

SOFTWARE ENGINEERING II PROJECT

**SafeStreets**

***DD – D****esign* ***D****ocument*

*Professor*:

Matteo G. ROSSI

*Authors*:

Matteo POZZI

Sara SACCO

Andrea VENTURA

December 9, 2019

*version 1.0*

1. **Introduction**

**1.1 Purpose**

SafeStreets is a mobile application that relies on the help of lawful citizens to make life in the streets less stressful and more organized. As opposed to the RASD, the purpose of this document is to provide a description of the design of the application with enough completeness to allow the development process to proceed with an understanding of what needs to be built and how.

**1.2 Scope**

In this section, we refer to what has been previously stated in the RASD, providing a general overview of the scope of SafeStreets.

The intent is to create an application that gives people the ability to report and notify violations, e.g. vehicles parked in the middle of bike lanes, or in spots reserved to people with disabilities, to the designated authorities. In particular, citizens should be able to register as users by providing meaningful credentials, so as to avoid wasteful data such as fake accounts, and a way to verify them, e.g. ID or driver license. Once successfully logged in, users should be able to send pictures as proof of vehicles parked illegally and attach additional information to provide authorities with a starting point for the reviewing process, such as the date, the time, the type of violation which is to be reported and the place in which it has occurred, which can be retrieved through the geographical position of the user itself. This means the device which the user is working with should at least be equipped with a camera and a GPS system.

SafeStreets stores the information provided by its users and employs it by identifying and highlighting the zones which are found to be subject to the highest amount of violations, making them visible to both authorities and citizens.

Furthermore, SafeStreets wants to exploit its own data by combining it with information about accidents and analyzing it in order to identify zones or streets whose safety could be improved by making interventions, possibly suggesting viable solutions as well. This functionality is developed in collaboration with a third party, i.e. the municipality, meaning its usefulness will depend on the possibility of the municipality itself to share its data and match it with the interface SafeStreets developed for the functionality.

Lastly, SafeStreets strives to assist the local police in generating traffic tickets, and possibly build various statistics of interest. To ensure the effectiveness of this service, it is necessary that the exchange of sensible data which must occur between SafeStreets and the municipality cannot be tampered with in any way, e.g. modifying the picture of the violation at hand. To avoid this scenario, SafeStreets should only accept as reliable information pictures that have been taken within the application itself, meaning it should be equipped with an internal camera system.

In the following diagram (Figure 1.1), we define the boundaries of SafeStreets by identifying and distinguishing between World and Machine phenomena, with particular attention to the shared ones.



*Figure 1.1: World and Machine phenomena.*

**1.3 Definitions, acronyms, abbreviations**

- Definitions:

**User**: a general actor which is registered into the application; all users can consult statistics about violations and highlight unsafe areas;

**Authority**: a user which receives complaints and is able to identify actual violations among them. It has the power to punish the culprits with traffic tickets;

**Citizen**: a user which is not an authority, he can send reports about violations;

**Violation**: a violation of traffic laws, in particular parking violations;

**Accident**: a traffic event involving two or more vehicles where people got injured or caused damages to the vehicles

**Report**: a notification sent by a citizen to indicate violations, containing all the meaningful information about it;

**Traffic ticket**: a sanction which force an offender of a violation to pay an amount of money, can be generated by authorities;

**Unsafe area**: an area in which many violations and accidents have been reported;

**Statistics**: a collection of data about issued traffic tickets for each kind of violation occurred in a certain area.

**Suggested intervention:** a suggestion made by a system manager to be possibly applied in order to avoid future violations of a certain type.

- Acronyms:

**RASD**: Requirements Analysis and Specifications Document;

**DD**: Design Document;

- Abbreviations:

[**Rn**]: n-th requirement.

**1.4 Revision history**

* **Version 1.0 –**  December 9, 2019
  + First Release

**1.5 Reference documents**

**1.6 Document structure**

The document at hand is composed of 5 chapters, plus an appendix:

1. Introduction: it includes the goal of the project and an analysis of the world and shared phenomena;
2. Architectural design: here we provide a description of the components used in the application and their interactions;
3. User interface design: this section includes a general overview of how the user interfaces of the application will look like;
4. Requirements traceability: it provides an explanation of how the requirements defined in the RASD map to the design components described in this document;
5. Implementation, integration and test plan: here we identify the implementation plan, the integration plan and the test plan, specifying the order in which each component has to undergo each of the three steps;
6. Appendix: an accessory part that contains a quantitative description of the effort each member put into the completion of the document;
7. **Architectural design**
   1. **Overview**

Considering the fact that data storage plays a major part in the system to be designed, we opted for a three-tier architecture, which is defined as a client-server architecture in which functional logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. The choice was made to give the IT infrastructures more scalability and flexibility, and in particular lighten the burden server-side by distributing it on two different nodes. The three tiers involved are:

