POLITECNICO
MILANO 1863

SafeStreets

Design Document

Davide Cocco - 944122
Marco Gasperini - 944922

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

Contents

1 INTRODUCTION

1.1 Purpose

This `Design Document (DD)` for the SafeStreets software will provide a functional description of the system by describing its architecture. It will eventually by used by the development team as a blueprint to guide the engineering of the application.

1.2 Scope

The main objective of the S2B will be assisting (thus not substituting) authorities and officers in handling traffic violations though a crowd-sourced platform in which civilians can participate. A mobile application will allow users, both civilians and officers, to report violations through the use of the camera and the GPS, sending the data to authorities who will process such reports. Law enforcement will be aided by a data mining system which will produce relevant data about the registered violations, and the S2B will be able to automatically compile a ticket and send it to the municipality's VLA as soon as an officer personally convalidates a violation.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- `Violation | Offence:` we will strictly refer to any kind of static traffic violation, especially parking violations.
- `Authority:` a law enforcement authority which is manages traffic violations, it could be the local police, the traffic wardens etc. We will refer with this term also to the personnel which operates the web application in the authorities' headquarters.
- `User:` refers to the users of the mobile application, that is officers and civilians.

1.3.2 Acronyms

- `S2B:` software to be.
- `HFVZ:` high violation frequency zone.
- `GPS:` Global Positioning System.

- `GDPR`: General Data Protection Regulation.
- `DW`: data warehouse.
- `VLA`: Vehicle Licensing Authority.
- `RACS`: Reliable Array of Cloned Services.
- `RAPS`: Reliable Array of Partitioned Services.
- `GUI`: Graphical User Interface.
- `DMZ`: Demilitarized zone.
- `ADS`: Application and Data server of the authorities.
- `API`: Application Programming Interface.

1.3.3 Abbreviations

- [Gn]: n-goal.
- [Rn]: n-requirment.
- [Dn]: n-domain assumption
- App: application.

1.4 Revision History

1.5 Reference Documents

1.6 Document Structure

- **Introduction**: includes the purpose and scope of the document along with some relevant definitions and acronyms used throughout the document.
- **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.

- **User interface design**: this section provides an overview on how the User Interface will look like, but it will be omitted due to the presence of this chapter in the RASD.
- **Requirements traceability**: in this sections the requirements highlighted in the RASD document will be associated with the design elements explained in this document.
- **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 overview

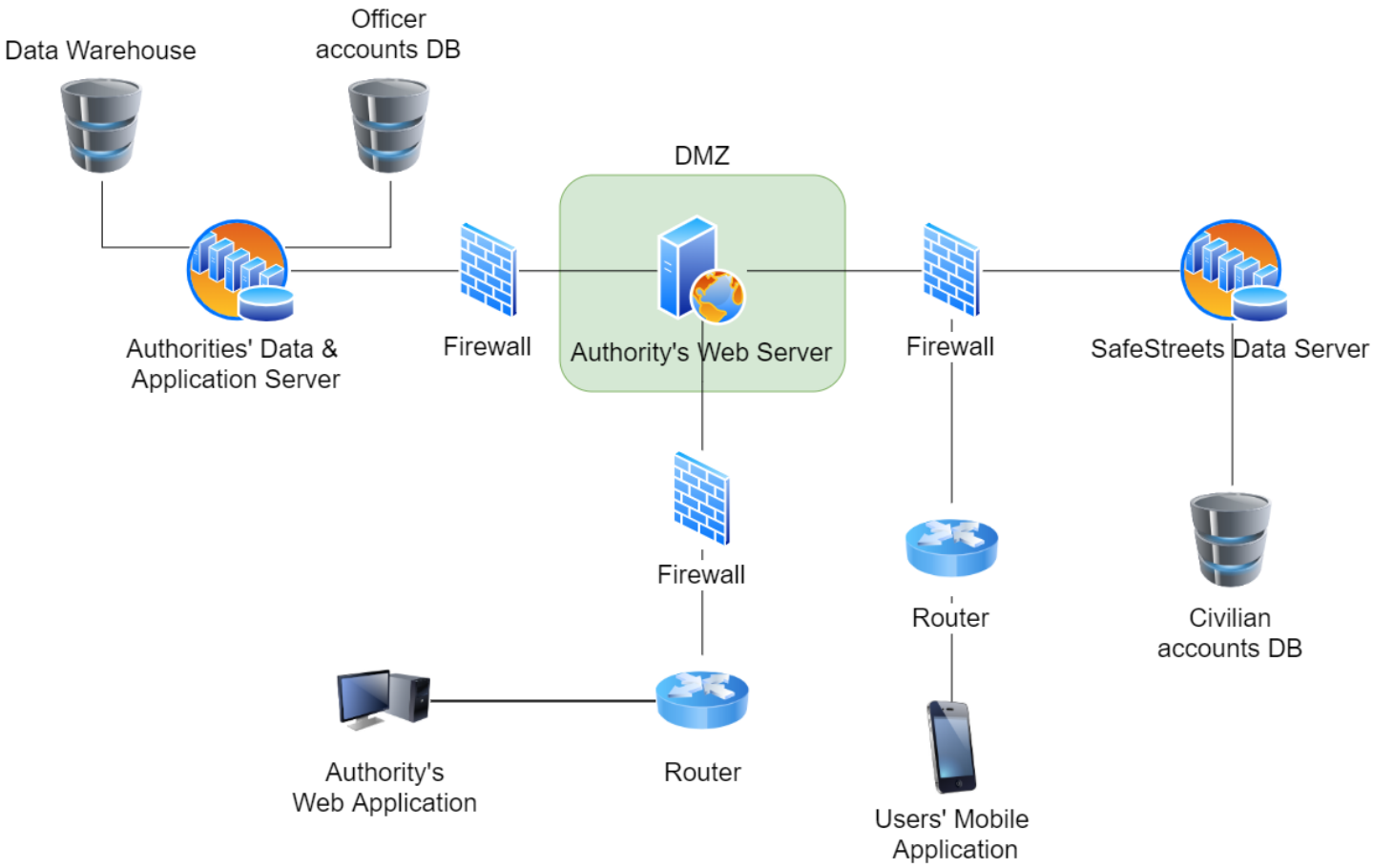


Figure 1: High Level Structure with a single Authority and a single User

The three logical layers of the S2B are distributed in a **three-tier architecture**:

