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

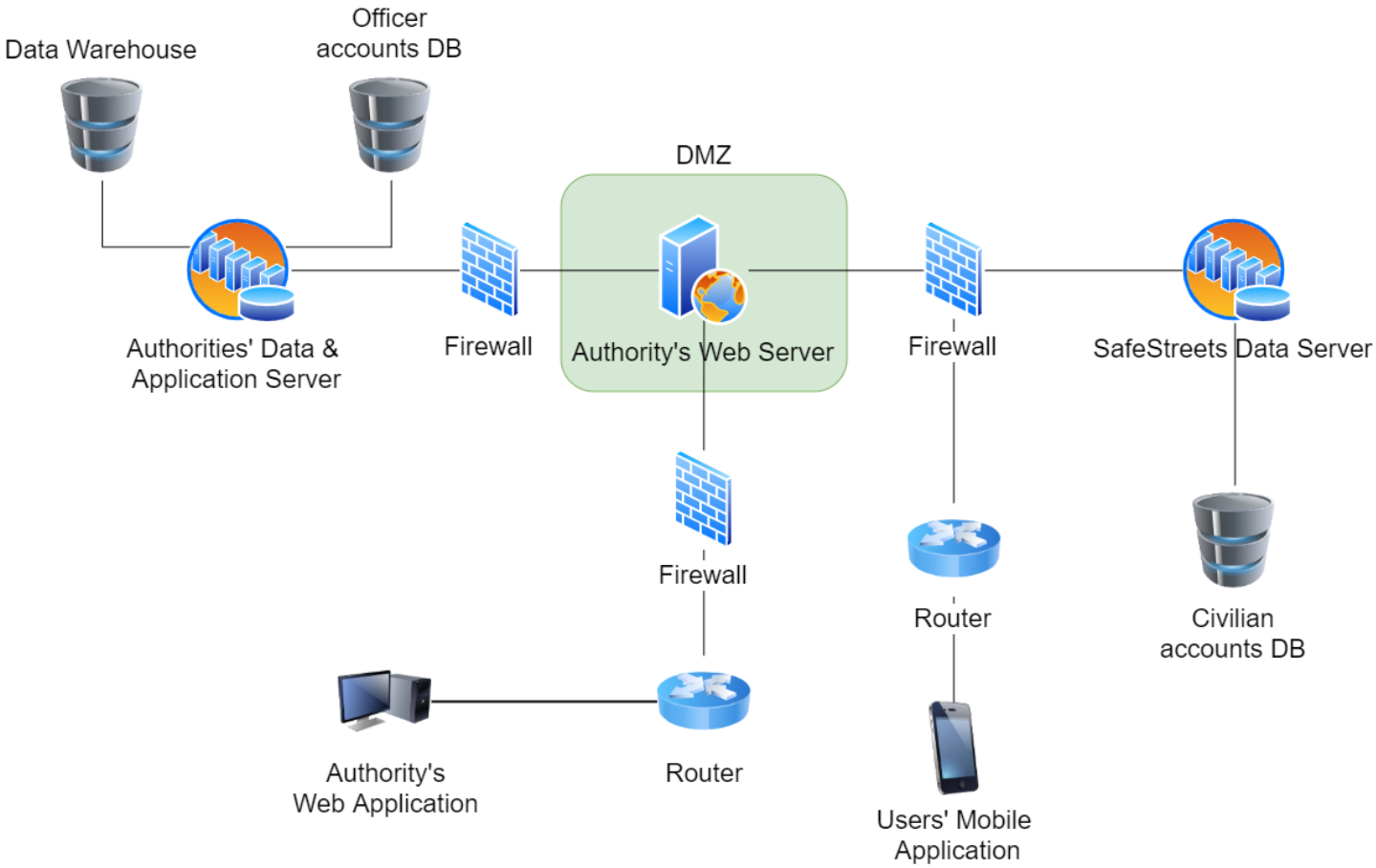


Figure 1: Figure 1: High Level Structure with a single Authority and a single User

The three logical layers of the S2B can be seen in different ways depending on the actor's client, as different actors can access different services. From the **Authority's perspective** we can see the layers distributed in a **two-tier architecture**:

- **Presentation layer:** 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 with the Application and Data server (we will call it ADS from now on) to which it sends validated reports and receives updates, the users' clients, 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.

