POLITECNICO MILANO 1863

Software engineering project

SafeStreets

DD : Design Document.

Authors:
Mattia Micheloni,
Francesco Montanaro,
Amedeo Pachera.

Professor:
M.Rossi.

Version 1.0
08/11/19

Contents

Chapter 1

Introduction

This chapter explains

1.1 Purpose

The purpose of this document is to give a more detailed description of the architectural choiches made to perform into the software system all the functionalities described into the RASD document.
Therefore, if the RASD describes all the features that the system must indulge according to the stakeholders needs, the DD describes how it has been thought to convert all the high-level system requirements into the design structure.
It provides guidance and material which is intended to assist technical staff to perform system testing and implementation and, moreover, to facilitate future maintenance and features extension.
List of topics covered by the document:

- High-level architecture.
- Main Components, interfaces and deployment.
- Runtime behavior.
- Design Patterns.
- UI details.
- Mapping between requirements and architecture.
- Implementation, integration and testing plan.

1.2 Scope

Here a short review of SafeStreet scope referred to what just stated in the RASD document.
Safestreet is a software thought to make users able to report traffic violations to authorities. Anyone can report violations simply opening SafeStreet application and taking several pictures of them. For each violation is required at most 6 pictures in which 1 of the involved vehicle license plate.
In order to guarantee to authorities the reliability of data gathered, which means that pictures can not be altered or modified, the latter can be only taken in the application itself.
To make it easier the reporting process all other information are automatically mined by the system, such as time, date, geographical location and license plate; the last one is extracted by the system from the license plate picture.
Finally, user is asked to choose the violation type from a predefined set and confirm the above listed data that will be sent.
If no errors occur, the system stores the violation, otherwise it is refused.
Authorities, in order to access information concerning reports made by users, must log into the SafeStreet website.
They are allowed to discard, modify and validate violations sent by users, in order to take into account only pertinent violations.
Thanks to the data granted by the SafeStreet initiative and the validation performed by authorities, they can even decide to generate traffic tickets from validated violations.
Authorities can also upload information about accidents that occur on the municipality.
The software provides also a service to offer both users and authorities statistics computed on the available data. Obviously, in order to make statistics consistent and reliable, the latter are generated only by information validated by authorities.
Statistics highlight streets or areas with the highest frequency of violations, the vehicles that commit the most ones or the infractions trend that occurs over time.
Crossing data gathered by users and authorities, the system can identify potentially unsafe areas and suggest interventions in order to improve streets safety.

1.3 Definitions, acronyms, abbreviations

This section gives some definitions in 1.3.1, acronyms in 1.3.2 and abbreviation in 1.3.3 which will be use in the document, in order to explain some concept and help the general understanding.

1.3.1 Definitions

1.3.2 Acronyms

- API: Application Programming Interface
- GPS: Global Positioning System
- UI: User interface
- OCR: Optical Character Recognition
- DBMS: DataBase Management System

1.3.3 Abbreviations

- Gn: nth Goal.
- R_n : nth Requirement.

1.4 Revision history

- Version 1.0 : 10/12/2019

1.5 Reference documents

- Design document: “Mandatory Project Assignment AY 2018/2019”.
- Lecture slides of professor M.Rossi and E.di Nitto of Politecnico di Milano.

1.6 Document's overview

The DD document is structured by seven chapters as describes below:

- Chapter 1:
- Chapter 2:
- Chapter 3:
- Chapter 4:
- Chapter 5:

Chapter 2

Architectural design

2.1 Overview: High-level components and their interaction

[Overview]

2.2 Component view

2.3 Deployment view

2.4 Runtime view

2.4.1 Report a violation

The sequence diagram in figure 2.1 shows the flow of reporting a violation. The flow starts from the mobile application which asks to the Violation Generator to create a report. The generator check if the device settings are acceptable (no fake-GPS, internet connection available..) and sends an error message in negative result. If settings are accepted the flow proceeds taking pictures from the mobile camera and the GPS location and finally reporting to the server subcomponent UploadViolation Manager all the pictures and metadata. This last subcomponent interacts with the DBMS to insert in the database the new data of the violation.
This operation can be done in loop.

