



**POLITECNICO**  
**MILANO 1863**

## **SAFESTREETS**

DESIGN DOCUMENT

*Sergio Cuzzucoli*  
*Daniele De Dominicis*

9 DECEMBER, 2019

---

<b>Deliverable:</b>	DD
<b>Title:</b>	Design Document
<b>Authors:</b>	Cuzzucoli Sergio , De Dominicis Daniele
<b>Version:</b>	1.0
<b>Date:</b>	9-December-2019
<b>Download page:</b>	<a href="https://github.com/Danielededo/CuzzucoliDeDominicis.git">https://github.com/Danielededo/CuzzucoliDeDominicis.git</a>
<b>Copyright:</b>	Copyright © 2019, Cuzzucoli Sergio, De Dominicis Daniele – All rights reserved

---

## Contents

<b>Table of Contents</b>	<b>3</b>
<b>List of Figures</b>	<b>4</b>
<b>1 Introduction</b>	<b>5</b>
1.1 Purpose	5
1.2 Scope	5
1.3 Definitions, Acronyms, Abbreviations	5
1.3.1 Definitions	5
1.3.2 Acronyms	5
1.3.3 Abbreviations	5
1.4 Revision history	5
1.5 Reference Documents	5
1.6 Document Structure	6
<b>2 Architectural Design</b>	<b>7</b>
2.1 Overview	7
2.2 Component view	8
2.3 Deployment view	10
2.4 Runtime view	11
2.4.1 User Access	11
2.4.2 Authority Access	11
2.4.3 User Files a report	12
2.4.4 Authority Checks a report	12
2.4.5 Authority/User reads list of violations	13
2.5 Component interfaces	14
2.6 Selected architectural styles and patterns	14
2.6.1 Three tier client-server	14
2.6.2 MVC	14
2.7 Other Design Decisions	14
2.7.1 Thin client	14
2.7.2 Data division	14
<b>3 User Interface Design</b>	<b>15</b>
<b>4 Requirements Traceability</b>	<b>16</b>
<b>5 Implementation, Integration and Test Plan</b>	<b>17</b>
<b>6 Effort Spent</b>	<b>18</b>
<b>References</b>	<b>19</b>

## List of Figures

1	High level overview of the system . . . . .	7
2	Component diagram . . . . .	8
3	Deployment diagram . . . . .	10
4	User access sequence diagram . . . . .	11
5	Authority access sequence diagram . . . . .	11
6	User files a report sequence diagram . . . . .	12
7	Authority checks a report sequence diagram . . . . .	12
8	Authority/User reads list of violations sequence diagram . . . . .	13

# 1 Introduction

## 1.1 Purpose

This document outlines the SafeStreets service, both basic and advanced functionalities, introduced in the corresponding RASD

## 1.2 Scope

SafeStreets is a service that basically allows users (k.a. normal citizens) to upload reports about violations. This reports are seen by authorities and investigated, if deemed appropriate, or dropped if not. Every user is also allowed to inspect data regarding violations w.r.t. the area interested by them, with limitations based on the type of user exploiting the service. In addition to the basic function, Safestreets can be used to acknowledge statistics about accidents in a fashion similar to that of the violations.

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

- **Client:** Piece of software or hardware that can access services offered by a server in different forms.
- **Server:** Piece of software or hardware that offers different services (that can constitute a part or the entirety of an application) to one or more clients.

### 1.3.2 Acronyms

Acronym	Meaning
DB	Data Base
DBMS	Data Base Management System
DD	Design Document
API	Application Program Interface
UI	User Interface
UX	User Experience
OS	Operating System
RASD	Requirement Analysis and Specification Document
GPS	Global Positioning System
CNN	Convolutional Neural Network
OTP	One Time Password

### 1.3.3 Abbreviations

**G.th** : n-th Goal

**D.th** : n-th Domain Assumption

**R.th** : n-th Functional Requirement

## 1.4 Revision history

## 1.5 Reference Documents

- Project assignment specifications:[1]
- UML: [2]

- DD to be analyzed: [3]

## 1.6 Document Structure

- **Introduction:** summary of the concepts already expressed in the RASD document.
- **Architectural Design:** detailed description of the architectural design w.r.t components and design patterns.
- **User Interface Design:** addition details on the UI previously sketched in the RASD document by means of UX modeling.
- **Requirements Traceability:** analysis on the requirements of the RASD and how they are satisfied by the design choices of the DD.
- **Implementation, Integration and Test plan:** showing implementation and integration of subcomponents in the defined order and giving details on the subsequential testing for the integration.