- **Application and Data layer:** the ADS manages application functionalities such as ticket emission and update spreading, and also data functionalities such as managing the officers' accounts in an ad-hoc database and enriching a standalone data warehouse with reports to be processed. This server's location can either be in a central authority location for reduced latency, a municipality etc. and can be replicated through a RAPS architecture, (e.g. every authority can have its own server) to be able to handle multiple requests in settings like big cities.

From the **Civilian's perspective** we can see the layers distributed in a **three-tier architecture**:

- **Presentation layer:** the GUI at the mobile application, which can be used to file reports and send them to the Authority's web server.

- **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 their application server), that are periodically spread to the clients in the network. This is done to avoid having the clients know the application and data servers' access points. This way the mobile application can only contact one of the authorities' web servers and all the report processing and updating procedures are black boxed from the users perspective.

- **Data layer:** the data server at the SafeStreets headquarters, which can be replicated through a RACS architecture to be simply scalable, manages civilian accesses and bans.

From the **Officer's perspective** we have a similar perspective as the civilian's, but officers' accounts are managed by the authorities' Application and Data server, thus we can see the layers distributed in a **two-tier architecture**:

- **Presentation layer:** the GUI at the mobile application which possesses the same features as the civilians' one but also includes the abilities to check a report and sign the automated ticket by inserting the

officer's secret PIN, and also receives slightly different updates which include sensitive data to be made unreadable by civilians.

- **Application and Data layer:** just as the civilians' perspective but in this case accounts are managed in the same location where the application functionalities resiede (as officers account aren't located in the DB at SafeStreets headquarters) and thus the application and data layers are merged. Officers, just as civilians, only communicate directly with one of the city's authorities' web server which then handles interacting with the Application and Data server to provide the 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. The only logic which will be implemented at the authorities' 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.

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 have done 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.

