POLITECNICO
MILANO 1863

SafeStreets

Software Engineering 2 Project - Prof. Matteo Rossi

DD Document

Salvatore Fadda - 944786
Adriano Mundo - 944684
Francesco Rota - 948714

A.Y. 2019/2020
Version 1.0

December 9, 2019

Contents

1 Introduction

1.1 Purpose

This document represent the *Design Document* (DD). It aims at providing an in-depth description of the architecture below *SafeStreets* application and its services. It will present a section related to the architectural design with different perspective on the components of the *System*, how they interact and how they will be implemented. All the requirements of the RASD document are mapped with the components to explain how they will be satisfied. Finally, a section for the testing plan for Q&A team is provided.

1.2 Scope

SafeStreets is a crowd-sourced application that aims at keeping safe the city's streets. The idea behind this service is to allow *Users* to notify the *Municipality* when a violation occur on the streets under its jurisdiction. The *User* can notice and notify the violation by sending a photo of the violation including date, time and position. *SafesStreets* stores all the data and uses a plate recognition algorithm to recognise the image content.

The **Basic Service** allows *Users* and *Authorities* to mine the information collected by the service, so they can access statistics built from the data.

As **AF1**, the application *SafeStreets* identifies potentially unsafe areas and suggests possible interventions to *Authorities* to solve the founded issues.

As **AF2**. the application *SafeStreets* allows the *Municipality* to generate traffic tickets directly from the application data. Also, using the data of issued tickets the *System* can build statistics and find insights to suggest to *Municipality* in order to improve their service.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Client:** a piece of computer hardware or software that accesses a service made available by a Server.
- **Server:** a computer program or a device that provides functionality and handle the requests of other programs or devices, called Clients.
- **N-Tier:** or multilayer is an architecture in which presentation, application processing, and data management functions are physically separated in n-layers.

1.3.2 Acronyms

- **GPS:** Global Positioning System
- **API:** Application Programming Interface
- **RASD:** Requirements Analysis and Specification Document
- **DBMS:** Data Bases Management System
- **GDPR:** General Data Protection Regulation
- **MVC:** Model-View-Controller
- **REST:** REpresentational State Transfer
- **Q&A:** Quality and Assurance
- **UI:** User Interface
- **HTTPS:** Hyper Text Transfer Protocol Secure

1.3.3 Abbreviations

- **[Rn]:** n-th Functional Requirement
- **A1:** Advanced Function One
- **A2:** Advanced Function Two

1.4 Revision history

Version	Date	Description
1.0	09/12/2019	First Delivery

Table 1: Revision History

1.5 Reference Documents

- Mandatory Project Assignment
- RASD Document of *SafeStreets* application

1.6 Document Structure

The other sections of the Design Document (DD) are organised in this way:

- **Architectural Design** (Section 2): an in-depth description of the System's architecture. It defines the main components, the relationship between them and the deployment of components. There are different views and levels of analysis of the components plus some subsection useful for identifying how the components interact and the architectural styles and patterns.
- **User Interface Design** (Section 3): a complementary section of what was included in the RASD. It includes the definition of the UX process through a model that represents the flows of the interfaces.
- **Requirements Traceability** (Section 4): a complementary section of what was included in the RASD. It contains all the identified requirements and show the relationship between them and design choices in order to satisfy them.
- **Implementation, Integration and Test Plan** (Section 5): shows the order of the implementation and integration of all the components and subcomponents, providing how the application will be tested.
- **Effort Spent** (Section 6): a section containing a table for identifying the hours and the effort spent by each team member to deliver the DD.

2 Architectural Design

2.1 Overview

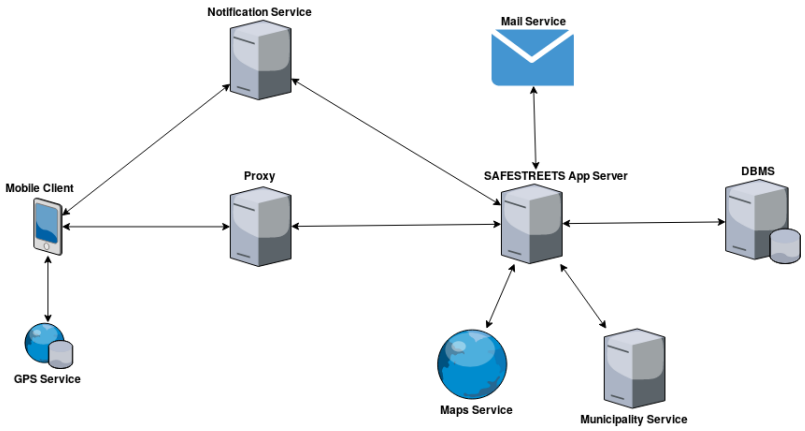

Figure 1: Overview of the System

The image above shows an high-level overview of the *System*'s architecture. The components interact with some external services.
The Mobile Client application accesses the GPS service in order to retrive geographical information and communicates with the Application Server through a Proxy.
The Application Server uses an external Notification Service to send notifications directly to the Client. It uses a Mail System service and a Map service to execute all the functions. Finally, it accesses the Municipality Service to retrieve data and to communicate with the Municipality through the an API service offered by the Municipality itself.
Further details on the *System* components will be explained in the next sections.