## 2 Architectural Design

### 2.1 Overview

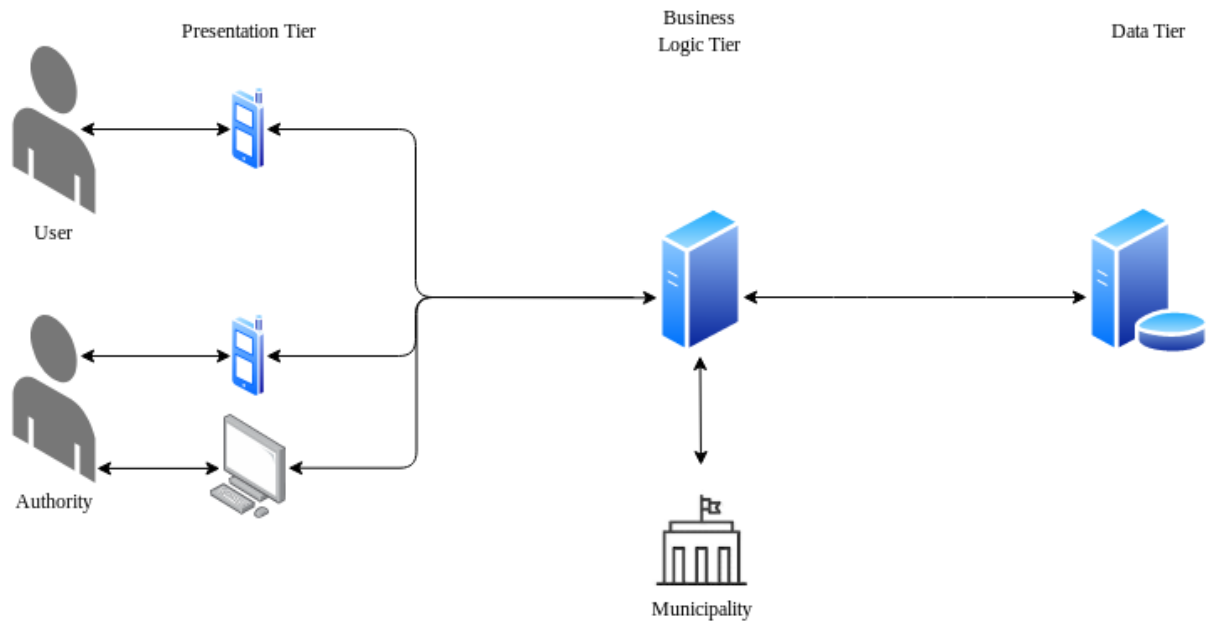


Figure 1: High level overview of the system

## 2.2 Component view

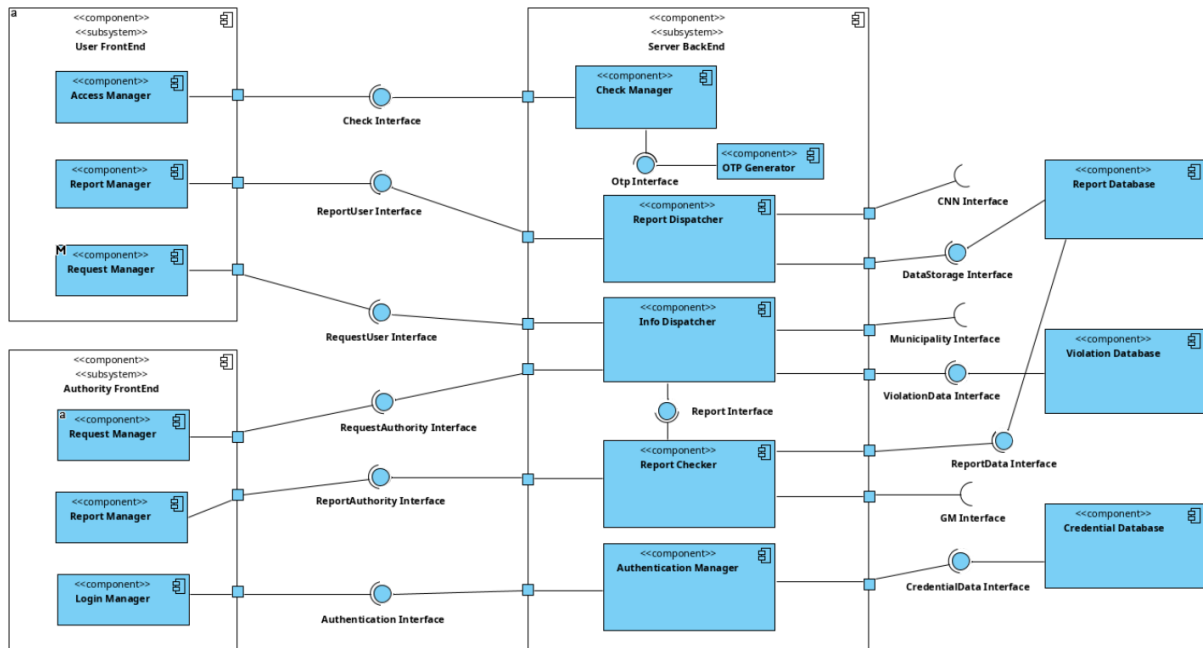


Figure 2: Component diagram

The UML Component diagram shows the internal modular structure of the components and their connections. To communicate, they can require a variable number of interfaces and attributes. A brief description of each component, w.r.t the most important interfaces, is presented:

- Clients: Citizens' and authorities' machines, they can be either mobile devices or computers.
  - Citizen Frontend
    - \* Report Manager: Establishes an exchange of messages with the Server and sends data regarding the alleged violation, to be later examined.
    - \* Request Manager: Retrieves the list of violations or accidents using the Server.
  - Authority Frontend
    - \* Report Manager: Retrieves a report and orders to approve it or to discard it, based on the information stored.
    - \* Request Manager: Retrieves the list of violations or accidents using the Server.
- Server Backend: Interacts with the clients and the Databases and contains the logic of the Application.
  - Check Manager: Requests and receives the OTP by OTPGenerator to withstand the login of the citizen
  - Report Dispatcher: Receives the information about the reports filed by citizens, completes them with data obtained by the CNN API (a.k.a the recognized plates) and pushes them to the Report Server if the report is correctly fulfilled, aborting the incorrect ones.
  - Authentication Manager: Exchanges information with the Database containing the credentials of the authority that wants to login and ultimates the login process



- Info Dispatcher: Communicates with both the types of users to retrieve information about violations or accidents (these are retrieved by the Municipality API). *Note: different kinds of users receive different quantities of information, for example, citizens do not obtain the license plates of the offenders.*
- Report Checker: Sends information about the open reports to AuthorityFrontend::ReportManager, locates the position using GM API and receives the approval of dismissal of said reports to store them as Violations on the Violation Database or not, by sending them to InfoDispatcher.
- Databases: They allow the retrieval and addition of data.