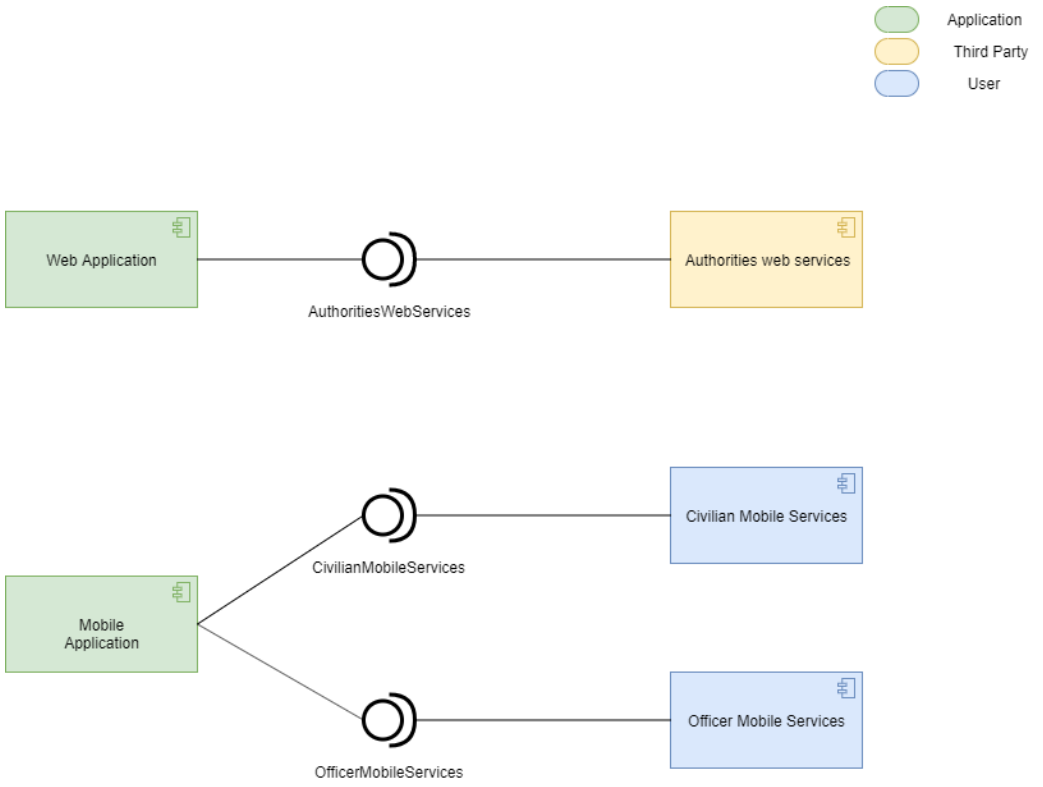