2.2 Component view

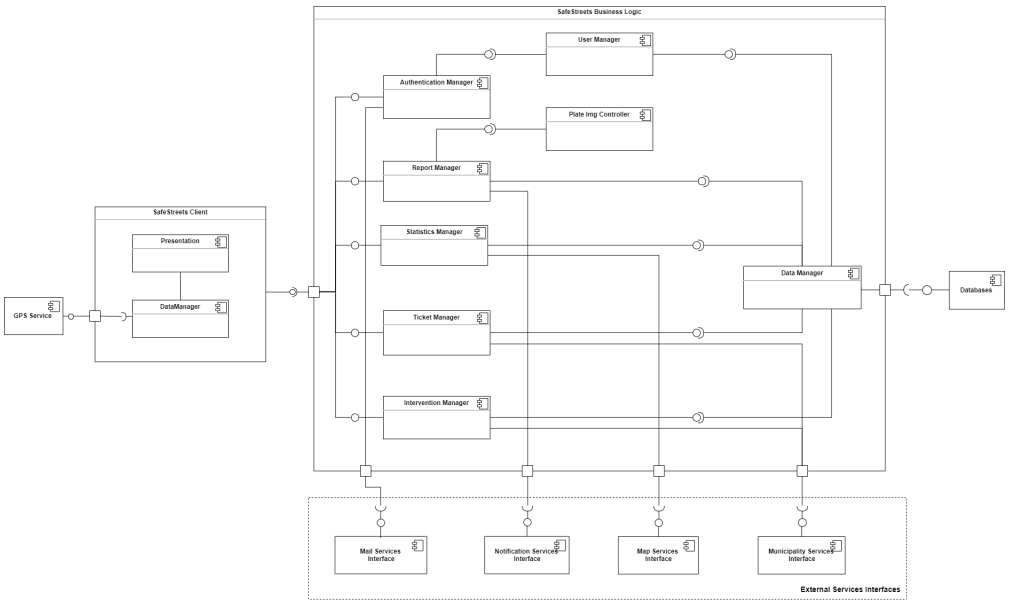

Figure 2: *SafeStreets* Component Diagram

The purpose of the UML component diagram is to capture the internal architecture of the *System*, showing the structure pf components, how they are connected together as a part of larger components.
It's divided in three main components: the *SafeStreets* Client Application, the Business Logic and the external Database.
Each main component is divided in modular components to provide specific functionalities. Components communicate between each other by providing interfaces with the required information.
This diagram uses assembly connectors for internal interfaces and delegation connectors for external interfaces.

SafeStreets Client
This component located on the *User*'s device represents the client machines that access to the Business Logic container. Its sub-modules are the Presentation and Data Manager. It is implemented as a thin-client, so does not contain any logic of the *System*.

- **Presentation:** this module corresponds to the View of the MVC Pattern, requires data from the Business Logic to display the correct UI to the *User*.
- **Data Manager:** this module has to communicate with the GPS Service in order to retrieve the data required to the application.

SafeStreets Business Logic
This component contains all the *SafeStreets* application logic. It's a module between the Client Application and the Database. It collects together all the components needed to satisfy the application functionalities. In the section below each component will be described.

- **Authentication Manager:** this component contains all the methods needed to access the application, so for the purpose of the system authentication. It's responsible for both the registrazione and log-in process. It continuously interacts with the Database through the Data Manager interface. It guarantees that all the constraints are respected and finally, when you need to confirm to the *Users* they have successfully registered, it communicates with the Mail Service Interface to send the confirmation e-mail.

- **User Manager:** this component handles all the functionalities related to the User account, both for simple *Users* and *Authorities*. Its services are performed by interacting with the Database through the Data Manager.

- **Report Manager:** this component contains the logic needed to perform and handle the violation reported by the *User*. Its service are linked to the image loading, the violation indication type and so on. It interacts with another component that's responsible for analysing the image, the Plate Img Controller and stores all the data in the Database through the Data Manager. Finally, the Report Manager communicates with the Notification Service Interface because it gives a direct feedback to the *User* about the loaded image through a push notification.

- **Plate Img Controller:** this component handles the request coming from the *User* that wants to load a violation image. The request comes from the Report Manager, in fact the component exposes the methods to verify that the image was not modified by third parties, so it makes a consistency check. In order to do this task, the component runs an algorithm.

