SmartCityAdvisor

**A close up of a logo

Description generated with very high confidencePOLITECNICO** MILANO 1863

ASSIGNMENT 2 – DD

Design Document

A Software Engineering II Project,

made by Luca Favano 899224

V1.0 - 28/04/2018

Table of Contents:

[1 - Introduction 3](#_Toc513385714)

[1.1 Purpose 3](#_Toc513385715)

[1.2 Scope 3](#_Toc513385716)

[1.2.1 Goals 3](#_Toc513385717)

[1.3 Definitions, Acronyms, Abbreviations 3](#_Toc513385718)

[1.3.1 Definitons 3](#_Toc513385719)

[1.3.2 Acronyms 4](#_Toc513385720)

[1.4 Revision History 4](#_Toc513385721)

[1.5 Reference Documents 4](#_Toc513385722)

[1.6 Document Structure 4](#_Toc513385723)

[2 – Architectural Design 6](#_Toc513385724)

[2.1 Overview 6](#_Toc513385725)

[2.2 Component View 7](#_Toc513385726)

[2.3 Deployment View 10](#_Toc513385727)

[2.4 Runtime View 10](#_Toc513385728)

[2.5 Component Interfaces 10](#_Toc513385729)

[2.6 Selected Architectural Styles and Patterns 10](#_Toc513385730)

[2.7 Other Design Decisions 10](#_Toc513385731)

[5 – Effort Spent and References 11](#_Toc513385732)

# 1 - Introduction

## 1.1 Purpose

The purpose of this Design Document (DD) is to show the architecture that lays below the application. It will show in details which components are necessary, how they interacts together and the algorithms used in the application. This document will also provide details of the user interface chosen.

All the requirements previously shown in the RASD document will then be specified more in details.

This document is particularly useful to project managers, developers and testers.

## 1.2 Scope

SmartCityAdvisor is a server-sided system that offers an app that let Server and Clients interact together. This interaction require an interface that can be easily accessible by any device that satisfy the given hardware requirements.

To develop this application we need a system that is both flexible and easily maintanable, so that in the future it could be easily improved and functionalities could be added as planned by the SmartCityAdvisor organization. Particular care should be applied to the application architecture which code should use coding best practices and principles like encapsulation.

The main issue that will be discussed in this documnet is how to make all the system work together flawlessly, in fact the system should provide the services in a reliable and easy to access way while keeping everything synchronized using an Internet connection.

### 1.2.1 Goals

**[G1]** Limit the traffic accordingly to current CO2 levels in Milan.

**[G2]** Warns the citizens of a change of the current status

**[G3]** Provide users with up-to-date information retrieved from the sensors

**[G4]** Manage the traffic accordingly to special events taking place

**[G5]** Simplify the process of finding and reserving an available parking spot

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitons

* **Sensor:** A component of that communicates with the system at regular intervals providing useful information for the system to work correctly**.**
* **Actuator:** A component that allows the system to have an impact on the environment based on choices that can be taken automatically or manually,
* **City Center:** An area of Milan which characteristics need to be preserved and kept constant over the time.
* **CO2 Level:** Current amount of CO2 particles detected by the sensor and measured in ppm (particles per million).
* **Superuser:** A user of the app that has higher privileges comparet to a “normal” user which is a common citizen that downloaded the app.

### 1.3.2 Acronyms

* **RASD:** Requirement Analysis and Specification Document
* **XMPP:** Extensible Messaging and Presence Protocol
* **HTTP:** Hypertext Transfer Protocol
* **DBMS:** Database Management System
* **UML:** Unified Modeling Language
* **IEEE:** Institute of Electrical and Electronic Engineers
* **DD:** Design Document
* **UX:** User Experience

## 1.4 Revision History

* **V1.0:** First Version, Released on ??/05/2018
* **V1.1:** Typo Correction, Released on ??/05/2018
* **V1.2**: Final Version, Released on ??/06/2018

## 1.5 Reference Documents

* **Specification Document:** “Assignment-for-the-second-project.pdf”
* **Alloy Specification:** on beep
* **Past Project Examples** taken from beep
* **IEEE Recommended Practice** for Software Requirements Specications.
* **RASD Document** that is also part of the project

## 1.6 Document Structure

This document will be structured into five different sections:

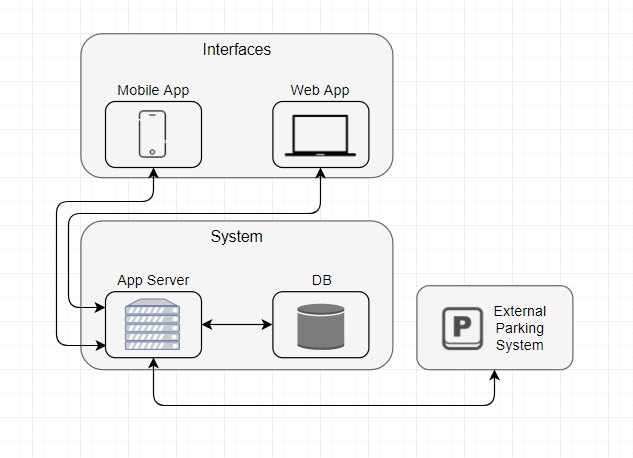
* **Section 1: The introduction**, a brief description of the project, the terms that will be used and some useful knowledge.
* **Section 2: Architectural Design**, it shows the main components of the system their design and how they interact. It includes the patterns and other choices made regarding the architecture.
* **Section 3: Algorithm Design**, a description of the most interesting parts of the algorithms needed to implement the system.
* **Section 4: User Interface Design**, a demonstration of how the user interface is going to look when completed, through mockups and UX diagrams.
* **Section 5: Requirements Traceability**, an explanation of how the decisions that have already been took in the RASD document now reflects to design elemnts included in this document (Design Document).
* **Section 6: Implementation, Integration and Test Plan**, aiming at identifying the planned order for the implementation and integration of the system components. Also including the testing of them.
* **Section 7: Effort Spent + References**, containing all the information used while writing the document and the time spent doing it.

# 2 – Architectural Design

## 2.1 Overview

This chapter focuses on the architectural structure, components will be described explaining how they interact with each other. The system will be illustrated both phisically and logically

In particular a simple high level representation of the entire SmartCityAdvisor system could be as follows:



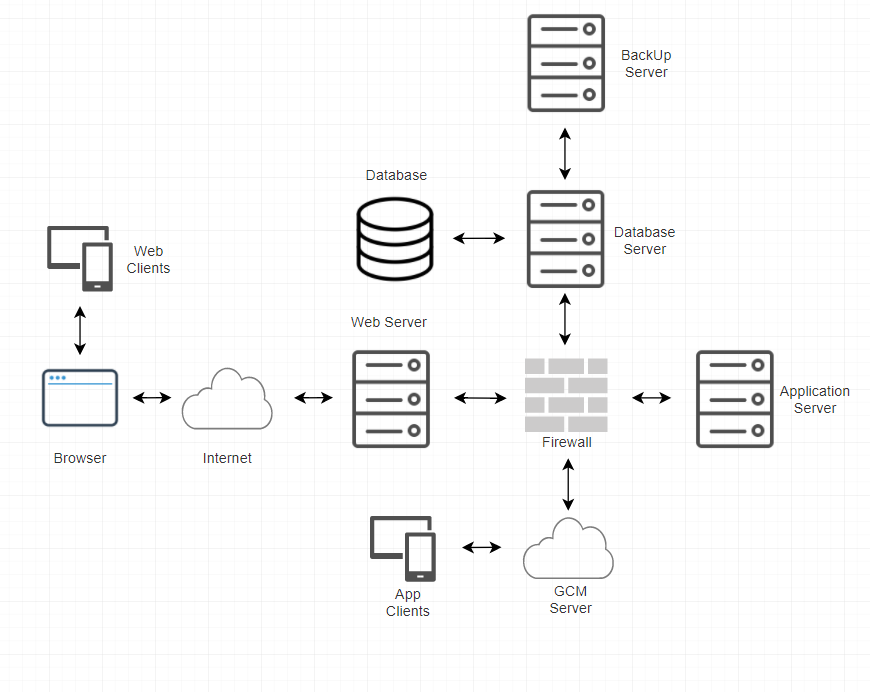
* **Interfaces:** Client side part of the application, it’s installed on user devices or loaded through a browser, it offers services that need the communication with the system.
* **Application Server:** Contains the server side logic of the system, implements useful APIs and handle the client requests.
* **Database:** Is responsible of storing and retrieving asked data, it doesn’t implement any logic and it is only used for data storing. DB guarantees the ACID properties.