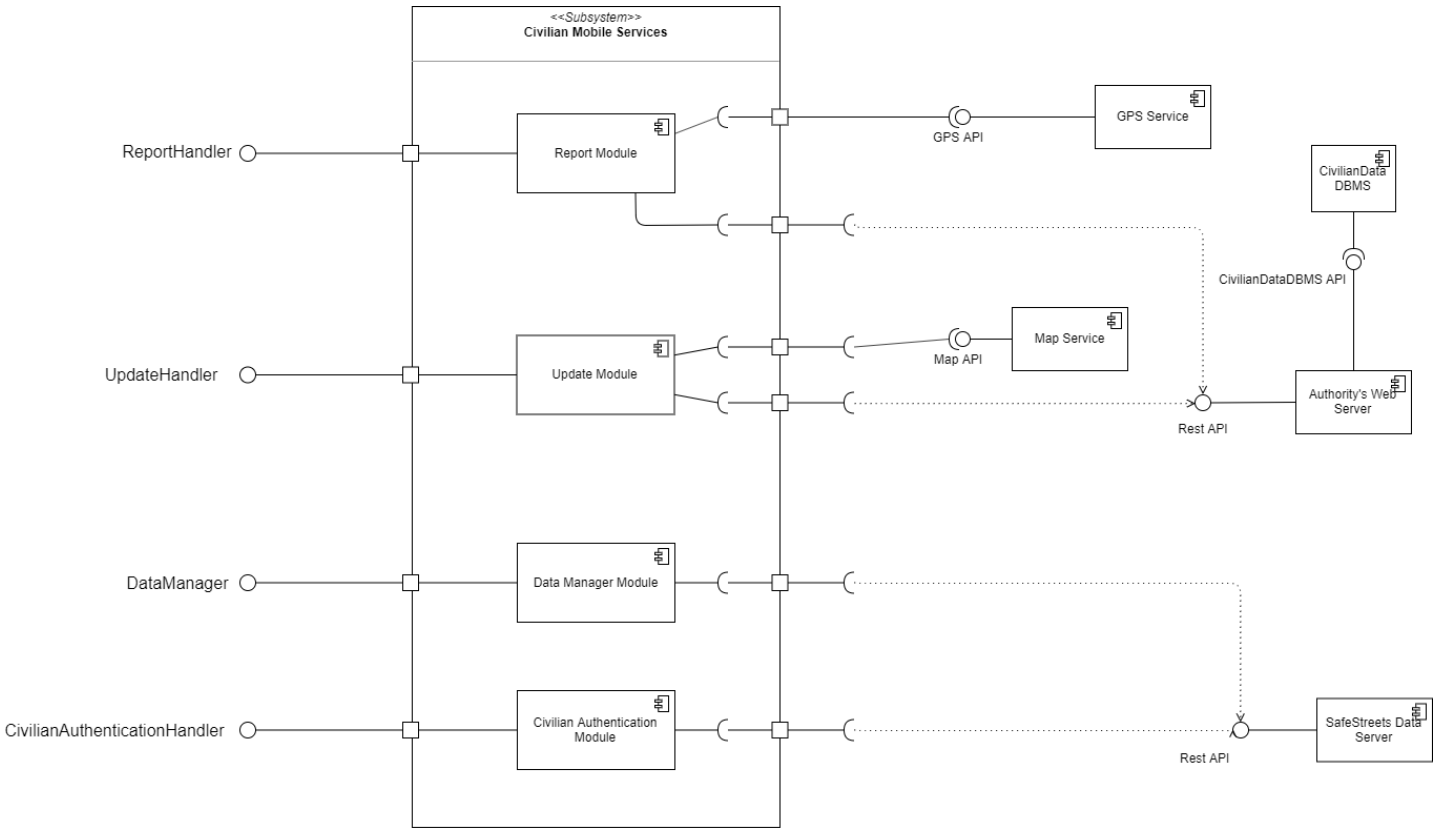