## 2.3 Deployment view

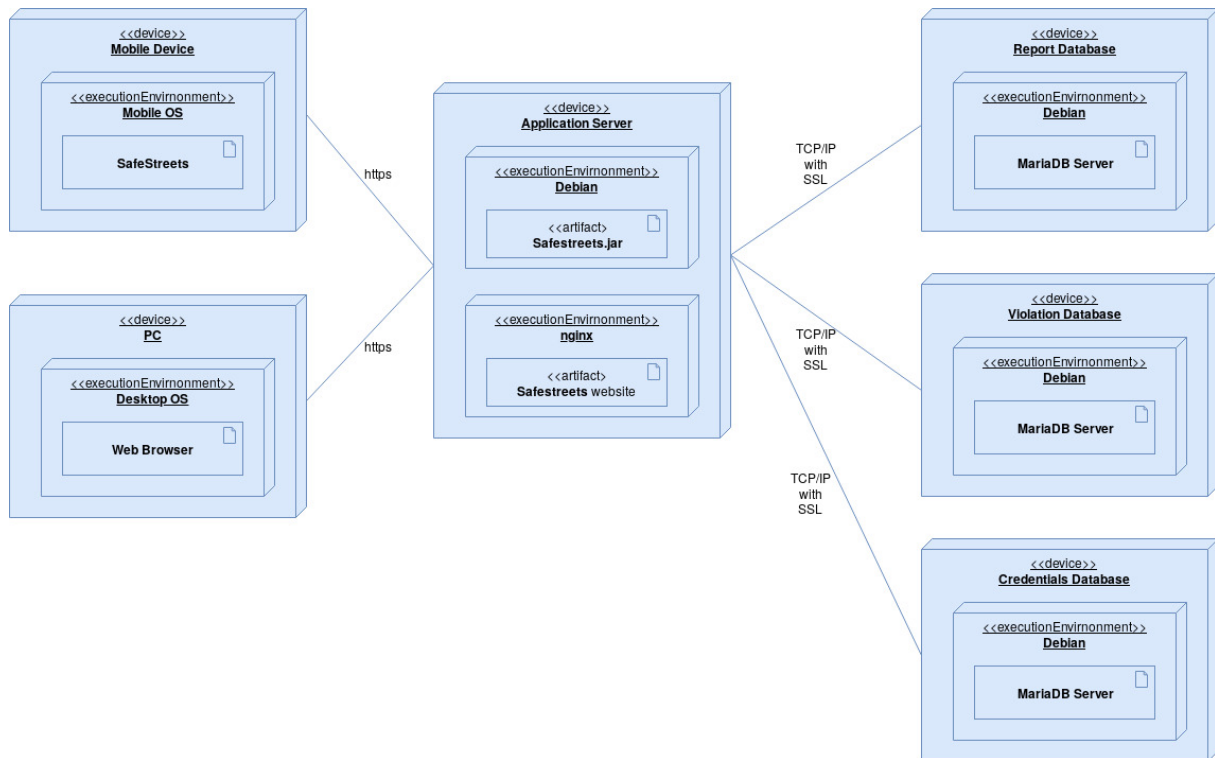


Figure 3: Deployment diagram

The purpose of the Deployment diagram is to specify the distribution of the components of the system, showing the configuration of run time processing nodes and components that live on them.