The given simple overview is not detailed enough to describe accurately the whole sysyem and its specifics will be addressed carefully in the following paragraphs.

## 2.2 Component View

First we need to provide details about all the physical components of the system and this can be achieved thanks to the physical architecture diagram:

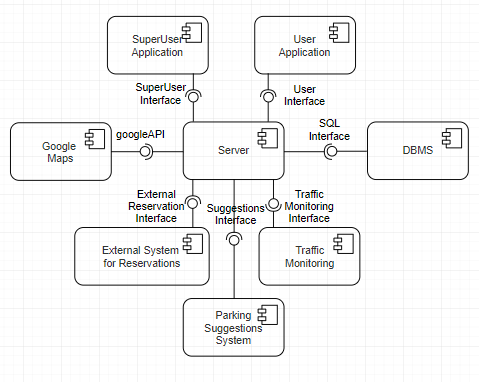
### 2.2.1 Physical Architecture Diagram



As shown in the diagram we need to introduce more components to our system:

* **Web Server**: A software running on a server that is able to communicate with the web application clients using the HTTP protocol.
* **Browser**: An application that retrieves web pages and presents them to the users.
* **GCM Server**: Google servers that take messages from an application server and sends them to a client app that is running on users devices. Useful to handle app services like the notifications.
* **Firewall:** Software that protects the system from unathorized access
* **Web/App Clients:** Two different clients that will interact with our system through an User Interface

Focusing on the interfaces provided by the server we obtain the following diagram that shows a representation of the logical architecture.



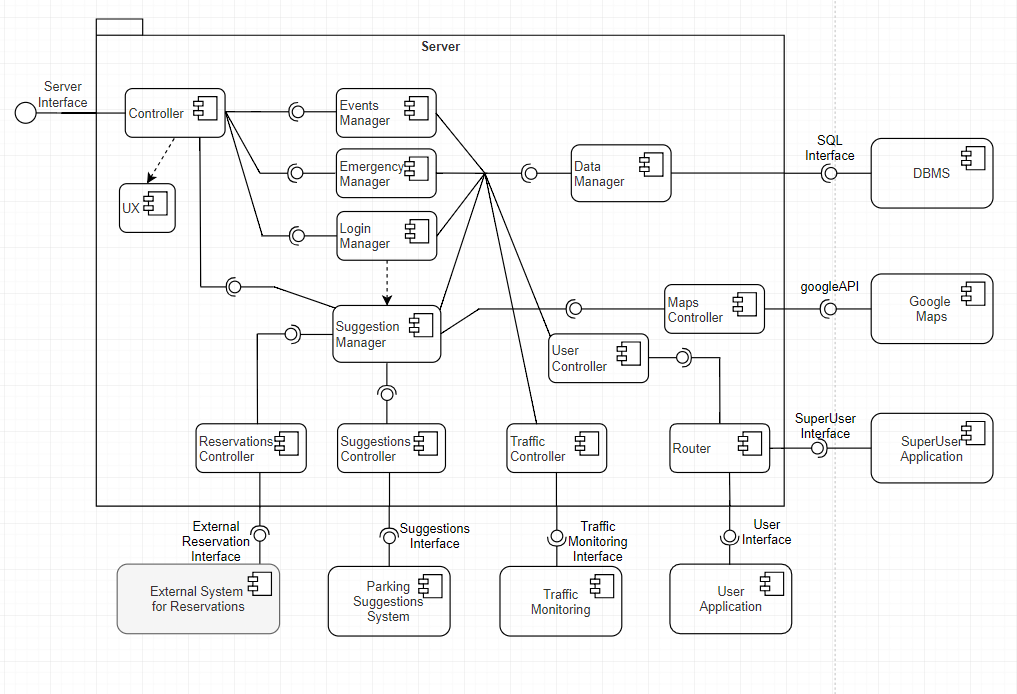
The obtained system include different components that should be integrated:

* **Google Maps:** A possible map system that is provided by Google and is accessible through google APIs. Useful to show to the user maps linked to the sugesstions of parking lots.
* **Database Management System (DBMS):** Used to access all the necessary data in an efficient way.
* **User/SuperUser Application:** A component that takes care of handling all the functionalities that are available to the different types of users.
* **Traffic Monitoring System:** A component that models the the traffic monitoring system parameters, allowing changes of system status.
* **External System:** A system not handled directly that allows our system to request reservations in parking lots available in Milan.
* **Parking Suggestions System:** A component that handle the suggestions when requested, the best possible choices should be elaborated.