- **Presentation layer:** two hard clients:
  - the GUI at the authority's web application is the tool thanks to which the authority's personnel can add officers, receive reports from the users through HTTP calls thanks to a REST architecture embedded in the web server, and validate or invalidate them. It can thus interact directly with the web server, and rarely with the SafeStreets' server to which it sends ban requests. The web server

is isolated from the outside by firewalls for increased security and to protect the ADS even if the web server is compromised. The only logic that will be implemented directly at the this client will be the supervised algorithm to recognize plates: this way we can reduce latency, because since the overhead to effectively send a correct plate frame will be pretty heavy, we remove the step to contact the application server and instead execute this function directly at the authorities' location. Anything else will be handled as notifications forwarded to the ADS to limit the computational power used at the authorities' locations.

  - the GUI at the mobile application, which can be used to file reports and send them to the Authority's web server by all users, and which allows officers to check reports and sign automated tickets (and also receives slightly different updates which include sensitive data to be made unreadable by civilians).

  Moreover, both clients must be able to "unpack" the received updates' representations into a format understandable to the user.

- **Application layer:** the hypermedia at the mobile application moves from a state to another thanks to the updates received by the authorities' web servers (which are themselves sent to the authorities by the ADS, and then are periodically spread to the clients in the network). While the web server constitutes an interface protected by a DMZ (to isolate it from the ADS) which clients are forced to invoke when applicative functions are needed, the ADS hosts the update producing and ticket emission functionalities. It is important to denote the fact that since every Authority owns a web server, calls received from the mobile units can be handled by any of the authorities for load balancing. Thus we can say that both the Web server located at the authorities' location and the applicative functions of the ADS constitute this layer.

- **Data layer:** three instances where data access is needed:

  - for managing the civilian accounts and bans, at the SafeStreets server;
  - for managing the officer accounts, at the ADS;
  - for reading and writing historical data about reports in the data warehouse, again at the ADS; Therefore we can say that application and data services are "partially merged" as the ADS hosts both an application and a data server.

It is important to notice how the ADS could be implemented in a `RAPS architecture` where multiple ADS each own a single microservice, for example a server could handle ticket emission, another one officer accounts, and another one update mining and spreading. Another configuration could be assigning an ADS for each authority. Web Servers are installed at the authority's location, one for each authority, to better serve nearby mobile units and balance loads, especially in big cities. The SafeStreet servers could be arranged in a `RACS architecture` to handle requests from great numbers of civilians. In some settings such as small towns a lightweight version of the architecture could be employed by running the web server directly in the same machine where the Authority's client is running instead of a standalone device, thus making it a two-tier architecture since the web server wouldn't be separed from the presentation layer anymore.

2.2 Component view

2.2.1 High Level Component Diagram

The following diagrams illustrate the system components and the interfaces through which they interact to fullfil their functionalities. The diagram is focused on the application tier, so the remaining tiers (the presentation tier and the data tier) are shown as black-box. First we highlight a distinction between Client side and Server side:

- The Client side is composed by two components, *Web Application* and *Mobile Application*. The first is referred to the Authorities client, latter to the User client (Civilian and Officer).
- The Server side is composed of three main components: *Authorities Web Services* manages the authorities client, through the *AuthoritiesWebServices* interface server-side; the other two services are the *Civilian Mobile Services* and the *Officer Mobile Services* that manage the *Mobile Application* from two distinct interfaces.