## 2.4 Runtime view

### 2.4.1 User Access

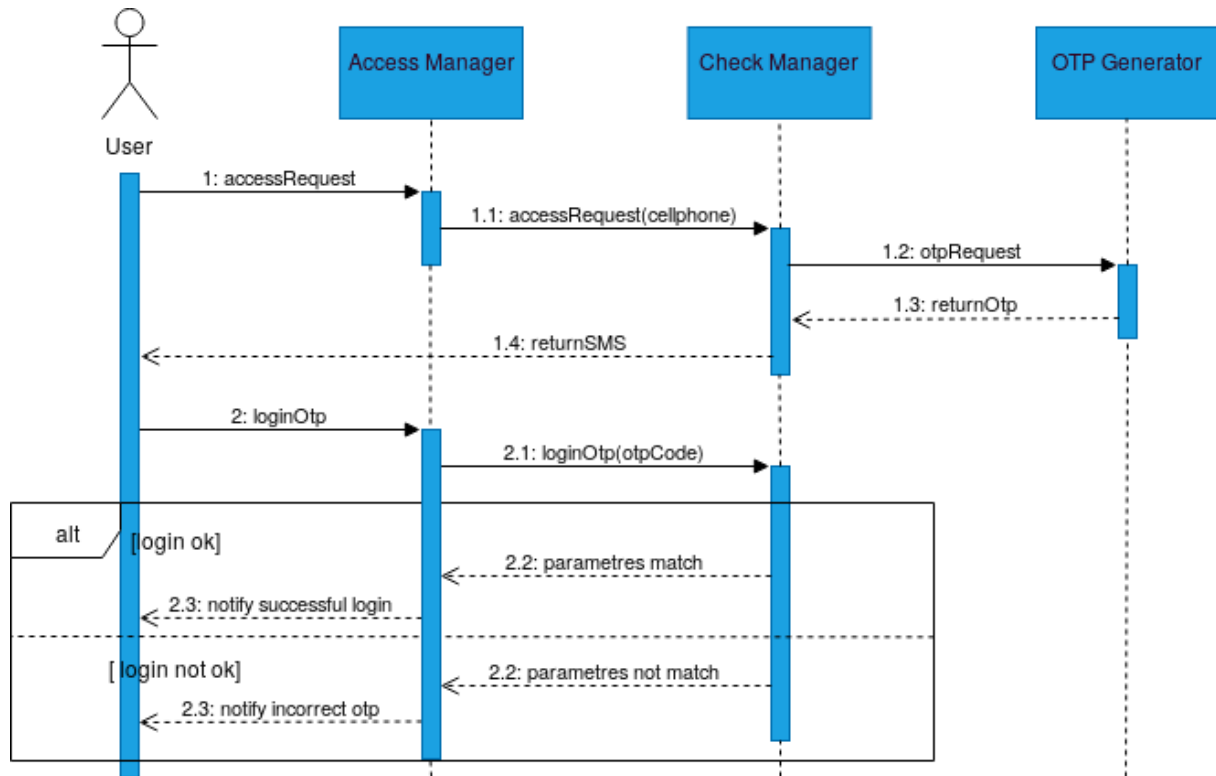


Figure 4: User access sequence diagram

### 2.4.2 Authority Access

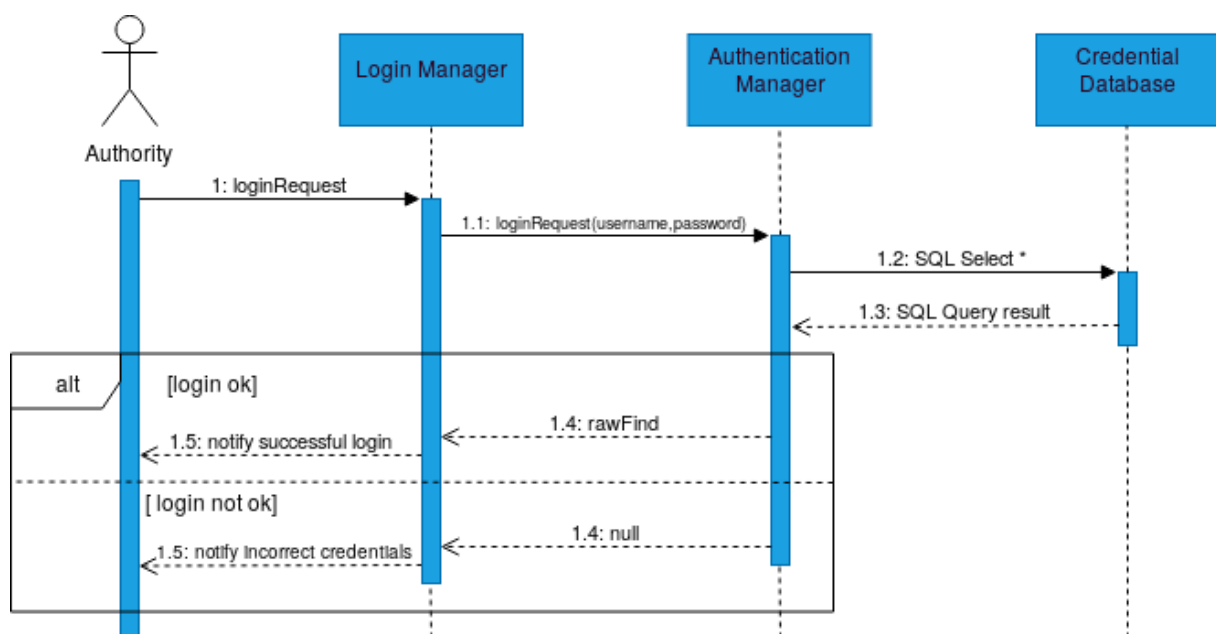


Figure 5: Authority access sequence diagram

### 2.4.3 User Files a report

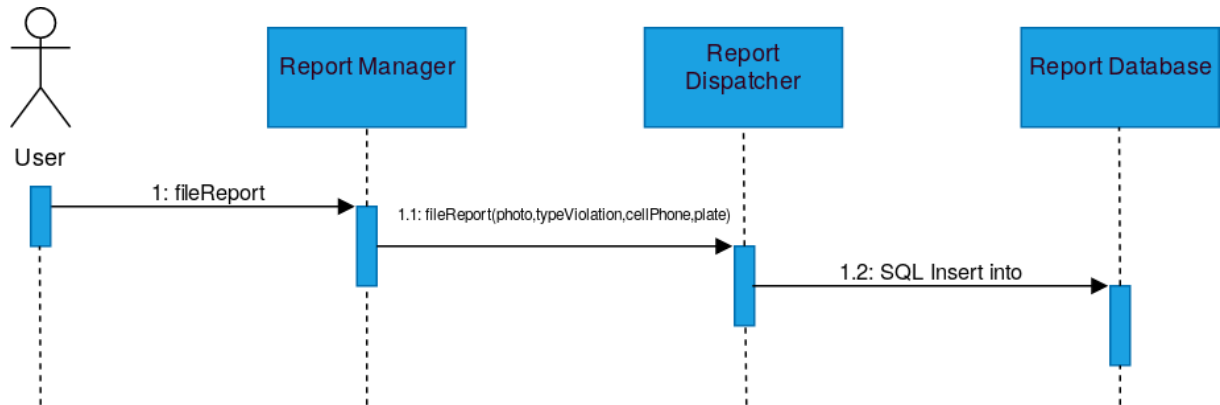


Figure 6: User files a report sequence diagram

### 2.4.4 Authority Checks a report

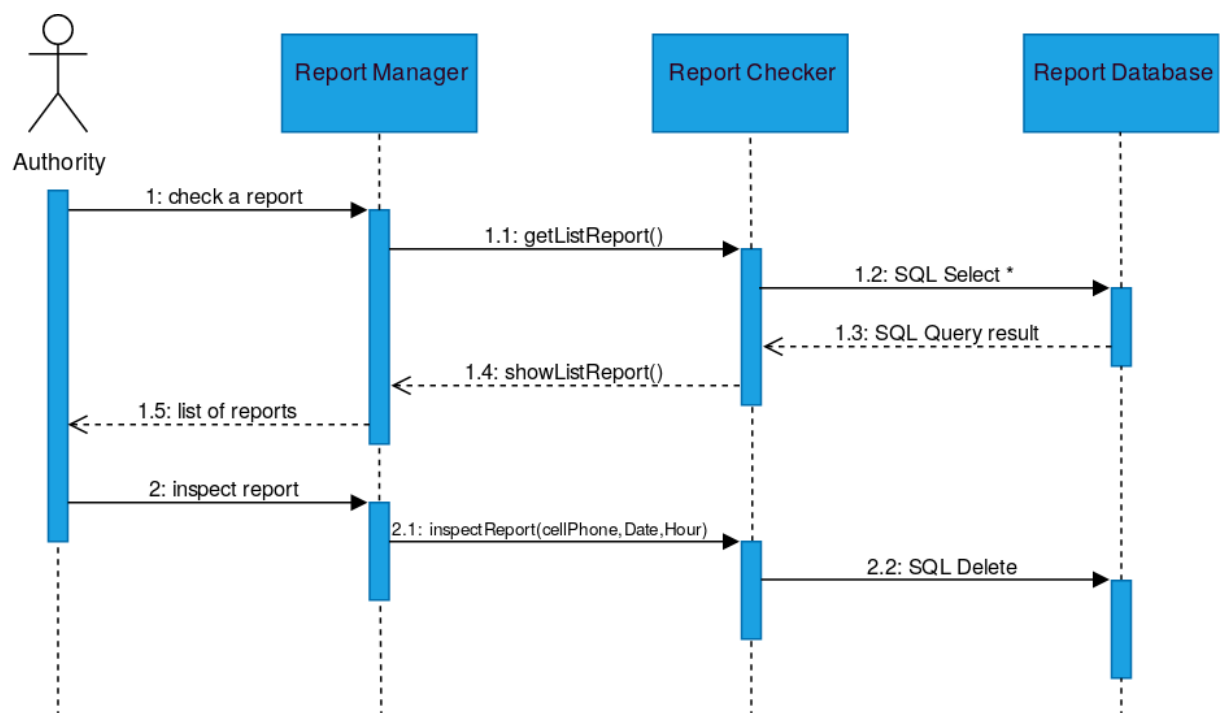


Figure 7: Authority checks a report sequence diagram

## 2.4.5 Authority/User reads list of violations

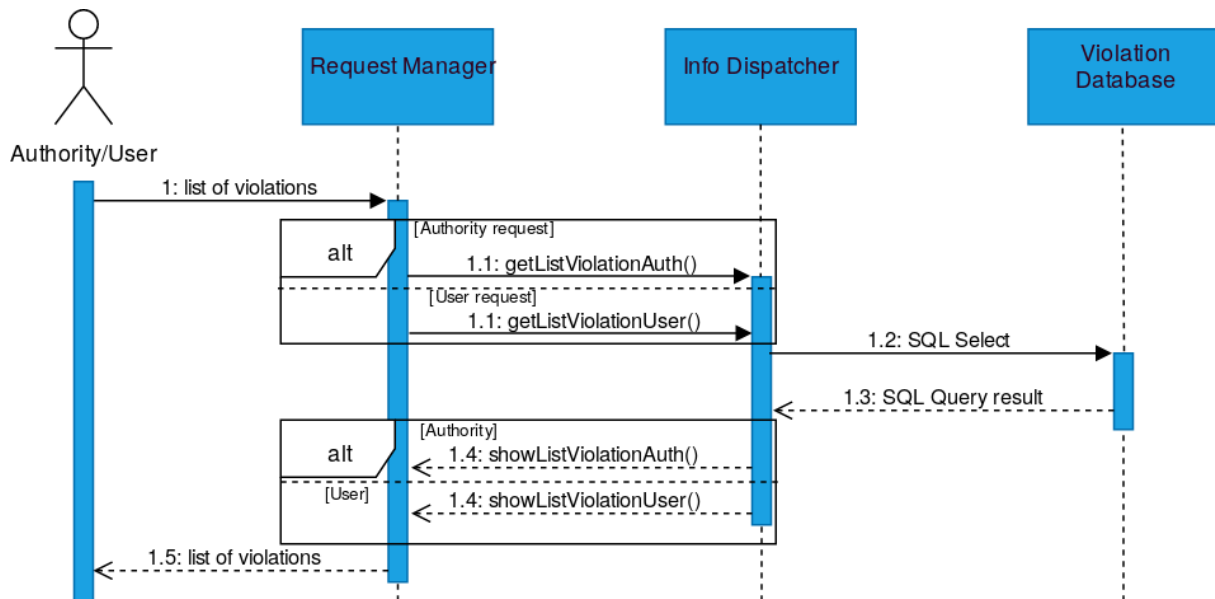


Figure 8: Authority/User reads list of violations sequence diagram

## 2.5 Component interfaces

## 2.6 Selected architectural styles and patterns

### 2.6.1 Three tier client-server

The architectural style chosen for Safestreets is the **client-server** in its **three-tier** variant. It allows the separation of the access to data, presentation and logic of the application. The main advantage of a multilayered architecture is the decoupling of Logic