Three logical level are still preserved (User Application / Server / DBMS).

### 2.2.2 Application Components Diagram

Lastly we analyze the components that together form what we know as “server”.



Together with components previously already specified we can find few controllers and managers that handle all the information for the system to work properly.

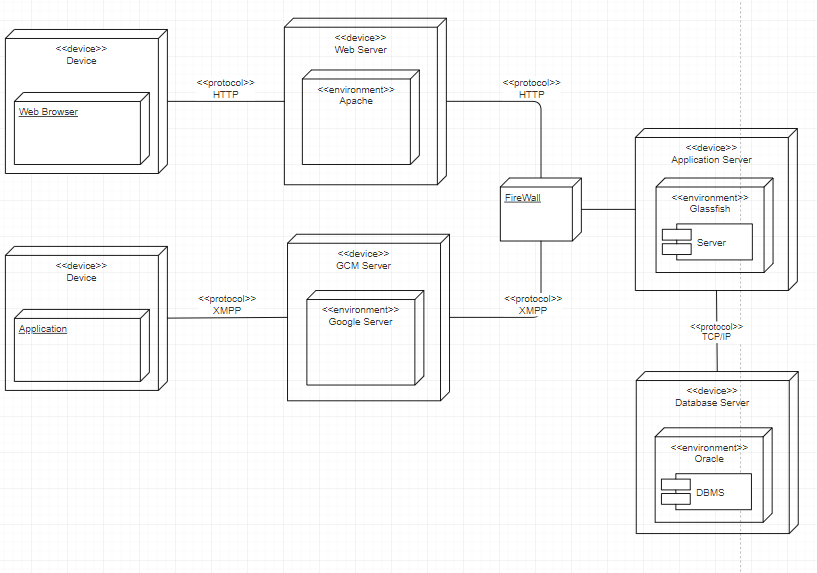
The most relevant components that we just added are:

* **A Data Manager**: A component that provides methods for accessing the data of the system and care sending those data to the server so that they can be synchronized. It collects the data of other controllers.
* **A Suggestion Manager:** A component that has to interact with many others to collect and elaborate all the necessary information, allow a reservation only if the user has logged in.
* **A Controller:** A component that manages the User Interface providing all the necessary information.
* **Router:** A component that dispatches all the requests coming from the User/SuperUser application to a Login Manager that can handle them.
* **Parking Suggestions System:** A component that provides interfaces useful to the Suggestion Manager for it to work properly.

## 2.3 Deployment View

Next we analyze how the system should be deployed, since the architecture should ensure the system security and maintanability Database and Application server run on two different machines, the Database Server also is covered by a backup server in case of any problem.

The GCM server interact with the application server and the application user devices using the XMPP protocol which can guarantee enhanced performances and is more flexible for future updates. The GCM server is also used to grant the system the possiblity to send notifications to the app users.



## 2.4 Runtime View

The most interesting dynamic behaviours istances of the system will be shown in this section. To show the most meaninful scenarios these istances may differ from the ones already analyzed in the RASD document.

### 2.4.1 User Registration and Login

## 2.5 Component Interfaces

## 2.6 Selected Architectural Styles and Patterns

## 2.7 Other Design Decisions

# 5 – Effort Spent and References

This project has been developed by a single member because of a lack of other teammates, probabily because not many others students are taking the second delivery of this year project.

The amount of time needed to complete this document is about 90-100 hours, many of them spent thinking which was the best way to display the information and making/modifying the needed diagrams.

Tools Used:

* The diagrams have been done using the draw.io free tool.
* The wireframes have been realized using Balsamiq.
* Microsoft Word to write this document

The commits on github are symbolical of the major steps of the work, as taking the project alone didn’t really require to update the other members and share the work.

A small personal thought is that i feel this kind of projects are much more useful than plain teaching, they are quite funny to develop (this is very personal) and offer many possibilities to look for interesting information online, while learning to use useful tools.