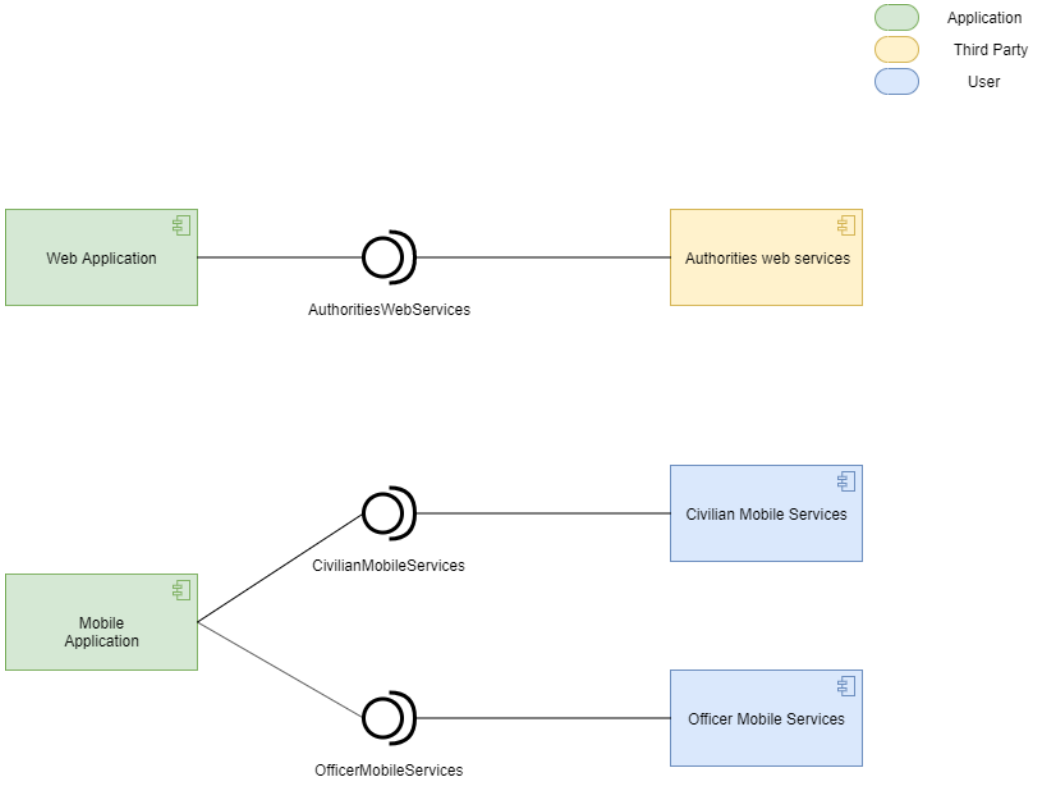
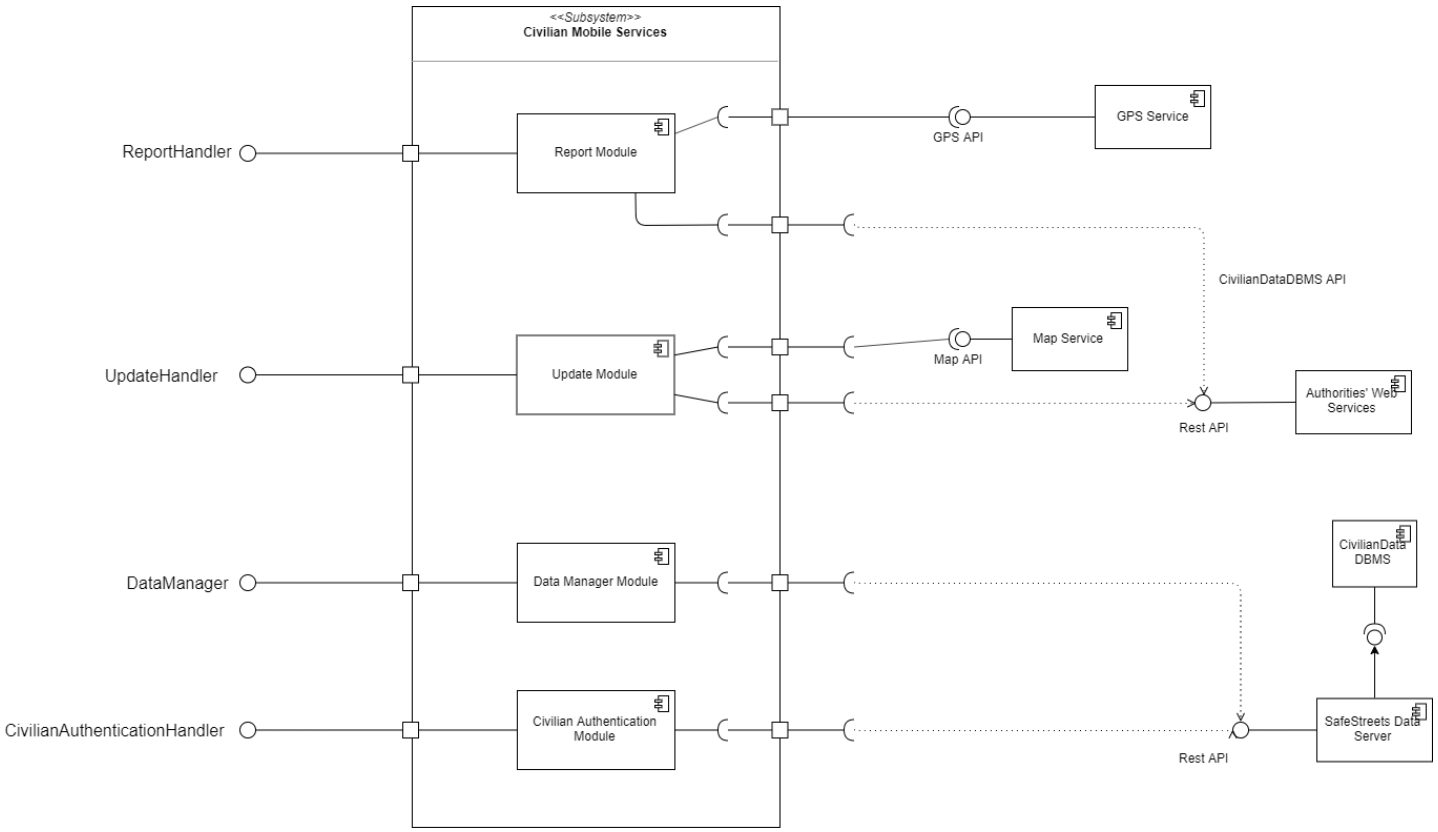


Figure 2: Component diagram

2.2.2 Civilian Mobile Services Projection

Civilian Mobile Services subsystem is made of four components: Report Module, Update Module, Data Manage Module and Civilian Authentication Module. These components provide the Mobile Application the following interfaces: ReportHandler, UpdateHandler, DataManager and AuthenticationHandler. The Modules also communicate with the map, gps services and two DBMS; the first one allows to collect the mined data about reports in the data warehouse, the latter manages user accesses of any form.



