POLITECNICO
MILANO 1863

Software Engineering 2

Design Document

Amorosini Francesco
Casali Alice
Fioravanti Tommaso

8 December 2019

Introduction

1.1 Purpose

While the RASD presented a general view of the SafeStreets appliction and its features, this document aims to further analyze the system's design and architecture, by describing for each of its components their runtime behaviour, integration, interfaces, implementation and testing plans. This document is mainly intended to be used by the test and development teams as a guidance in the development process, but also to prevent structural degradation during maintainance and extension phases. Nonetheless, the document is also addressed to all the stakeholders who are interested in supervising the development process.

1.2 Scope

SafeStreets: an application that aims to improve the safety of urban areas by giving its Users the possibility to report traffic violations to Authorities. Users are logged in either as Citizen, those who report the violations, and Authorities, those who are notified about newly reported violations and are supposed to take action on them.
The system is in charge of collecting all the Reports, storing them, and notifying the Authorities about them. The stored Reports are then used to build statistics, find unsafe areas, and compute suggestions on how to improve the safety of such areas. The system can also communicate local Municipalities' Systems in order to retrieve information about accidents and iussued traffic ticket: in this case some of the above functions are enhanced, and some new functions are enabled (g.e computing statistics on traffic ticked).

Further information about the scope of the application can be found in the Chapter 1 of the RASD.

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

- *Client*: a piece of computer hardware or software that accesses a resource or a service made available by a Server.
- *Server*: a device or a computer program that provides resources or functionalities to other programs or devices.
- *Reverse Proxy*: particular proxys that are responsible of forwarding requests to one or more Servers which will handle it.
- *Firewall*: a network security systems that monitors incoming and outcoming network traffic, applying predefined predefined security rules.
- *Port*:

1.3.2 Acronyms

- **RASD**: *Requirement Analysis and Specification Document*, the document in which all the requirements and goals of the application are throughly described.
- **API**: *Application Programming Interface*, interface, or communication protocol between Client and Server intendend to simplify the building of the Client-side software.
- **OCR**: *Optical Character Recognition*, software dedicated to the detection of characters contained in a document and to their transfer to digital text that can be read by a machine. In this context, OCR will be used to read license plates.
- **UML**: *Unified Modeling Language*, is a standard visual modeling language intended to be used for analysis, design, and implementation of software-based systems.
- **GPS**: *Global Positioning system*, technology widely used to get the user's position.
- **DBMS**: *Data Base Management system*, software that provides organized space memory to store information.

1.3.3 Abbreviations

- [R_i]: i-th requirement.

1.4 Reference Documents

- Specification document: *SafeStreets Mandatory Project Assignment.pdf.*
- Requirement Analysis and Specification Document: *RASD.pdf.*

1.5 Document Structure

This document is presented as it follows:

1. **Introduction** presents a general overview, the scope and the purpose of the document.
2. **Architectural Design** shows the main components of the system and their relationships. This section will also discuss the arichitectural choices of the design process.
3. **Algorithm Design** presents and discusses the algorithms that will enable the system's functionalities.
4. **User Interface Design** provides some further details on the user interface defined in the RASD.
5. **Requirement Traceability** maps all the functional requirements defined in the RASD over the components that will accomplish them.
6. **Implementation, Integration and Testing Plans** shows the order in which the implementation and the integration of the components will occur, and how the testing phase will be carryed out.
7. **Effort spent** displays the time spent writing this document by each member of the team.

Architectural Design

2.1 Overview

In this chapter the architectural structure of the system will be discussed at multiple levels of abstraction. A high-level view of the components and their interactions is represented in Figure 2.1. The details will be explained in the next sections.

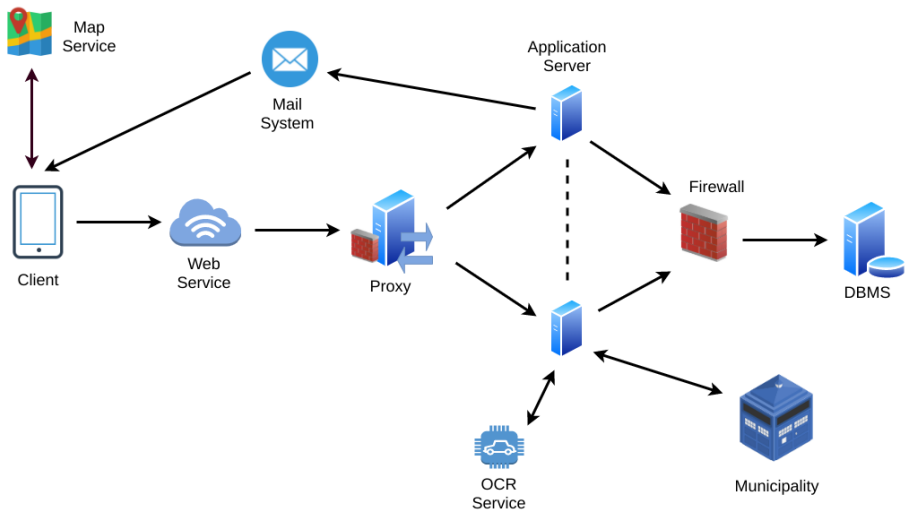

Figure 2.1: Overview of the system.

2.2 Component view

2.3 Deployment view

2.4 Runtime view

2.5 Component interfaces

2.6 Selected architectural styles and patterns

The following architectural patterns are used for defining the structure of the system:

2.6.1 Client-Server Architecture

In this structure there are two main roles with different duties, the client uses data and services offered by the server. The server stores all data in order to provide them to the client.
Motivations: this pattern dues to several advantages. Server can be replicated to have more scalability. Accessibility: users can use the services without hte need for a specialized device or client. The security is granted because the server allows only registered user to consult data. Furthermore, it is easy to update, replace, repair, move a server without affecting clients.

2.6.2 Three-tiered Architecture

This type of architecture is a client-server architecture where three tiers are phisically separated.
The *presentation tier* provide an interface to allow the user to consult data and result of services executed in a different layer. It is the top-most tier and the only one accessible from the client.
The *application tier* executes the logic of the application throw functions that elaborate data. The reverse proxy is needed to handle the client requests and to balance the workload, the requests are forwarded to the application serves in order to provide the right data.
The *database tier* includes the data persistence mechanisms and the data access layer that encapsulates the persistence mechanisms and exposes the data.

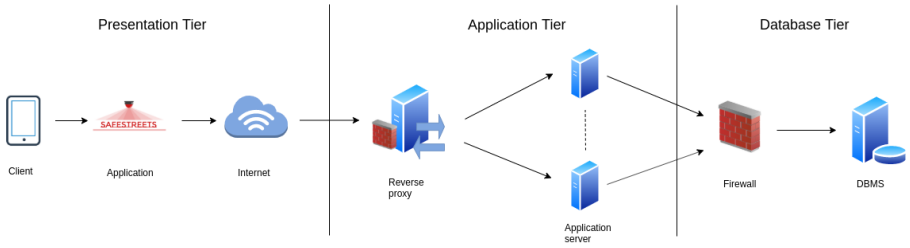

Figure 2.2: Overview of the system.