2.4.2 Validate a violation

The sequence diagram in figure 2.2 shows the flow of validating a violation. The flow starts from the website which asks to the web authentication manager to login into the system. This subcomponent forward the request to the server authentication manager which interact with the DBMS to verify the correctness of credentials and return an error message in negative result.
After the authentication, a request to the website violation manager is made, which send it to the Get Violation manager on server. This last subcomponent interacts with the DBMS to query the database (a notification is sent in the case of empty list).
The result is bark-propagated to the website in which a violation can be modified. After the update, a request is made to the web violation manager passing the new violation and the update, this data are forwarded to the Update Violation Manager on the server which interacts with the DBMS to patch the violation.
This operation can be done in loop.
This flow is similar in the case of deleting the violation, with the difference of Delete Violation Manager instead of Update Violation Manager on the server.

2.4.3 Report an accident

The sequence diagram in figure 2.3 shows the flow of reporting an accident. The flow starts from the website which asks to the web authentication manager to login into the system. This subcomponent forward the request to the server authentication manager which interact with the DBMS to verify the correctness of credentials and return an error message in negative result.
After the authentication, a request to the website accident manager is made, passing all the data needed, which send them to the upload accident manager. This last subcomponent interacts with the DBMS to insert in the database the new data of the accident.
This operation can be done in loop.

2.4.4 Request statistics

The sequence diagram in figure 2.4 shows the flow of requesting a statistic. Firstly all the type of statistics are computed by the Compute Statistic Manager on the server. This subcomponent request data to the DBMS, makes all the computations and insert the new data to the database.
Then in the mobile app a request to the Mobile Statistics Manager (on the

website the process is the same) passing the type of statistics desired. This subcomponent forward the request to the Retrieve Statistic Manager on the server which interacts with the DBMS to get data from the database and back-propagate the results to the mobile application.
In the case of request a map-typed statistic the UI component request to GoogleMapsAPI the map passing the list of coordinates to plot.
This operation can be done in loop.

2.5 Component interfaces

2.6 Selected architectural style and patterns

2.7 Other design decision

2.7.1 License plate recognition algorithm

Tesseract is a free OCR engine, it can recognize words of approximately every language. It offers a huge variety of API for any type of programming languages and also a C/C++ library to developers who want to build their own application on it. Tesseract is very powerful in recognizing black on white text so it's perfect for license plate recognition, and it has also an API for Android and iOS developers so it's optimized to be run on mobile devices, which make easy to SafeStreets to use it directly on devices instead of running it in the server

