SAMUELE MOSCATELLI, ANDREA POZZOLI

SAFESTREETS - DD

Version 1

10th November 2019

SAFESTREETS

Software Engineering 2 Project

Andrea Pozzoli and Samuele Moscatelli

SafeStreets DD

Software Engineering 2 project

Politecnico di Milano

TITLEBACK

This document was written with LaTeX

CONTACTS

☞ pozzoliandrea97@gmail.com

☞ sem.mosca@gmail.com

CONTENTS

1 INTRODUCTION

1.1 PURPOSE

This DD (Design Document) document aims to continue with the work done in the RASD document, this time going much more in details in the definition of the system SafeStreets, describing the high-level architecture and the technical aspects that characterize it. In particular, the computational components and the interactions among these components are presented and explained accurately, clearing up their role, their behaviour and their implementation. The audience to whom the document is addressed is represented by the development team, which will rely on the principles exposed in this document in order to implement the system.

1.2 SCOPE

SafeStreets is a crowd-source project which has three different targets of people:

- The citizens (referred to as End Users)
- The Authorities
- The municipalities

The core idea is that of giving people the opportunity of participate in making their cities safer from the point of view of traffic regulation. In particular, the primary aim of the project is to give citizens a concrete tool that can grant them the possibility of signal a traffic violation to authorities. For this reason the End User is the key category for the correct functioning of the application. In more details, SafeStreets software allows citizens to report at any moment a traffic violation that is being perpetrated in front of them by specifying the type of the infringement, the description of it and also attaching a picture that depicts the vehicle, being sure to include also the license plate, so that the system can have a guarantee of the truthfulness of the information and at the same time identify the transgressor. Then the application automatically detects the time and the date of the report, together with the position from which it has been sent. So, for example, if a citizen, while walking through via Golgi on 30th October 2019, sees a car with license plate XXX parked in the middle of the bike lane, he can open SafeStreets, take a photo of the vehicle location, insert the type of violation and a description and then send the report to the system. Once received the information, the application reads the license plate from the picture and store it together with the other data provided by the user and the position, date and time. So, in the example previously shown, SafeStreets would memorize a parking violation of the

type "car on bike lane" in via Golgi, on 30th October 2019, with license plate XXX, and a brief description of the situation. All the traffic violations sent by an end user are not lost, in fact every end user can see on the application his past contribu- tion to the traffic regulation. An end user cannot see the traffic violations sent by the other users.

Once received the information, SafeStreets firstly has to make it available for reading by all the authorities, also notifying those to whom is assigned the area in which the violation has occurred. The authorities notified can also decide to go and check directly the infringement, so, in order to avoid a concentration of them on the same event, they can also warn the other authorities of it. Another functionality offered to this type of users, as to all the other, is the possibility of mining some information from the system. In particular, SafeStreets has to elaborate the data received and combine them together in order to calculate traffic violation statistics. The possible statistics are: which streets are characterised by the highest number of infringements, in which moment of the day there are more violations, which type of violation is most perpetrated. The statistics regarding traffic violations can be accessed by all the three types of user. Instead data regarding a specific traffic violation is visible only by authorities.

Another type of user is municipality user, who can collaborate with SafeStreets with the aim of making roads under his jurisdiction safer through prevention. In particular, municipality provides information about accidents that has occurred and occur on its territory to the system. The system then can merge this data with those coming from violations and in this way it can identify the most dangerous areas, and at the same time suggest the best interventions that can be applied to make them safer.

1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

1.3.1 Definitions

- **End user:** The end user is a person that sees a traffic violation and wants to notify the authorities about it by using SafeStreets application. He can't see the violations sent by other users. He can see the statistics. End user is one of the three user types of the system.
- **Authority:** Authority is the second type of user and usually he is a police man. He does not send violation data to the system. He can see the violations sent by the end users. He can access to the statistics.
- **Municipality:** Municipality represents mayor and municipal employees of a city which decide to collaborate with SafeStreets and it is the third type of user. Municipality sends data about accidents to SafeStreets in order to allow it to cross them with violations data and find out the unsafe areas on its territory. He also can see the statistics.
- **System:** The system is a synonymous of SafeStreets. The system receives data form the end users, elaborates and stores data, shows data

to the authorities, calculates statistics and unsafe areas, suggests interventions.