Figure 2: Figure 2: Component diagram

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.



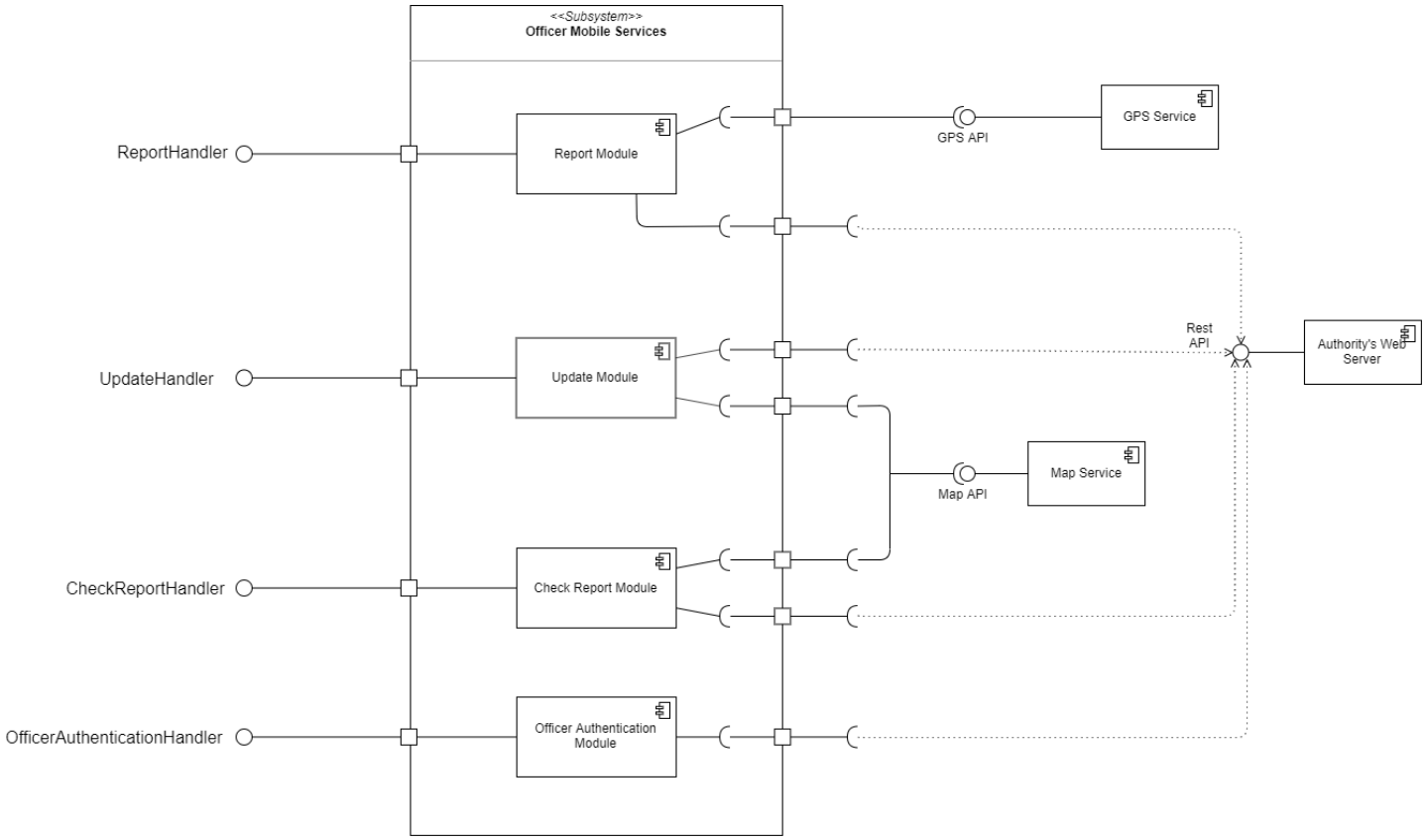
Module Functionalities

- `Report Module`: this module manages the reports done by Civilians and exploits three APIs:
  - GPS API is used to mark the reports with their corresponging place and timestamp them.
  - REST API is used to make HTTP calls, in this case towards the Authority's web server that will receive the report.

- `Update Module`: this module handles the receiving (through a REST API) and visualizing of the mined data about HFVZs, through the Map API:
  - Map API is the tool through which the data is visualized.
  - REST API is used to make HTTP calls again towards the Authority's web server to fetch the updates (thus following a pull-based policy).
- `Data Manager Module`: this module handles the users data. It allows the Civilian to manage their personal data (for instance changing their e-mail address or password):
  - REST API is used to make HTTP calls towards the SafeStreets server.
- `Civilian Authentication Module`: this module handles the registration and login requests by civilians, and thus providing the effects of an eventual ban, using the same REST API as the Data Manager Module:
  - REST API is used to make HTTP calls towards the SafeStreets server.

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`: this module manages the reports done by Civilians and exploits three APIs:

  - GPS API is used to ensure that photos' reports are taken in the same place in which they appear on the map.
  - REST API is used to make HTTP calls, in this case towards the Authority's web server that will receive the report.