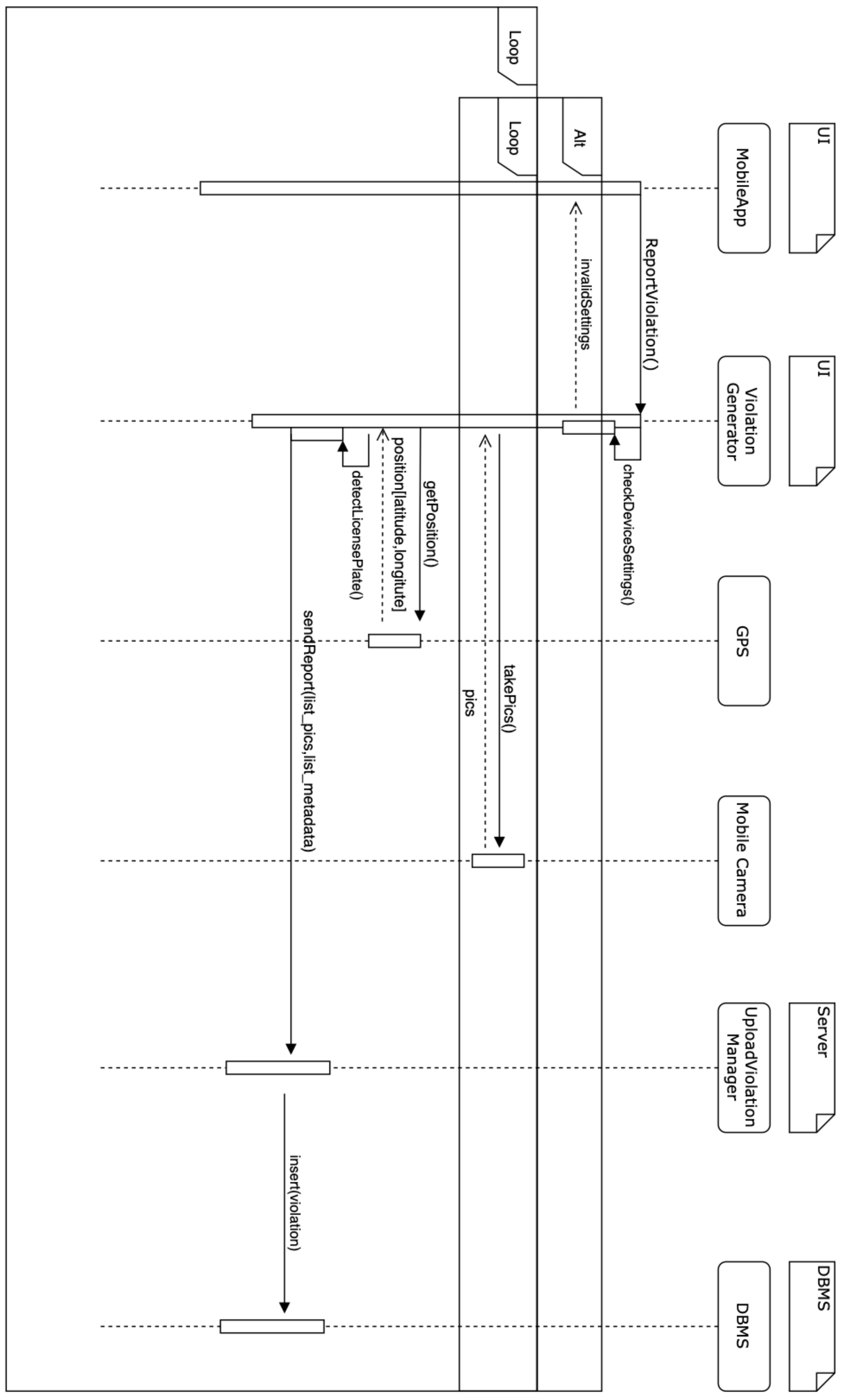

Figure 2.1: Sequence diagram of reporting a violation