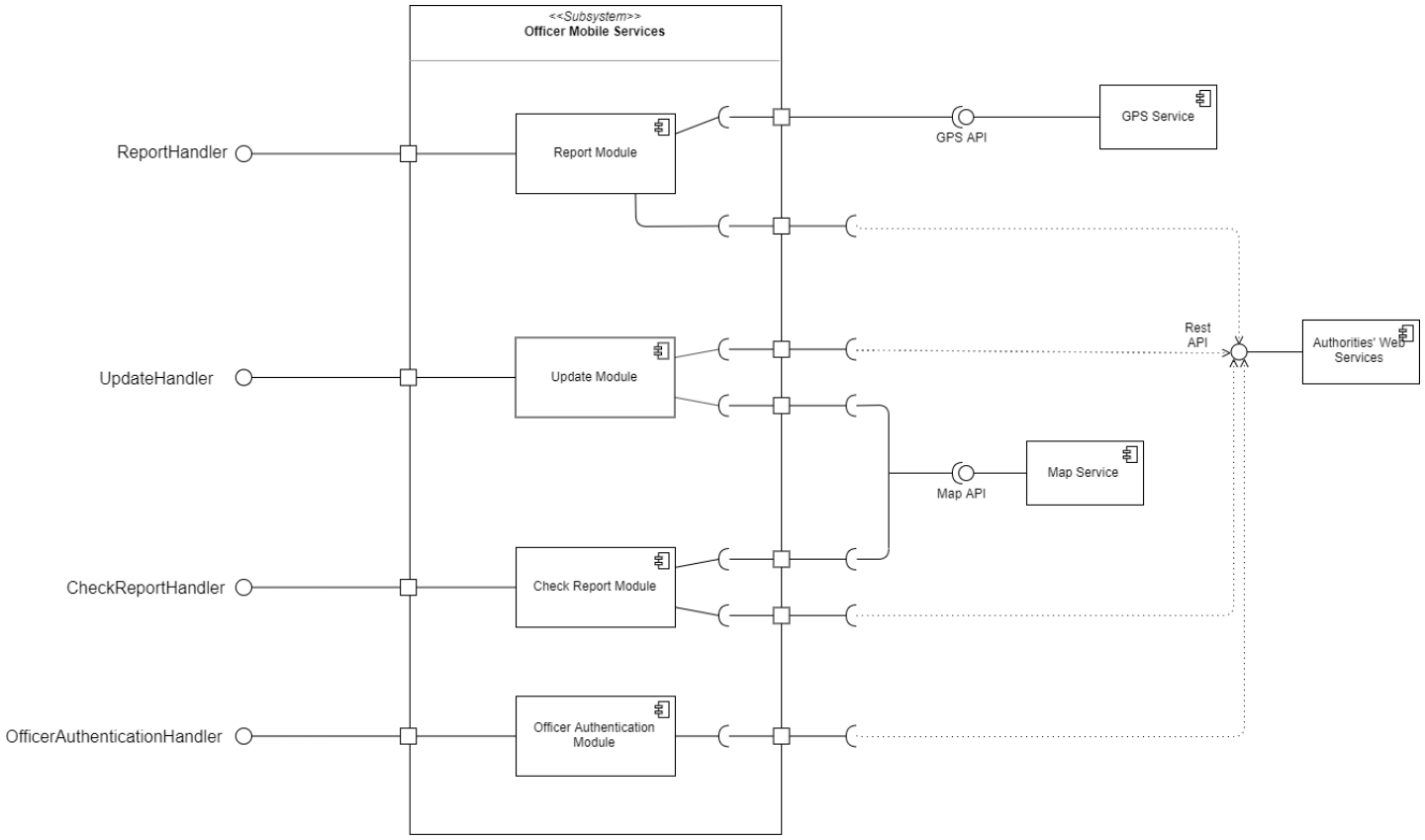
Module Functionalities

- `Report Module`: this module manages the reports done by Civilians and their sending to the Authority's Web server. It will allow the user to take photos of the violation, enter a description of it, timestamp it and attach geolocation coordinates thanks to the GPS API, and also handle all the communication needed for the plate recognition algorithm to work.

- `Update Module`: this module handles the receiving and visualizing of the mined data about HFVZs, through the Map API.
- `Data Manager Module`: this module allows civilians to manage their personal data (for instance changing their e-mail address or password).
- `Civilian Authentication Module`: this module handles the registration and login requests by civilians, and thus providing the effects of an eventual ban.

2.2.3 Officer Mobile Services Projection

Officer Mobile Services subsystem is an enriched version of the civilian one to accomodate the advanced functions accessible to officers. It is made of four components: Report Module, Update Module, Check Report Module and Officer Authentication Module. The Report Module works in the same way as the civilian mobile services, while the Update Module possesses quite a few different features to assist the officer. The Check Report Module manages the practice of checking violations by the officer which might eventually result in a ticket being compiled. The Officer Authentication Module communicates with the Authorities Web Server and not with the SafeStreets server because officer data is kept in the ADS.



