SafeStreets

Design Document

Giancarlo Danese - 945265
Davide Savoldelli - 928676

A.Y. 2019/2020 - Prof. Di Nitto Elisabetta

Contents

1 INTRODUCTION

1.1 Purpose

This document represents the Design Document (DD) for SafeStreets software and contains a functional description of the system. Since we provide an overall guide of the systems' architecture, it's addressed to the software development team

1.2 Scope

SafeStreets will have an embedded algorithm which will analyze pictures of the vehicle plates sent by the user in order to recognize the vehicle. This information, together with the position of the vehicle and the type of violation that has been committed, will be stored in the software's database.
Authorities will have the chance to mine the information retrieved in the database by highlighting the streets/areas in which most of the violations are committed, the type of vehicles which commit most of the violations and which type of violations occur the most, suggesting possible interventions that could be taken.

1.3 Definitions, Acronyms, Abbreviations

- `User Device`: any compatible device with the SafeStreets application, like a smartphone or a computer

- `Personal Information`: information provided by the user during the registration process. It includes name, surname, birth date, address, e-mail address, mobile number.

- `Violation Report`: the act in which users can denounce violations on the streets, by providing the system its position, a photo and by selecting a violation from a precompiled menu

- `Mobile App`: an application that can be run by mobile devices, both smartphones and smartwatches.

- `Violations Map`: a map, accessible only by authorities, which contains notifications and alerts about all the unsafe areas where several violations are committed

1.3.1 Acronyms

- RASD: Requirements Analysis and Specification Document.

- DD: Design Document

- API: Application Programming Interface.

- GPS: Global Positioning System.

- PRA: Pubblico Registro Automobilistico

- AUC: Authority Unique Code

- DBP: Device-bound PIN

1.4 Revision History

1.5 Reference Documents

1.6 Document Structure

- **1 Introduction** This section introduces the Design Document. It explains the Purpose, the Scope and the framework of the document.

- **2 Architectural Design** This section is focused on the main components used for this system and the relationship between them, providing information about their deployment and how they operate. It also focuses on the architectural styles and the design patterns adopted for designing the system.

- **3 User Interface Design** This section provides an overview on how the User Interface will look like. In our case this section won't be detailed since all the product functions have been already represented in the RASD.

- **4 Requirements Traceability** This section explains how the requirements defined in the RASD map to the design elements defined in this document.

- **5 Implementation, Integration and Test Plan** In this section we identify the order in which we plan to implement the subcomponents of the system and the order in which we plan to integrate and test them.

2 ARCHITECTURAL DESIGN

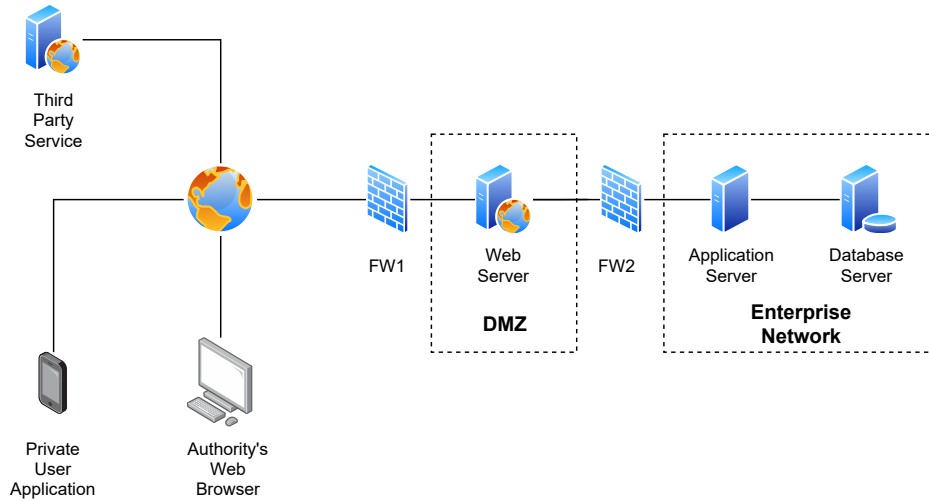## 2.1 High-level components and their interaction