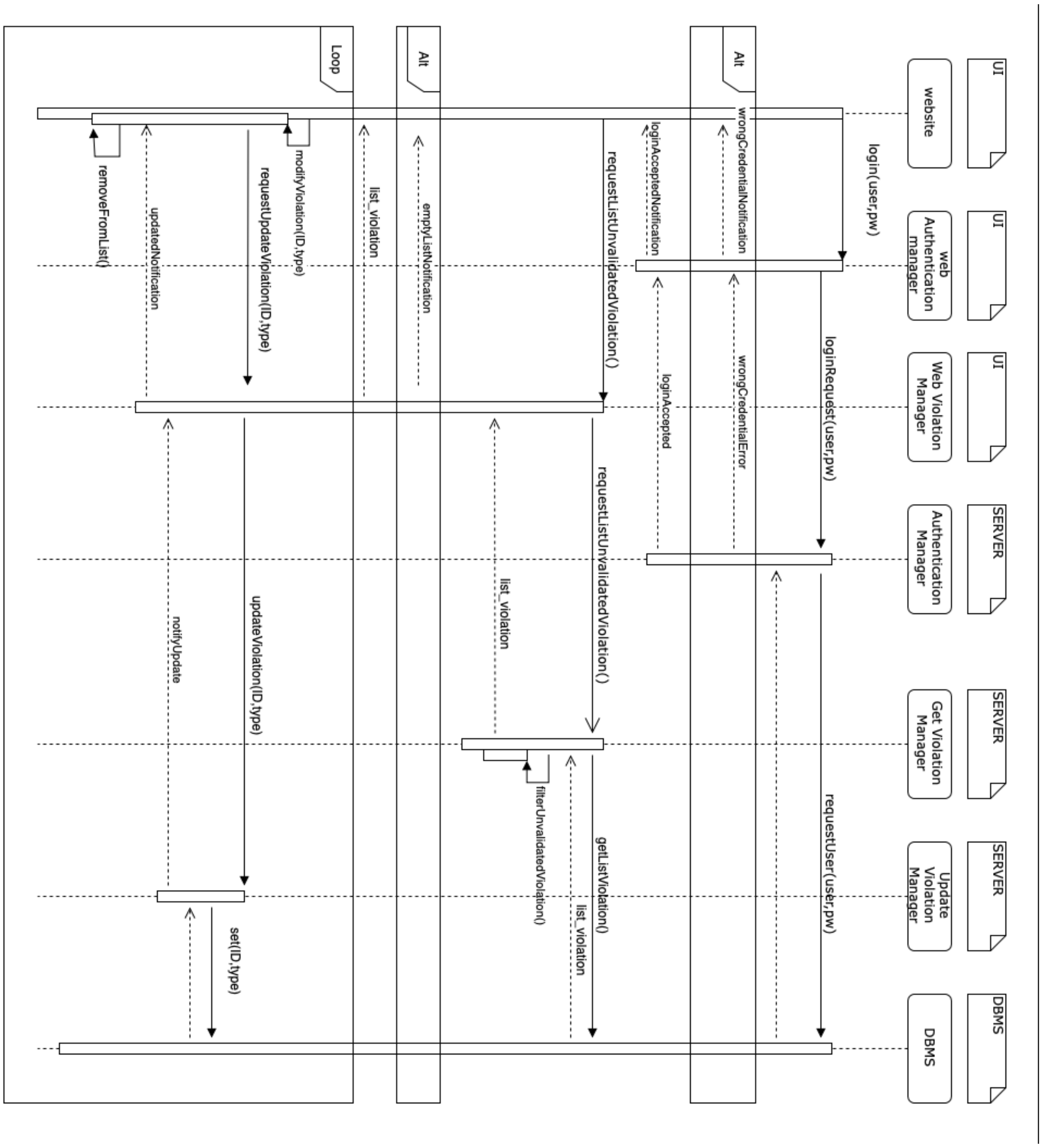

Figure 2.2: Sequence diagram of validating a violation.