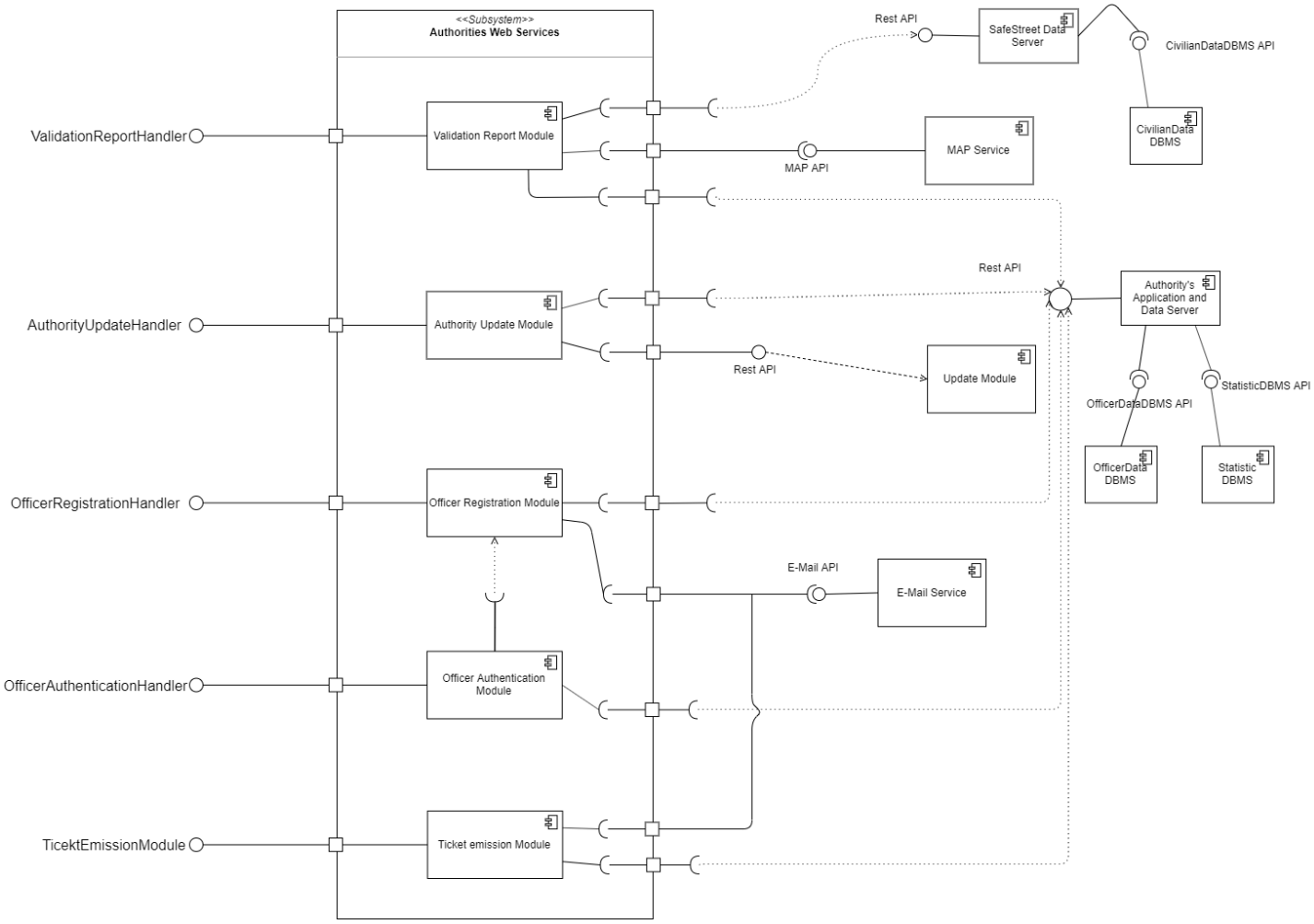
- `Update Module`: this module handles the receiving (through a REST API) and visualizing of the mined data about HFVZs, newly confirmed reports and most egregious offenders, through the Map API:
  - Map API is the tool through which the data is visualized.
  - REST API is used to make HTTP calls again towards the Authority's web server to fetch the updates (thus following a pull-based policy).

- `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:
  - REST API is used to make HTTP calls towards the Authority's web server to send them the secret PIN.
  - Map API is used by the officer to confirm he's checking a violation and show the specific data about such violation via a tooltip, so that the officer can correctly check it.

- `Officer Authentication Module`: this module handles the registration and login requests by officers.
  - REST API is used to make HTTP calls towards the Authority's web server.

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:
  - two REST APIs are used, one to communicate with the ADS which takes care of saving valid reports in the data warehouse with a DBMS.
  - the Map API which will be enriched with the newly validated reports.
- `Authority Update Module`: this module not only handles the receiving of updates by the ADS at the web server location, 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 to the mobile units.
  - a DBMS will be used to query the data to create the updates.
  - one REST API is used to transfer newly mined updates from the ADS to the Web server, and the other one to multicast the updates from the web server to the connected 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.

2.3 Deployment view

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 HFVZ, to help Officers to find a report that must 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 to avoid the possibility of forging the location which could happen if users manually enter the location.

- `Rest API` are used to simplify the communication beetween the services offered on client side and the server of Authorities and SafeStrees.

- `OfficerDataDBMS API` to allows Authority to communicates with data Warehouse in which officer's data are collected.

- `CivilianDataDBMS API` to allows SafeStreets to communicates with database in which civilian's data are collected and to ban a civilian in case of tragression.

- `StatisticDBMS API` to provide some methods to Authority's Application and Data Server for mining data through Statistic DBMS on Data Warehouse and to make thees statistics available to civilians and officers.

- `E-Mail API` to send an Email to the officers that have to join 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
Total		**

- Marco Gasperini

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

7 References