Figure 1: High Level System Structure

Our system can be summarized into 3 logical layers:

- **Presentation Layer**
  This layer is divided into the Client tier (Mobile App for users) and the Web tier (Web App for authorities). From here, user, authorities and third parties, such as the municipality, can access the system's functionalities and the users' data through the Web Server, protected by a DeMilitarized Zone (DMZ)

- **Application Layer**
  In this layer the application server acts as a set of components accessible to the software developer through a standard API defined for the platform itself. We decided to separate the Application Server from the Web Server mainly for security reasons, and secondly because our application server targeta much more than just Web page generation: it implements services like clustering and load-balancing.

- **Data Layer**
  Here we have the Database Server and the Database itself, where all the data about the users, the authorities and the violations reported are stored

2.2 Component view

2.2.1 High Level Component Diagram

The following diagrams represents the systems' components and its interfaces throughout which they interact in order to execute their functionalities. We'll divide this view into two sides, the Client side and the Server side:

- The Client side is composed of three different components, the `Authority Web Application`, intended for authorities only, the `Third Party Web Application`, for municipality, and the `Mobile Application`, for users. The first two refers to the `AuthoritiesWebServices` service for authorities and to the `Third Party Web Services` for the municipality, the last to the `UserMobileServices` service for users

- The Server side is consequently also composed of three different components, the `Authorities Web Services` which will enable authorities to check the map, solve violations, etc., the `User Mobile Services` which gives users the chance to report street/parking violations with pictures, check/modify their personal data and review their report history and the `Third Party Services`, which is primarily intended for the municipality and enables them to receive and share data-mined information about violations to SafeStreets in order to generate suggested interventions etc.
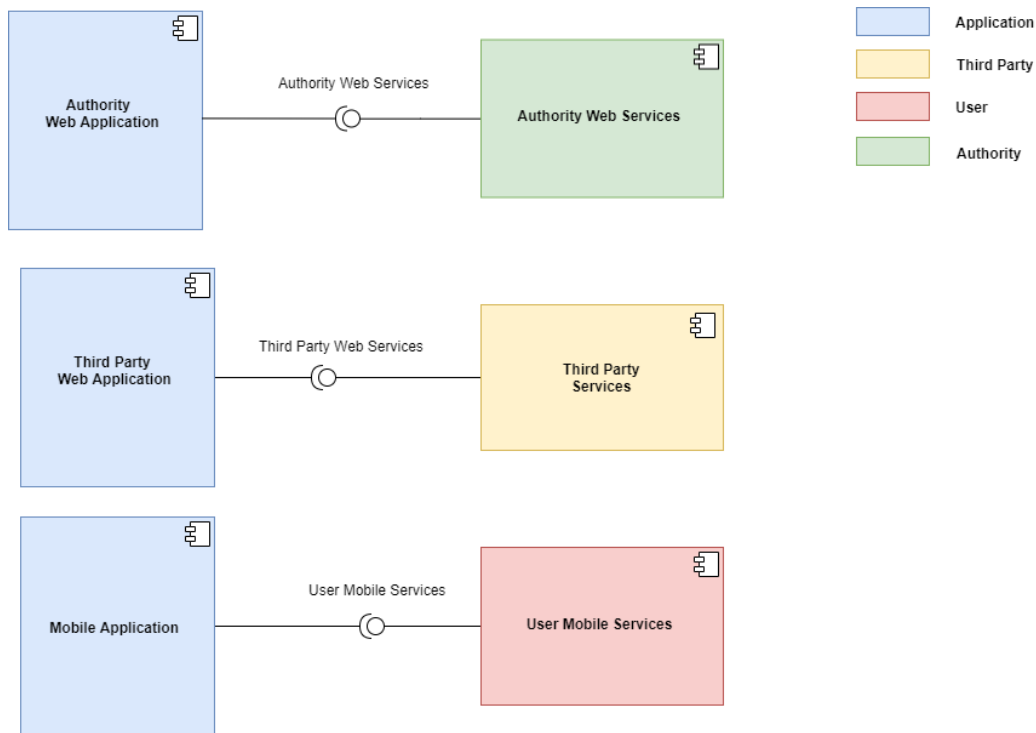
2.2.2 User Mobile Services Projection