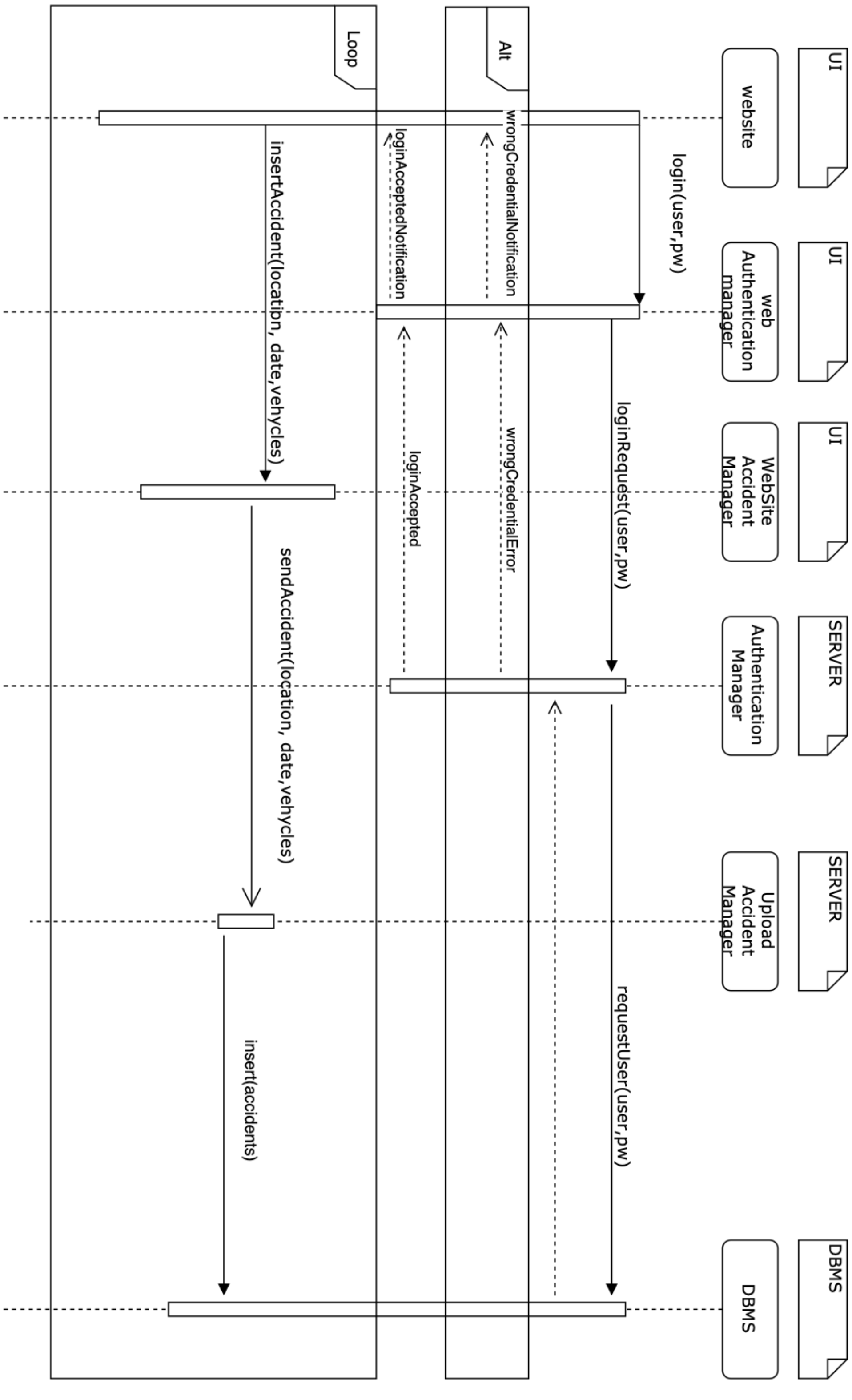

Figure 2.3: Sequence diagram of reporting an accident.