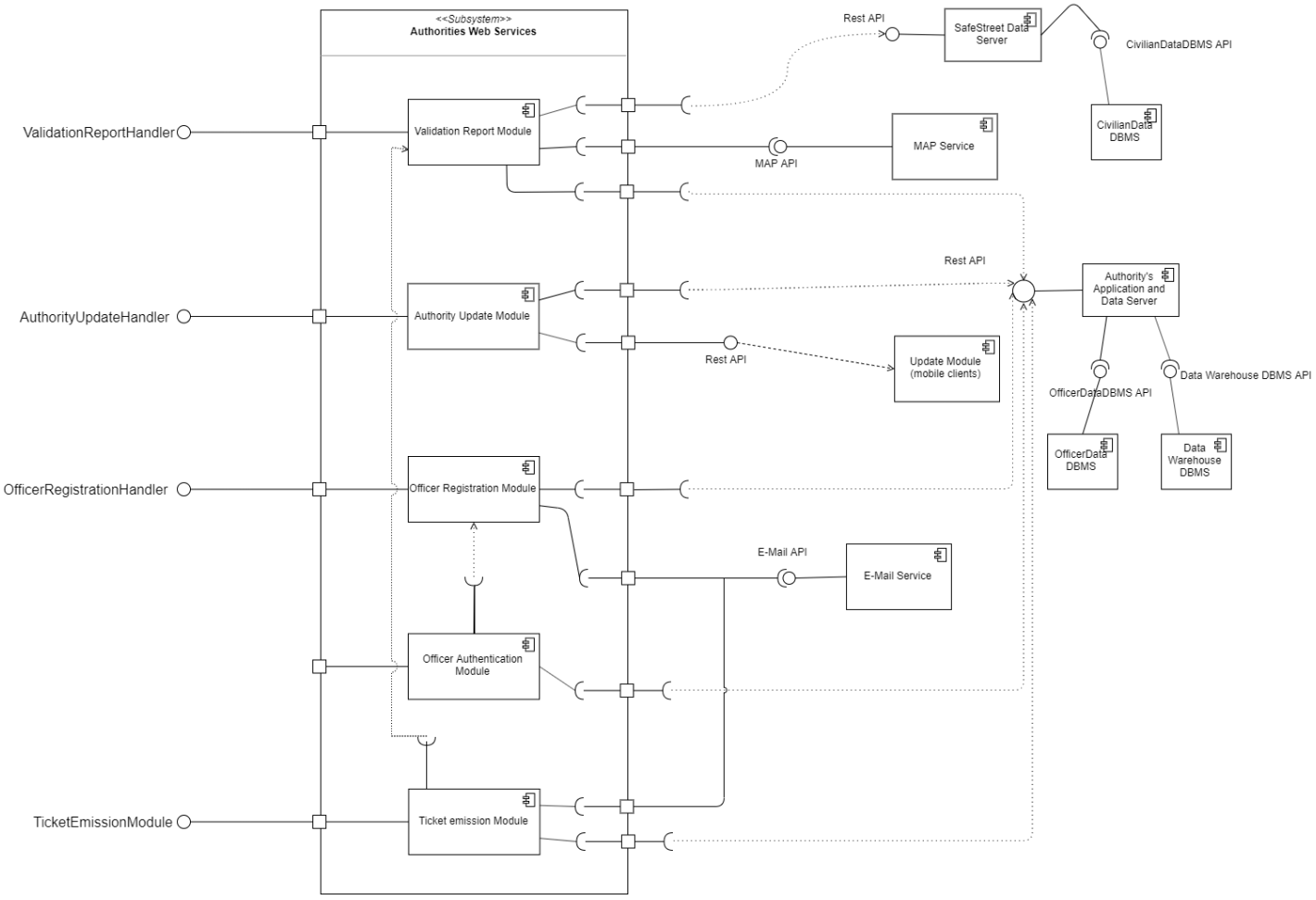
Module Functionalities Pretty much the same as the Civilian Mobile Services:

- `Report Module`: works in the same way as the Civilian's one, as officers will also be able to report violations.

- `Update Module`: just as the Civilian's one, but enriched with more updates that should only be accessible to law enforcement agents, such as the most egregious offenders list, newly added reports to the map, etc.
- `Check Report Module`: this module allows the officers to confirm the presence of a violation (and the correctness of the plate shown in the report) and enter a secret PIN to allow the emission of a ticket.
- `Officer Authentication Module`: this module handles the registration and login requests by officers.

2.2.4 Authorities Web Services Projection

The Authorities Web Services are carried out by the Web Server and the ADS, which implements most functions of the application. Five Modules will be implemented to fulfill every functionality: Validation Report, Authority Update, Officer Registration, Officer Authentication, Ticket Emission. Web Server and ADS are strongly intertwined to provide the functionalities which all the three actors' clients need: the web server is in fact an interface which provides a layer for communication for the clients to interact with the ADS, for enhanced security, as every HTTP call which eventually has to reach the ADS to execute some logic has to pass through the web server at some authority's location.



Module Functionalities

- `Validation Report Module`: this module handles the functionality

for validating reports by the authority's personnel and the subsequent adding to the DW. Also the issuing of bans which is performable while invalidating reports.

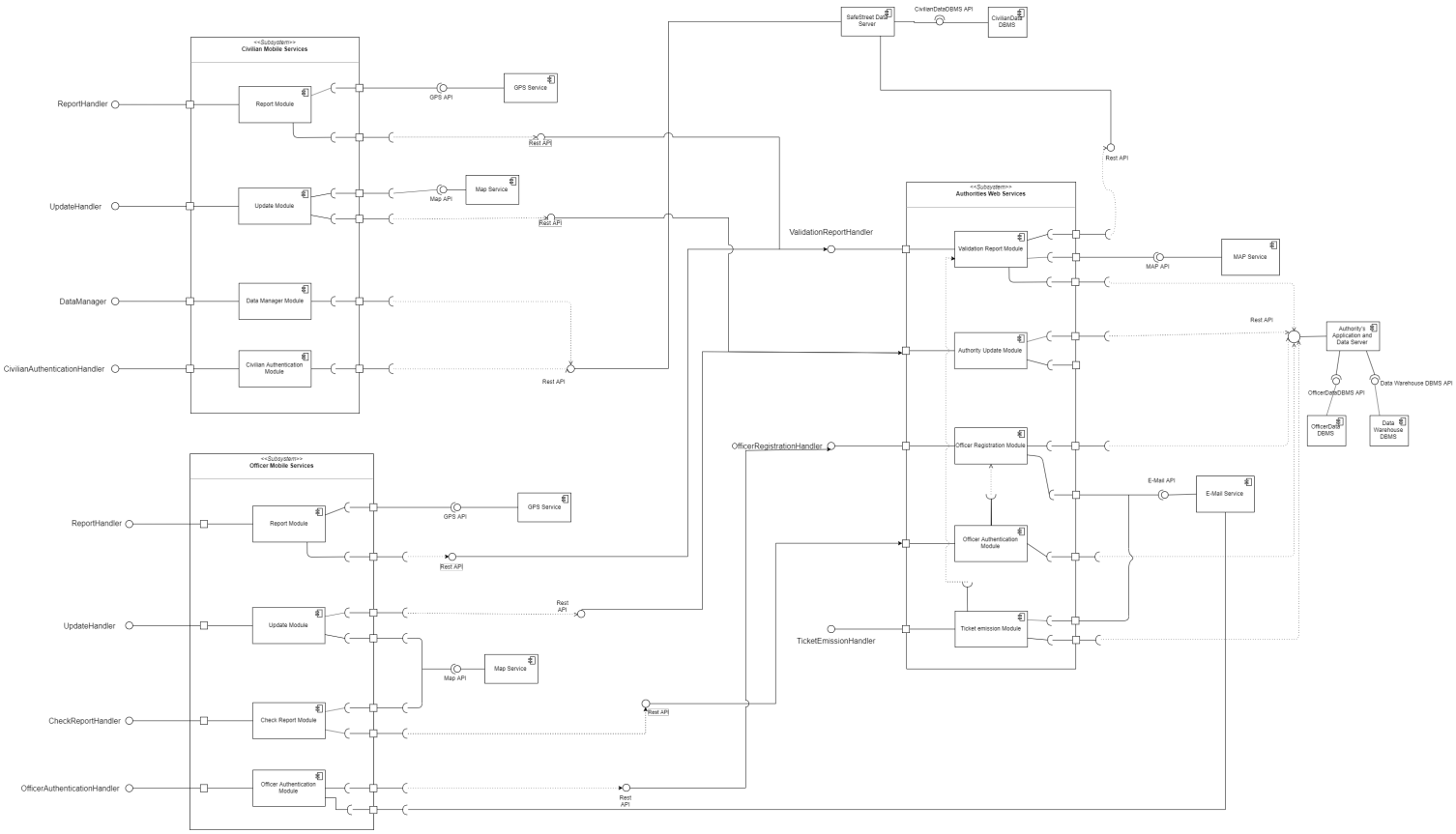
- `Authority Update Module`: this module not only handles the receiving of updates by the ADS at the web server location (the updates are the same received by the officers, plus the list of currently active officers), but also the mining which will be run in the ADS by periodically querying the DW to retrieve the data which will be packed to form the updates, that will be sent to the web servers and spread by them to the mobile units.