The User Mobile Services Projection is subdivided into 4 modules: the
`Subscription Manager Module`, `Violation Reporting Module`, the `Reports
History Module` and the `Profile Manager Module`

Module Functionalities

- **Subscription Manager Module**: this module manages the registration and login processes of the user. It will allow them to insert their personal data and register to the system, and, subsequently, login with the same credentials.

- **Violation Reporting Module**: this module is very important as it enables the user to report street/parking violations by taking pictures of the vehicle involved and uploading them to the system

- **Reports History Module**: this module manages the users' report history, and, when requested, queries the data from the database and shows it to the user

- **Profile Manager Module**: this module provides all the necessary functions concerning the update and deletion of any existing profile information

2.2.3 Authority Web Services Projection

The Authority Web Services Projection is subdivided into 4 modules: the `Data Access Module`, the `Authentication Manager Module`, the `Violation Solving Module` and the `Suggestion Generating Module`

Module Functionalities

- **Data Access Module**: this module deals with the access to violations reported by users, but no private data is shared

- **Authentication Manager Module**: this module manages the registration and login of the authority. It will allow them to insert their personal data and to receive their AUC and DBP in order to successfully register to the system.

- **Violation Solving Module**: this module lets the authority interact with the list of violations, checking their validity and solving them.

- **Suggestion Generating Module**: this module manages the generation of suggestions on possible interventions to take in order to prevent violations from being committed in that precise area.

2.2.4 Third Party Services Projection

The Third Party Services Projection is subdivided into 2 modules: the `Data Sharing Module` and the `Authentication Manager Module`

Module Functionalities

- **Data Sharing Module**: this module is in charge of the sharing violation-related mined data to authorities in order to generate suggestions on street interventions to take

- **Authentication Manager Module**: this module manages the authentication procedure the municipality must execute to access the Web App

2.3 Deployment view

In order to deploy the system, we decided to implement a 4-tier architecture:

- **Tier 1** The first tier will be composed by the Client (we opted for a Thin Client) that includes a Mobile Application (for users) and a Web Application (for authorities).
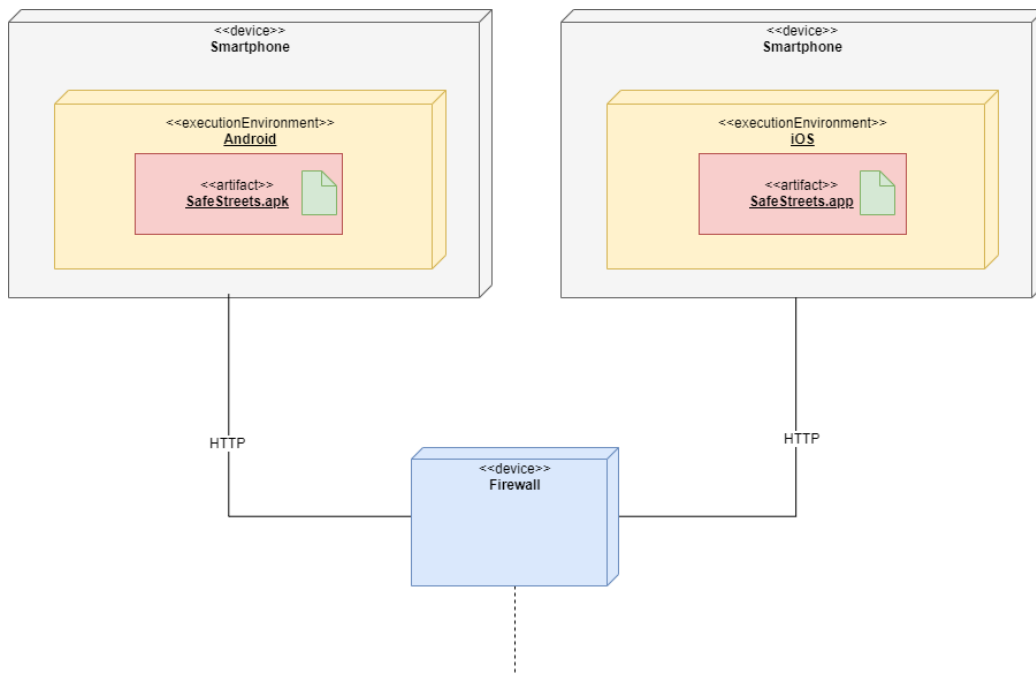