- **Traffic violation:** Data sent by a end user is called traffic violation or only violation. A traffic violation is composed by a license plate (taken from a picture or a text inserted by the user), date, time, GPS position, the type of violation and a description of it. Example of violations can be vehicles parked on the stripes or in places reserved to people with disabilities, double parking, parking in no parking places.
- **Statistics:** The statistics are some information calculated by SafeStreets in order to highlight the streets with a high number of violations, the days and times at which there are more violations, the most common types of violations.
- **Unsafe Area:** Municipality sends to SafeStreets all the accidents occurred in a city. The streets and the areas in which there is a high number of accidents are called unsafe areas. An unsafe area can be only in one municipality jurisdiction and two different municipality jurisdiction cannot have the same unsafe area.
- **Intervention:** After having discovered some unsafe areas, SafeStreets suggests to municipality some actions (interventions) to do in order to make these areas safer.
- **Accident:** an accident is data that municipality sends to SafeStreets in order to find the unsafe areas and suggest interventions. It describes a dangerous situation that occurred in one of the streets or areas under the municipality jurisdiction.

1.3.2 Acronyms

- RASD – Requirement Analysis and Specification Document
- API - Application Programming Interface
- GPS - Global Positioning System
- RMI - Remote Method Invocation
- UML - Unified Modeling Language

1.3.3 Abbreviations

- Gn: n-goal.
- Dn: n-domain assumption.
- Rn: n-functional requirement.
- rn: n-row of the matrix.

1.4 REVISION HISTORY

This is the first version of the document. The date of release of this document is 9th December 2019.

1.5 REFERENCE DOCUMENTS

1.6 DOCUMENT STRUCTURE

The first chapter is a brief introduction to the document. It describes the purpose and the scope of SafeStreets software. In order to understand better the document, the first part also presents definitions, acronyms and abbreviations that will be used in the document.

The second chapter represents the core of the document. The first section is a formal overview of the application, so here specific styles used are identified and exposed. Then the document get more into architectural details of the system, showing and explaining the architectural components and their distribution with respect to the each other and to the hardware resources. In order to do so, appropriate UML diagrams (component diagram, class diagram and deployment diagram) are presented and outspread, giving a static view of the application architecture. Then also a dynamic view is given through the use of sequence diagrams, so that to show in a more detailed way the behaviour of the components and the interactions among them, defining in this way how the system responds to the external invocations. Subsequently, the attention is focused on the interfaces offered by the previously presented components, going more in detail in the explanation of the methods exposed by them. Finally, patterns used in the architecture and other design decisions are shown.

The third chapter gives a precise and polish idea of the user interface design, displaying how they actually are implemented at the implementation level.

The forth chapter represents requirements traceability, so requirements are allocated to the introduced components, highlighting which of them are involved in the satisfaction of which goals.

The chapter number five involves the planning of implementation, integration and testing phases, referring to what is expressed in chapter two, and so relating the introduced components with the way in which they will be implemented and the way in which the system will be integrated and tested.

Finally chapters number six and seven concern respectively the effort spent by Andrea Pozzoli and Samuele Moscatelli in order to complete this document and the references consulted during the development of the project.

2 ARCHITECTURAL DESIGN

2.1 OVERVIEW

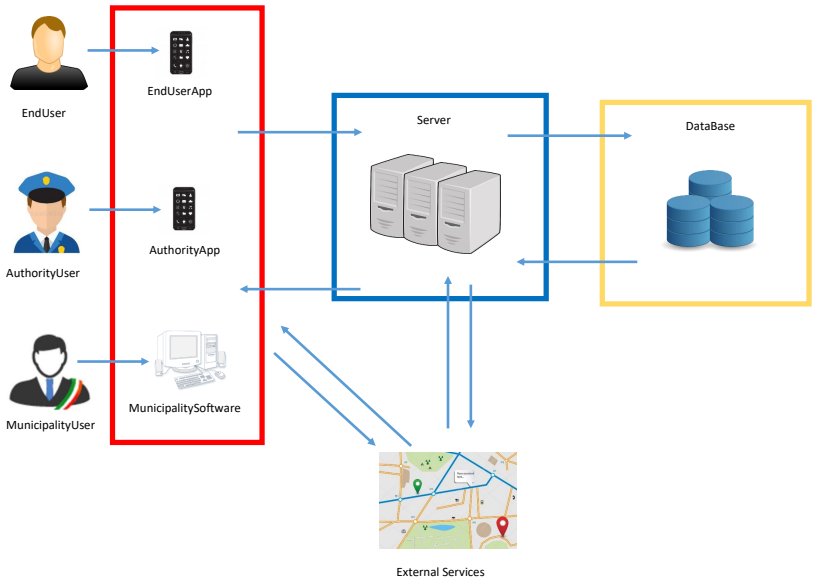

Figure 1: Overview Diagram

This diagram aims to show an overview of the SafeStreets environment. The picture highlights the three tiers architecture that it has been given to the SafeStreets system. The structure of the system is client-server. In the red box there are the three types of client, each one represents one type of user of the system. In the blue box there is the server, which communicates with the clients, the database and the external services. In the yellow box instead there is the database. In the diagram it is presented also the external services, in particular they consist in the map tools, provided by OpenStreetMap, that are used by the server to collect and process data, and by the users to visualise information.

