated from the past one.
* **RequestsQueuesManager** creates a queue for each ZONE, in which all the requests for that zone are enqueues.
* Accounting stores in the database all the taxis and users personal data and looks for them when requested.
* **PastRequests** stores in the database all the past requests of the registered costumers and looks for them when requested.

## 2.3 Component view

[…]

## 2.4 Deployment view

[…]

## 2.5 Runtime view

[…]

## 2.6 Component interfaces

Authentication:

* void checkCredentials(String username, String password)

RequestsManager:

* void createGuestRequest(GuestUser user, Position pos, int[] phoneNumber)
* void createRegisteredRequest(RegisteredCostumer user, Position pos)
* void createReservation(RegisteredCostumer user, Position pos, Time t)

RequestQueuesManager:

* void **enqueueRequest**(Request req)
* void **dequeueRequest**()

TaxiQueueManager

* void enqueueTaxi(TaxiDriver t)
* void **enqueueTaxiAsFirstItem**(TaxiDriver t)
* void dequeueTaxi()

Accounting

* void lookForAccount(String username)

PastRequests

* void storeRequest(RegisteredRequest r)
* void storeReservation(RegisteredReservation)
* void lookForListOfRequests(String username)

\*\*\*ViewInterface (implemented by PagesCreator and MobileApplication)\*\*\*

* void createHome()
* void createPersonalAreaHome(Class typeOfUser)
* void createNewRequestPage()
* void createNewReservationPage()
* void createWaitingPage()
* void createEnqueuedRequestPage()
* void askConfirmation(String taxiDriverUsername, Position pos)
* void notifyTaxiId(String id)

## 2.6 Selected architectural styles and patterns

We selected the 3-Tier style. We chose this style because it provides:

* **Performance**: network utilization is minimized and the load is reduced on the Application and Data tiers.
* **Flexibility, maintainability** and **scalability**: since presentation, application and data layers are deployed on different tiers, it is quite easy to make changes to a layer without affecting the others.
* **Security**: because data are stored in the bottommost tier.
* **Manageability**: the project implementation can be divided into simpler projects, that could be assigned to different programmers or programming teams (divide et impera principle).
* **Reusability**: it is easy to share and reuse the components and services.
* **Decoupling**: dividing presentation, application and data layers according to MVC pattern reduces the possibility of interdependency between classes.

The approach to architectural design we provide is an object-oriented approach, that enriches Object Oriented Analysis described in RASD (especially in chapter 3.4) with architectural details. The advantages of this pattern are:

* Making problems simpler by dividing them in sub-problems and approaching them independently (Divide et impera principle).
* Having a high level of abstraction.
* It is easy to reuse components and classes generated by this approach.

The platform that, according to us, better fits our decisions is Java EE ver. 7. The benefits of using this platform are:

* The system will be maintainable in terms of updating the application and the website.
* It is heavily object oriented (according to the chosen style).
* Good grade of performance compared to other possible solution.
* Many API are provided.
* It is good for large-scale application (our application is not much large, but neither so small).