- `Officer Registration Module`: allows the authority to add new officers, communicating the addition both to the officer via mail and to the ADS. Also manages the first insertion of the secret pin by the officer.

- `Officer Authentication Module`: this module authenticates the officers' login requests from the Officer Authentication Module of the Officer Services, and the pin insertion from the Check Report module.

2.2.5 Complete Component Diagram

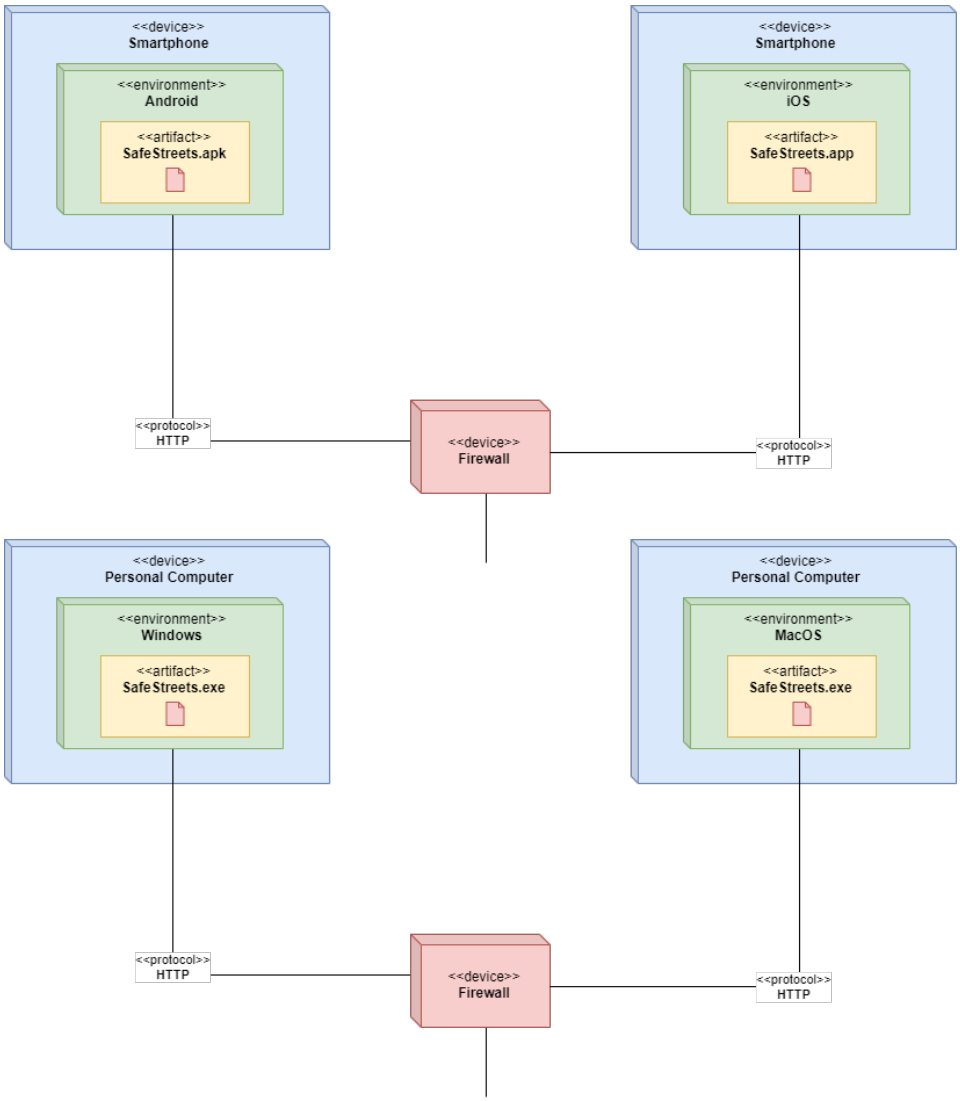
This diagram shows the interactions between all the subsystems of the application for better clarity. The presence of multiple REST APIs instead of a single comprehensive one has been modeled to show the different connections between modules without ambiguity.