Figure 2: Tier 1 Deployment View

- **Tier 2** This tier corresponds to the Web Server, whose main focus is to store, process static content and deliver web pages to the Clients. Infact, we opted for Apache Tomcat thanks to its high-availability, clustering feature for load balancing and its optimal java-focused environment
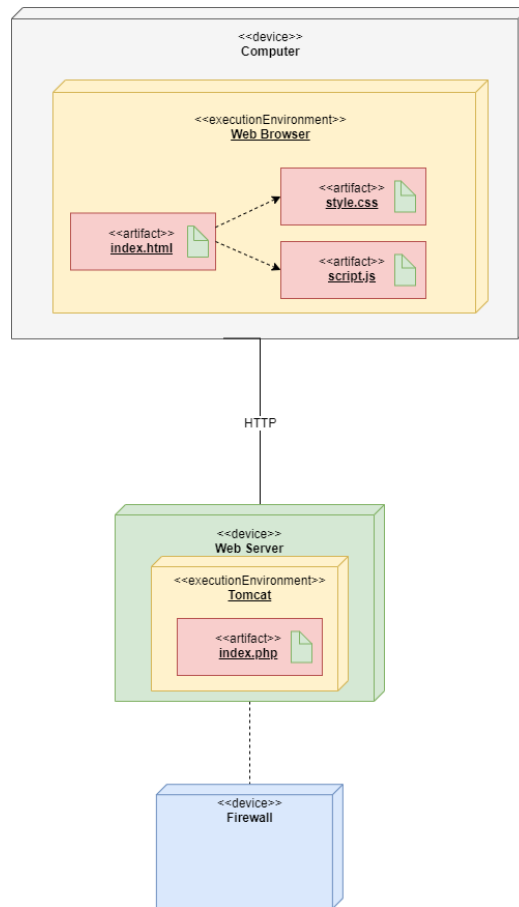

Figure 3: Tier 2 Deployment View

- **Tier 3** This tier corresponds to the Application Server, that both facilitates the creation of web applications and provides a server environment where to run these applications. We opted for Apache Geronimo, since one of its components is Tomcat, and because of its compatibility with Java EE 6 specifications.
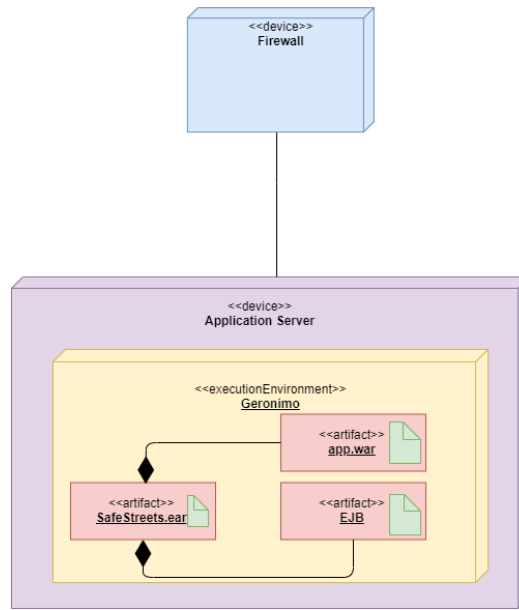