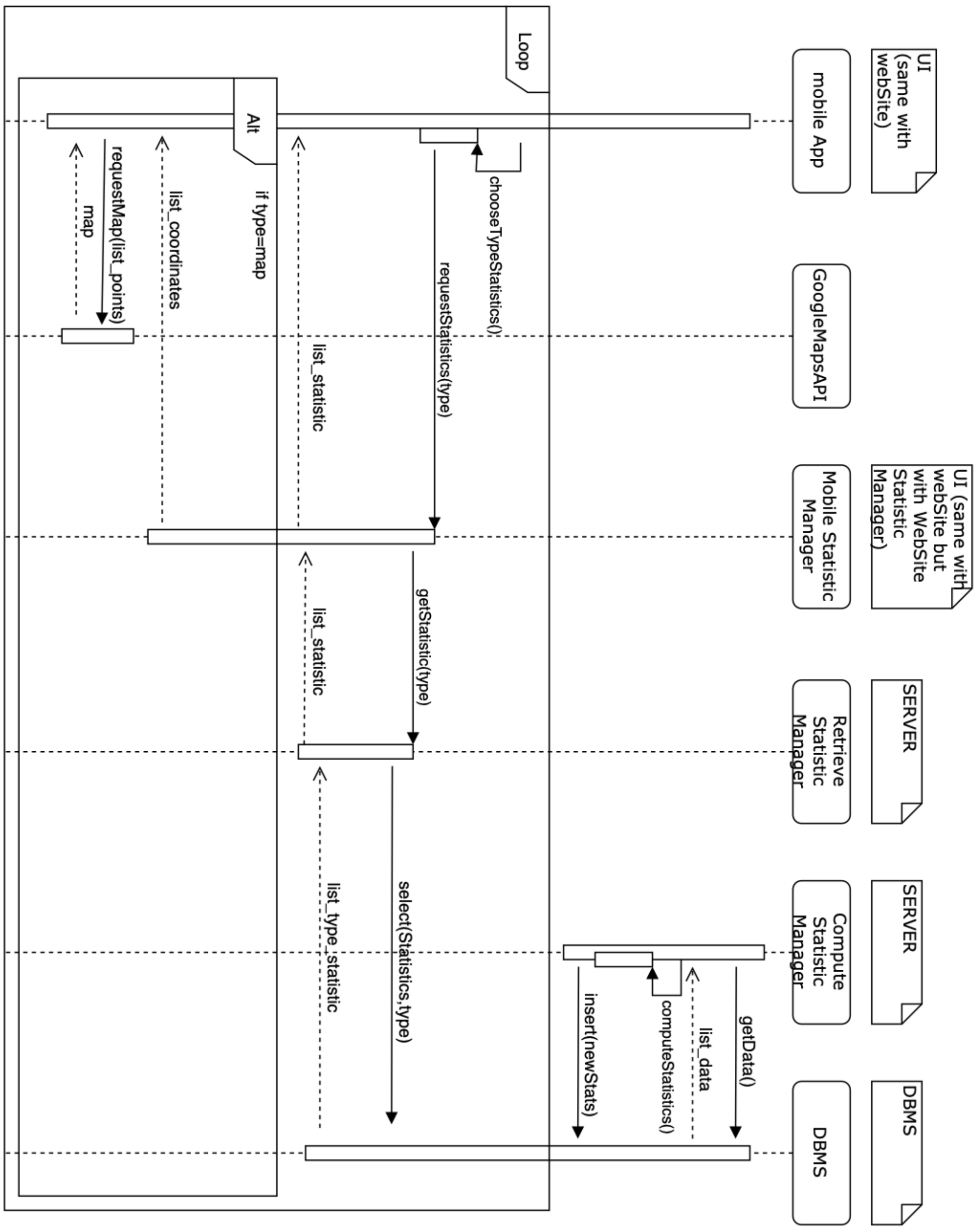

Figure 2.4: Sequence diagram of requesting statistics.

Chapter 3

User interface design

Chapter 4

Requirements traceability

This chapter explains which of the components are used to meet the requirements defined in the RASD document (in order to achieve the prefixed goals). It contains also a general description on how each component should work and how they interact with the others.

- R_1: The system allows to take pictures only within the software itself.
  $UI \rightarrow MobileApp \rightarrow ViolationGenerator$: This component allows users to take pictures only within the Mobile Application itself, through the device‚Äôs camera interface.

- R_2: The report can be made only if the internet connection is available.
  $UI \rightarrow MobileApp \rightarrow ViolationGenerator$: Before making a report, this component checks if the device's internet connection is available and works properly. Otherwise it prevents the user from proceeding with the report.

- R_3: The system imposes the user to include a picture of the license plate of the vehicle that committed the violation.
  $UI \rightarrow MobileApp \rightarrow ViolationGenerator$: During the process of taking pictures this component ensures that at least one of them contains the license plate of the vehicle involved in the violation.

- R_4: The system gets date and time from its internal clock when the picture is taken.
  $UI \rightarrow MobileApp \rightarrow ViolationGenerator$:

- R_5: The system obtains the violation's position from the device's GPS.
  $UI \rightarrow MobileApp \rightarrow ViolationGenerator$: When the first picture is taken, this component retrieves the geographical positioning of the user‚Äôs mobile device from its GPS.

- R_6: The system converts the geographic coordinates obtained in the corresponding address.
  $UI \rightarrow MobileApp \rightarrow ViolationGenerator$: Before sending a report to the system this component converts the geographical positioning of the user‚Äôs mobile device in the corresponding address. Then, it integrates it in the reporting information.

- R_7: The system automatically detects the vehicle's license plate number.
  $UI \rightarrow MobileApp \rightarrow ViolationGenerator$: When the license plate picture is taken this component launches an algorithm to extract the license plate number. If for any reason an error occurs, it will impose the user to take the picture again.

- R_8: Access to the violation reports is only permitted to authorities.
  $UI \rightarrow WebSite \rightarrow WebsiteViolationManager$: This component allows authorities to access to the Violation‚Äôs Information. In particular, it sends a request to the server component and awaits for its response.
  $Server \rightarrow ReportsManager \rightarrow GetViolationManager$: This component receives the request to access the violations from the UI and queries the database in order to get them. Then, it will send a response to the UI containing the information requested.

- R_9: The system stores violations data.
  $UI \rightarrow MobileApp \rightarrow ViolationGenerator$: If a report has been formulated correctly, this component sends a request to the server containing the report information in order to upload them.
  $Server \rightarrow ReportsManager \rightarrow UploadViolationManager$: This component receives the request to store the violation from the UI and write it into the database.

- R_{10}: Authorities must log in to the system.
  $UI \rightarrow WebSite \rightarrow WebsiteAuthenticationManager$: This component allows authorities to insert the credentials in order to access to the Website‚Äôs services. Then, it will send a request to the server and awaits for its response.
  $Server \rightarrow AuthenticationManager$: It collects the requests coming from the Website UI containing the credentials and checks for their correctness. Then, it sends back a positive or negative response.