and Data. This leads to a high level of Maintainability and Scalability, qualities that are extremely useful when working with cities that tend to expand their territory and consequently the number of streets to be controlled. At the same time the system is safe as the access to data is transparent to the end user.

### 2.6.2 MVC

MVC is a design pattern that breaks an application into different layers of functionalities: Model, View and Controller. This allows for the internal representation of information to differ from the information presented to the user. This produces easily reusable and flexible code and allows parallel development. The division is expressed here:

- **View:** Clients.
- **Controller:** Application.
- **Model:** DBs.

## 2.7 Other Design Decisions

### 2.7.1 Thin client

In SafeStreets the client contains no logic of the Application at all, but it is just able to format and send reports and to request lists. This guarantees the App to be "light" on resources and power consumption (the citizens do not need to have a high end smartphone) while keeping it easily upgradable.

### 2.7.2 Data division

Data is spread among three different DBs as different types of data requires different levels of security and replication:

- **Report Server:** The data saved on this server is not important as reports are very dependent on time. The images and information stored are not legally binding (only authorities can administer fines), thus it must be able to handle large quantities of data and be operative in a short span of time but data loss can be sustained.
- **Violation Server:** The data stored here is composed of light elements (only text) which do not require particular encryption thus the server is able to sustain a great deal of ACID operations but requires data duplication.
- **Credentials Server:** This server contains information for the Authentication of the authorities so, not only does it require data duplication to be quickly operative also in case of failure, but encryption is also crucial to this device. Data is stored following good practices of security (e.g. only the salted hashes of the password are saved).

### **3 User Interface Design**

## **4 Requirements Traceability**



## **5 Implementation, Integration and Test Plan**

## 6 Effort Spent

Summary of the work dedicated to the project by the individuals composing the group

*Cuzzucoli Sergio*

Date	Task	Hours
25/11/2019	Introduction	2
26/11/2019	First steps and discussion on Architectural Design	3
27/11/2019	Additions to Architectural Design	5
28/11/2019	Explication of design choices	4

*De Dominicis Daniele*

Date	Task	Hours
25/11/2019	Introduction	1
26/11/2019	First steps and discussion on Architectural Design	4
27/11/2019	Additions to Architectural Design	3
28/11/2019	Addition of runtime views	4

## References

- [1] Di Nitto Elisabetta. Mandatory project: goal, schedule, and rules, 2019.
- [2] The Object Management Group (OMG). Unified modelling language: Infrastructure, 2011. version 2.4.1. omg document: formal/2011-08-05. Technical report, , 2011.
- [3] Unknown Students. Dd–design document, 2018.