Figure 4: Tier 3 Deployment View

- **Tier 4** The final tier corresponds to the Database Server, where the DBMS is running. We opted for MySQL, since it is one of the most secure and reliable database management system used in popular web applications, while at the same time guarantees scalability and high performance.
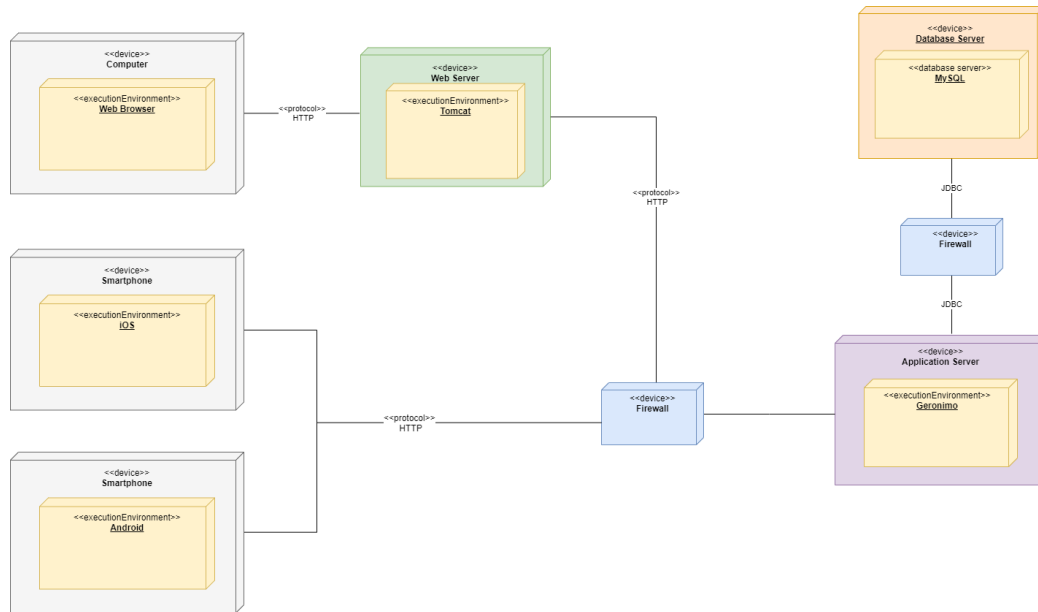

Figure 5: Entire System Deployment View with Tier 4

2.4 Runtime view: You can use sequence diagrams to describe the way components interact to accomplish specific tasks typically related to your use cases

2.5 Component interfaces

2.6 Selected architectural styles and patterns

2.6.1 Design Patterns

Here are the design patterns we used to make our architecture more flexible:

- **Model View Controller** Nowadays, most Mobile and Web Applications rely on this pattern. These type of applications infact retrieve data from the Database and updates the Users' Interface consequently and according to the provided input, while the Controller checks the

validity of the requested operations. Therefore the main scope of this pattern is to separate the Data (Model), the Users' Interface (View) and the user's input validator (Controller)

2.7 Other design decisions

3 USER INTERFACE DESIGN

As already mentioned in the `Requirements Analysis and Specification Document` (RASD), our idea is to provide users with a Mobile App (for smartphones and tablets) and authorities with a Web App (for computers).
The applications will be different both in functionalities and in style, and we decided to separate the two and therefore make them exclusive, because of the different use and different authentication method they have.
For a closer look at what the two interfaces look like, we invite you to check our RASD where several mock-ups have been presented

4 REQUIREMENTS TRACEABILITY: Explain how the requirements you have defined in the RASD map to the design elements that you have defined in this document.

5 IMPLEMENTATION, INTEGRATION AND TEST PLAN: Identify here the order in which you plan to implement the subcomponents of your system and the order in which you plan to integrate such subcomponents and test the integration.

6 EFFORT SPENT

7 REFERENCES