- **Statistics Manager:** this component manage and calculate all the statistics that the application is intended to provide to both *Users* and *Authorities*. It has two main task: retrieve and store the data from the Database through the Data Manager, where are stored all the data inserted from the application's users; and to calculate all the statistics that are accessible from the UI. It manipulates all the data in order to show simple graphs or to find useful insights. It accesses to Map Services Interface because it needs to retrieve geographical information in order to show statistics directly associated with the city's zone.

- **Ticket Manager:** this component is responsible for the ticket verification by the *Authorities* thanks to reported *User* violation data. In fact he retrieve from the Database through the Data Manager all the violation reported from the *Users*, then let the verification to be done manually by

an *Authorities*, exposing all the needed functions. Finally, it communicates with the Municipality Service Interface in order to advise that the ticket is verified. Therefore the Municipality is able to generate a ticket from this information.

- **Intervention Manager:** this component handles the information coming from the Municipality, in fact it interacts with the Municipality Service Interface to retrieve data about accidents. Then, it cross the information with the data stored in the Databases and retrieved through the Data Manager in order to provide suggestions on interventions.
- **Data Manager:** this component provides all the methods to interact with the Database such as data retrieval, storage and update, it is the unique point of access to the Database.

Data Base
This component represents the DBMS, which provides the interfaces to retrieve and store the data. Data about the application and data about Users are securely stored and encrypted.

External Services Interfaces
Some components of the diagrams that represent the *SafeStreets System* and described above communicate with external components. This are are third party services that expose their API. These communications are bilateral and essential to guarantee all the functionalities.

- **GPS Service:** this interface communicates with the Client Application that have to access data from the *User*'s device, in particular GPS data through the Data Manager interface. GPS data are essential for tracking position and retrieve all suitable metadata necessary to the application to work properly.
- **Mail Services Interface:** this interface is responsible for the interaction with the *User* when it's sent a confirmation e-mail after the registration phase. It's accessed by the Authentication Manager.
- **Notification Services Interface:** this interface is needed to send push-notification to the *User* as an alert when the image processing is not done correctly, so it needs to communicate with the Report Manager in order to advise the *User* that have to re-do the process of reporting.
- **Map Services Interface:** this interface is essential for *SafeStreets*, in fact access to a mapping service is necessary to show and calculate statistics with respect to the different city's zone. It's a service accessed by the Statistics Manager.
- **Municipality Services Interface:** this interface is essential to have a means of communication with the Municipality services. It lets to retrieve data and to pass information to the Municipality in order to do all the services offered by the *System*. It's accessed by Ticket, Intervention Mngs.

2.3 Deployment view

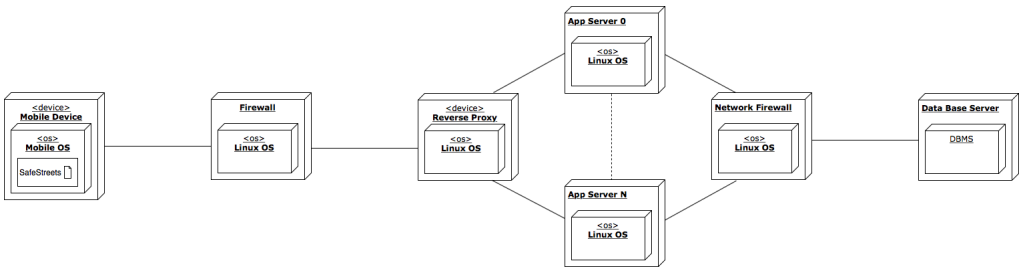

Figure 3: *SafeStreets* Deployment Diagram

The *System* presents a Multi-tier architecture. The role of each node will be specified in the next section.

Mobile Device
It represents the Client in the architecture, hosting the System's mobile application. This is equally valid for both Users and Authorities.

Firewall
It filters the access to the Reverse Proxy and is used to protect a trusted network from an untrusted network. A firewall provides protection from unauthorized requests or from malicious attacks.

Reverse Proxy
It retrieves resources on behalf of a client from the servers and balances the load of the various requests. It helps to achieve increased parallelism and scalability of the application.

Application Servers
They include all the business logic of the system, which is completely replicated to allow workload balancing.

Network Firewall
It does the same job of the Proxy Firewall but protecting and filtering the access to the DBMS.

Data Base Server
All data are stored in the Data Base Server equipped with a relational DBMS and can be retrieved with appropriate queries.

2.4 Runtime view

Account Creation Runtime View

Log-in Runtime View

Violation Report Runtime View

Maps Access Runtime View

Statistics Access Runtime View

Tickets Verification Runtime View

Intervention Suggestion Runtime View

2.5 Selected architectural styles and patterns

2.6 Other design decisions

3 User Interface Design

4 Requirements Traceability

Component (DD)	Requirements (RASD)
Component Name	•
Component Name 2	•

Table 2: Requirements Traceability

5 Implementation, Integration and Test Plan

5.1 Overview

5.2 Implementation

6 Effort Spent