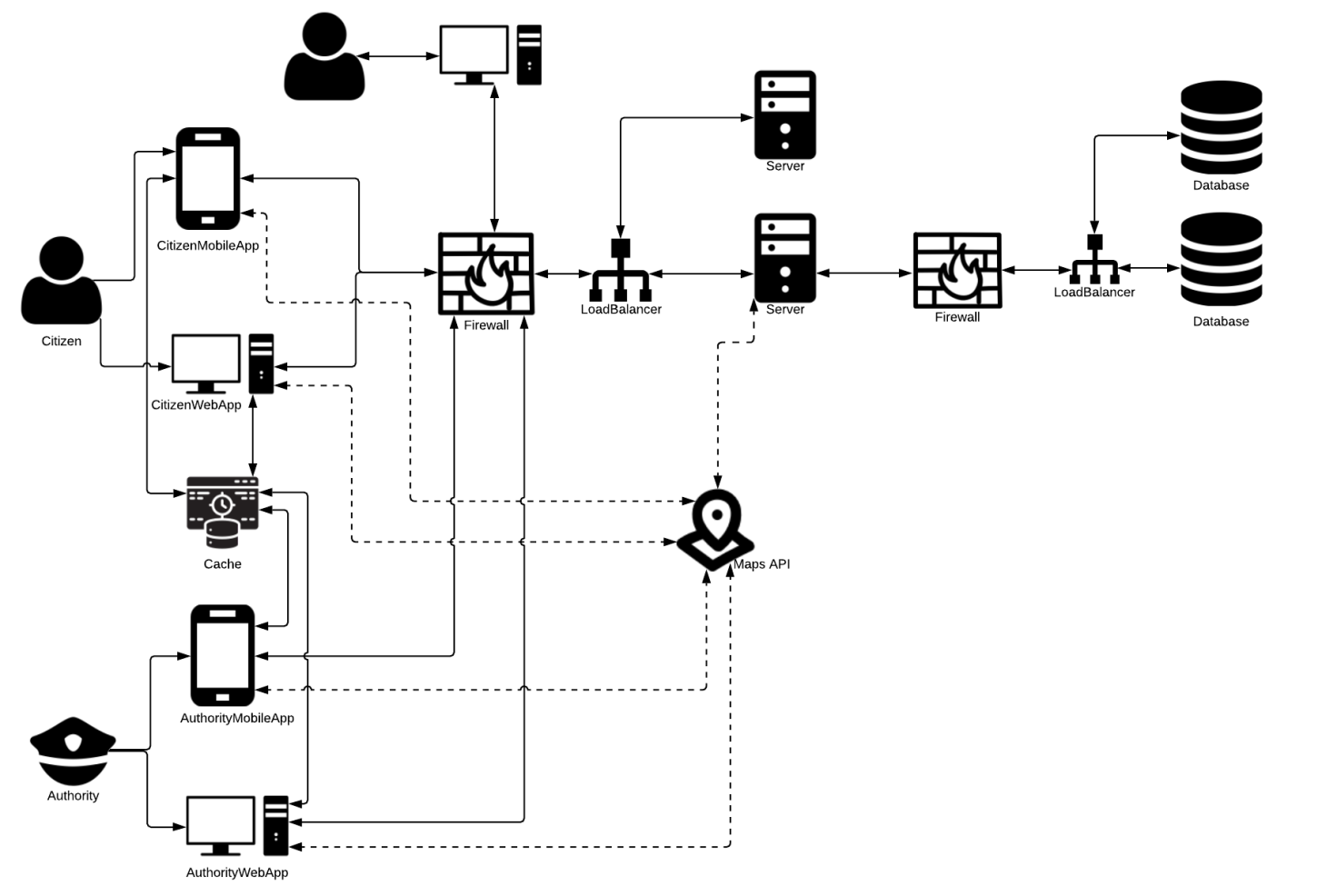
* Tier One: a light application layer and the presentation layer for the client;
* Tier Two: the application layer for the server;
* Tier Three: the data layer.

Each layer previously mentioned has the following characteristics:

* Presentation layer: it is the highest level of the application and it shows information related to the available services;
* Application layer: it controls the different functionalities of the system by performing detailed processing;
* Data layer: it stores data, keeping it independent from the logic of the application layer.



*Figure 2.1: Three-Tier Architecture.*

*Figure 2.2: System architecture.*

* 1. **Component view**
  2. **Deployment view**
  3. **Runtime view**
  4. **Component interfaces**
  5. **Selected architectural styles and patterns**
  6. **Other design decisions**

1. **User interface design**
2. **Requirements traceability**
3. **Implementation, integration and test plan**

The system is composed of different subsystem:

* CitizenMobileApp
* CitizenWebApp
* AuthorityMobileApp
* AuthorityWebApp
* PictureAnalyzer
* SafeStreetsServer
* External systems: DBMS, GoogleMaps, AdministratorDataEditing

Considering the architecture of SafeStreets system, for the integration testing it’s has been chosen the bottom-up strategy. In bottom-up approach, individual components are specified in detail and then they are connected to each other, until the realization of the complete system. This strategy allow us to start the integration and its testing without waiting for the complete implementation and the unit testing of each component in the system. In fact, bottom-up testing provides for components at lower hierarchy are tested individually and then the components that rely upon these components are also tested. Although top level components are most important, they are tested last using this strategy of integration testing.

1. **Appendix**