- R_{11}: The system checks that a fake GPS service is not active on the user's device.
  $UI \rightarrow MobileApp \rightarrow ViolationGenerator$: Before making a report,

this component checks if the user has not a fake GPS service activated. Otherwise it prevents the user from proceeding with the report.

- R_{12}: Authorities can modify, discard or validate the violation reports present on the system.
  $UI \rightarrow WebSite \rightarrow WebsiteViolationManager$: This component allows authorities to edit the information of a violation with an appropriate interface. Then, it sends a request to the server component containing the information to update.
  $Server \rightarrow ReportsManager \rightarrow EditViolationManager$: It collects the requests coming from the UI component and uploads the information stored into the database.

- R_{13}: Authorities can insert data concerning accidents that occur on the territory.
  $UI \rightarrow WebSite \rightarrow WebsiteAccidentsManager$: This component provides an interface through which authorities can insert the information on an accident that occurred on the territory. Then, it sends a request to the server containing the information to upload.
  $Server \rightarrow ReportsManager \rightarrow UploadAccidentManager$: It collects the requests coming from the UI interface concerning the insertion of a new accident and writes in the database.

- R_{14}: The system builds statistics based on validated data.
  $Server \rightarrow StatisticManager \rightarrow ComputeStatisticsManager$: This component queries the database in order to get all the validated violations and computes statistics on it. Then, it uploads them into the database.

- R_{15}: The system produces statistics concerning possible unsafe areas built on the accidents information.
  $Server \rightarrow StatisticManager \rightarrow ComputeStatisticsManager$: This component queries the database in order to get the accidents uploaded by authorities and computes ‚Äúunsafe areas‚Äù statistics. Then, it uploads them into the database.

- R_{16}: Authorities and users can set the temporal granularity and the category of the statistics that they want to consult.
  $UI \rightarrow Website \rightarrow Websitestatisticsmanager$: It provides the tools through which authorities can filter the statistics they are interested in. Then, it sends a request to the server containing the information to retrieve and awaits for its response.
  $UI \rightarrow MobileApp \rightarrow MobileAppStatisticsManager$: It provides the

tools through which users can filter the statistics they are interested in. Then, it sends a request to the server containing the information to retrieve and awaits for its response.
$Server \rightarrow StatisticManager \rightarrow RetrieveStatisticsManager$: This component collects the requests concerning the statistics to retrieve coming from the UI component and queries the database in order to get them. Then, it sends back a response containing the information requested.

- R_{17}: Authorities can consult the interventions suggested by the system.
  $UI \rightarrow WebSite \rightarrow WebsiteStatisticsManager$: It allows authorities to send a request to the server component in order to retrieve the information concerning the intervention suggestions built by the system. Once a response is returned it display the information in a correct way.
  $Server \rightarrow StatisticManager \rightarrow RetrieveStatisticsManager$: This component collects the requests coming from the UI component concerning the intervention suggestion and queries the database in order to get them. Then, it sends back a response containing them.

- R_{18}: The system has an associated suggestion for each type of violation.
  $Server \rightarrow StatisticManager \rightarrow ComputeStatisticsManager$:

- R_{19}: The system places a marker on the map where the intervention is suggested.
  $UI \rightarrow WebSite \rightarrow WebsiteStatisticsManager$:
  $Server \rightarrow StatisticManager \rightarrow RetrieveStatisticManager$

- R_{20}: The system allows to filter the suggestions on the map by category.
  $UI \rightarrow WebSite \rightarrow WebsiteStatisticsManager$: It allows authorities to set the suggestion category to retrieve. Then, it sends a request to the server component in order to retrieve the information and awaits for a response.
  $Server \rightarrow StatisticManager \rightarrow RetrieveStatisticManager$: This component collects the requests coming from the UI component concerning the intervention suggestion category and queries the database in order to get them. Then, it sends back a response containing them.

Chapter 5

Implementation, integration and test planning

Chapter 6

Effort spent

Mattia Micheloni

TASK	HOURS
Ch.1	1
Ch.2	4
Ch.3	18
Ch.4	12

Francesco Montanaro

TASK	HOURS
Ch.1	4.5
Ch.2	4
Ch.3	13
Ch.4	13

Amedeo Pachera

TASK	HOURS
Ch.1	2
Ch.2	3
Ch.3	13
Ch.4	14

Chapter 7

References