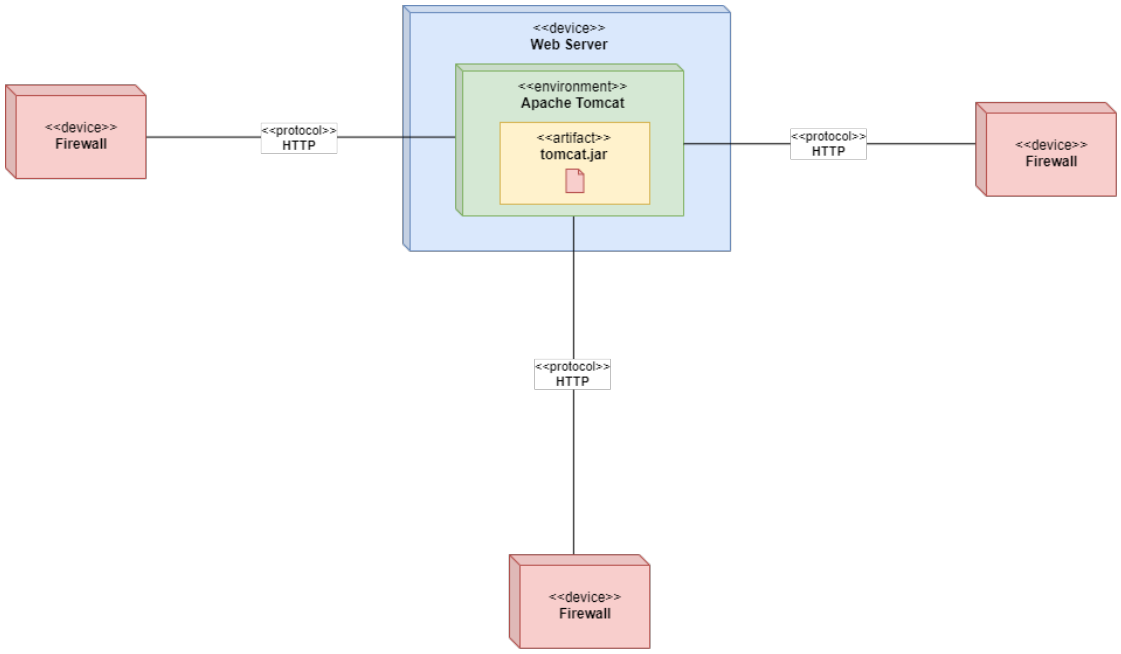
2.3 Deployment view

The architecture of the S2B, as shown in the component view paragraph, will be arranged in three tiers:

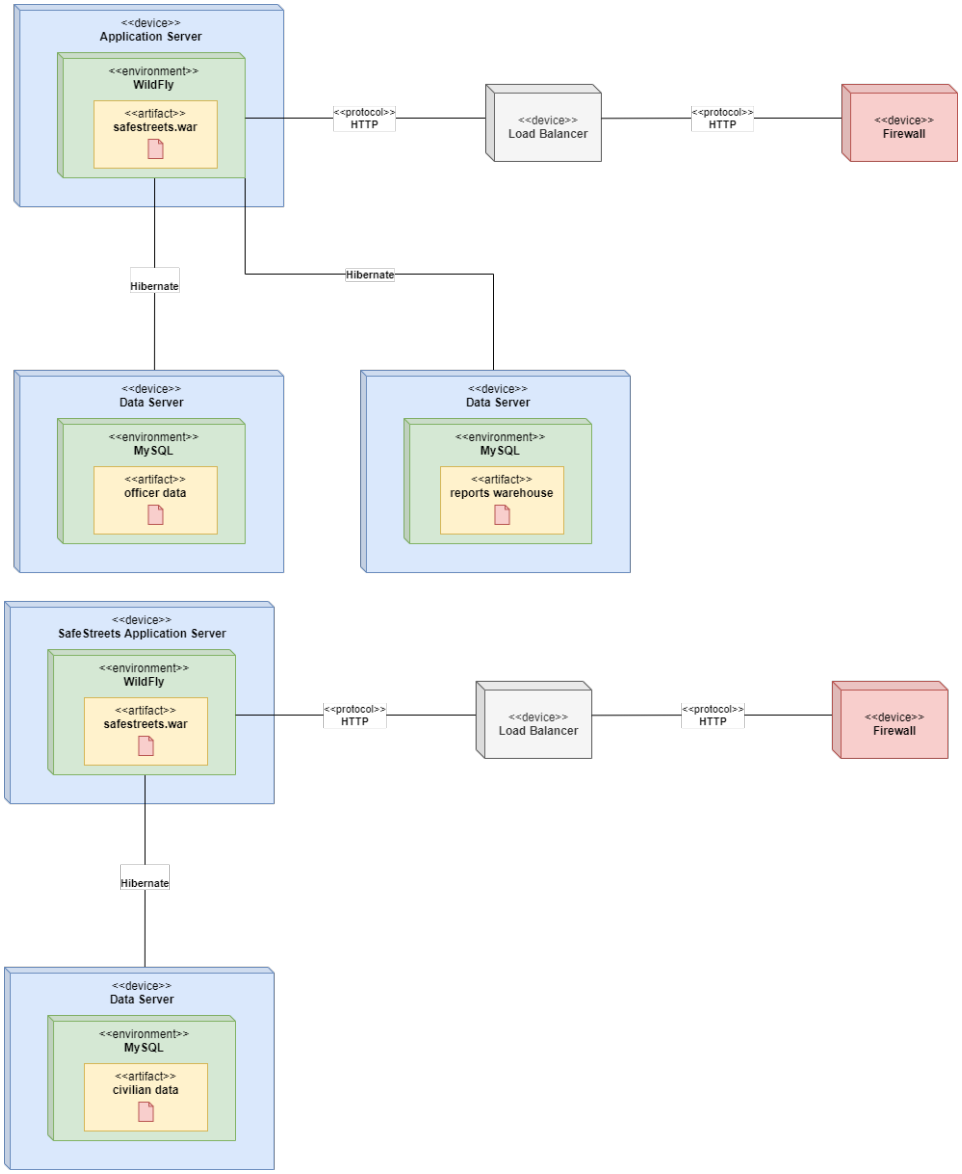
- **First tier:** is composed of the smartphones of the civilians and officers which run the mobile application, both in Android and iOS operative systems, and by the computer located at the Authority. All these clients communicate with the web server via HTTP thanks to a REST architecture, while mobile units can log in as civilians contacting the SafeStreets server.



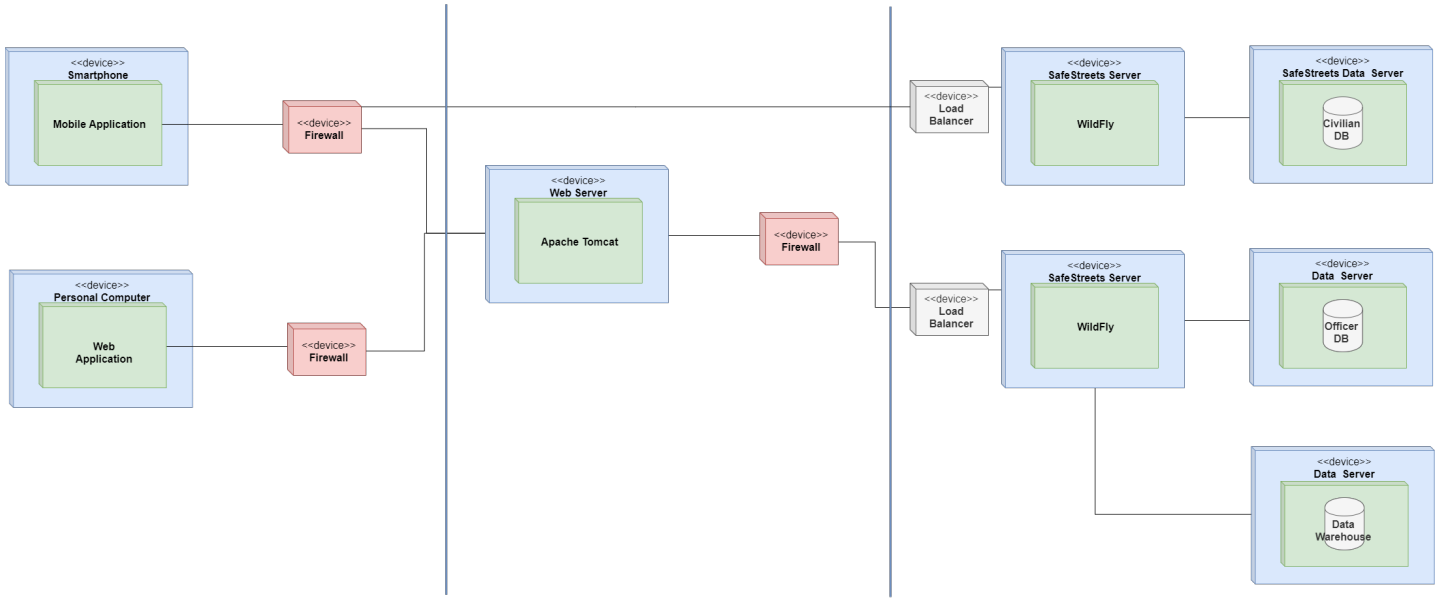
- **Second tier:** is made of the Web Server which acts as a layer of communication for clients which want to interact with the ADS. We chose Apache Tomcat as web server as it is efficient and easy to implement through Spring Boot.



- **Third tier:** the ADS with its application server and data servers, which communicate with the Officer Accounts database and with the Data Warehouse whose data is accessed through Hibernate which is again made easily interoperable with Spring Boot. The same implementation is used by the SafeStreet server which manages civilian accounts. We chose WildFly for the application servers because like Tomcat it provides a well documented and easily implementable framework.



This diagram represents the deployment view for the entire system:



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

2.5.1 External Interfaces

SafeStreets will make use of some Application Programming Interfaces to simplify the implementation, since these components are largely used and compatible with the majority of the devices currently on the market:

- `Maps API` to have a visual representation of users' reports and HFVZs, to help Officers in finding a report that coud be confirmed or rejected. For our purpose the Google Maps API is perfectly suitable, since it can be embedded in both Android and iOS applications.

- `GPS API` to geolocate the reports made by users and avoid the possibility of forging the location, which could happen if users manually enters it. GPS will also provide the application with precise timestamps for the reports.

- `REST API` every distributed component will be provided with a REST API, especially mobile clients will exploit it to interact with the web server at the Authority's location, which is the only layer visible to them. Also the SafeStreets data server's services and the ADS's will be invokable through a REST interface. Spring Boot, which uses Apache Tomcat as a web server, could be used as a versatile, easy to implement solution to build a functioning REST architecture across all subsystems.

- `DBMS API` will be used to read and write all the database components:

- one at the SafeStreets server which manages civilian accounts;

- one DW at the ADS location which holds records for all reports ever compiled and is queried to create the updates;

- another one at the ADS location which manages officer accounts. We will use MySQL as the DBMS, which will be manipulated directly from the software from the Hibernate API which can be easily configured to work with Spring Boot for simplicity and performance.

- `E-Mail API` to send an Email to the officers added to the SafeStreets project and to send the compiled ticket to the VLA.

2.6 Selected architectural styles and patterns: Please explain which styles/patterns you used, why, and how

2.7 Other design decisions

3 USER INTERFACE DESIGN: Provide an overview on how the user interface(s) of your system will look like; if you have included this part in the RASD, you can simply refer to what you have already done, possibly, providing here some extensions if applicable.

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

- Davide Cocco

Day	Subject	Hours
14/11/2019	High level components	3
19/11/2019	Component view	4
20/11/2019	Deployment view	2
Total		**

- Marco Gasperini

Day	Subject	Hours
14/11/2019	Purpose and Scope	1
19/11/2019	Component view	5
Total		**

7 References