The presentation layer is developed in the client side. The application layer is divided between the server and the client and so the system has a fat client architecture. The data layer is contained in the database. The arrows that connect the boxes represent the requests and the responses that occur between the components.

2.2 COMPONENT VIEW

2.2.1 Component Diagram

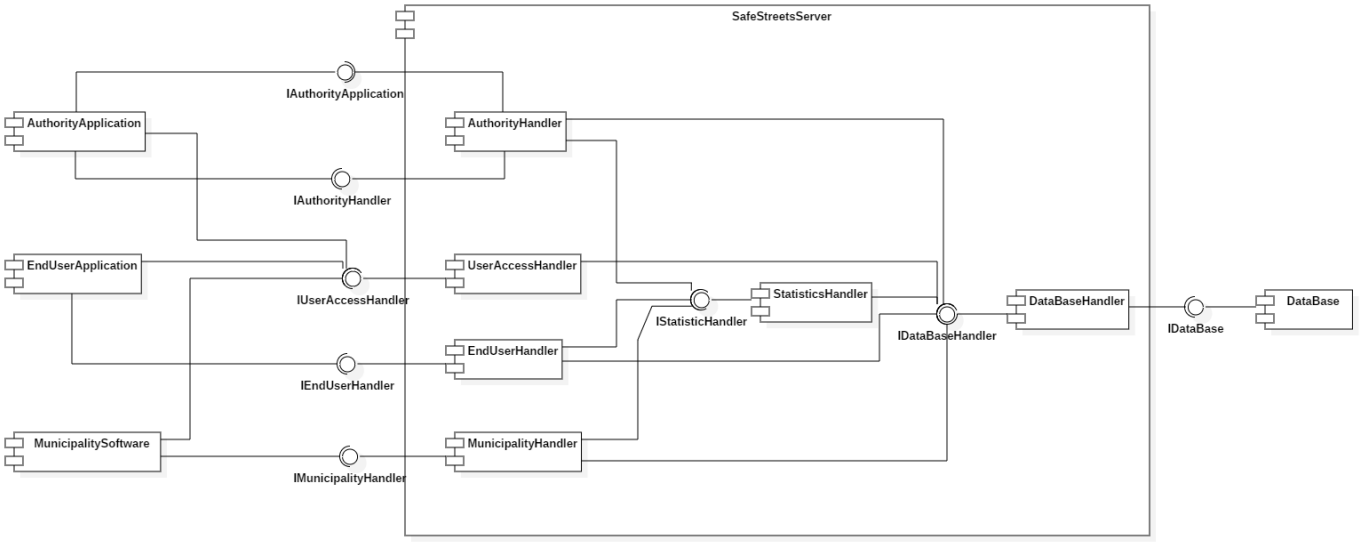

Figure 2: Component Diagram

The component diagram aims to give a static view of the SafeStreets system. Also in the component diagram the three tiers architecture is highlighted. In fact, on the left there are the components that represent the different types of client. In the middle there is the server which is divided in several handlers. Finally, on the right there is the database component. In order to understand better the component structure, it is necessary to explain all the components and the interfaces that they expose.

2.2.2 Clients

AuthorityApplication

The AuthorityApplication component depicts the application that is used by an authority user in order to participate in SafeStreets project. This component exposes the IAuthorityApplication interface which is necessary to the AuthorityHandler in order to notify the authorities and to get their position.

EndUserApplication

The EndUserApplication component stands for the end user application that is used by the citizens in order to send new traffic violations, see past contributions and watch statistics. This component does not expose any interface.

MunicipalitySoftware

The MunicipalitySoftware component represents the software that a municipality user uses in order to collaborate with SafeStreets. Also this component does not expose any interface.

2.2.3 SafeStreets Server

UserAccessHandler

The UserAccessHandler component symbolises the handler that manages the logins and the registrations of all the types of users. This component exposes the IUserAccessHandler interface that is used by all the clients in order to access the SafeStreets system.

AuthorityHandler

The AuthorityHandler component depicts the handler that manages the authority users. This component exposes the IAuthorityHandler interface that is used by the authority users (AuthorityApplication component) in order to communicate with the SafeStreets server. In particular, the authorities can request all the violations, can be notified and can see the statistics.

EndUserHandler

The EndUserHandler component represents the handler that manages the end users. This component exposes the IEndUserHandler interface that is used by the end users (EndUserApplication component) in order to communicate with the SafeStreets server. In particular, the end users can send new traffic violations, can see their own past contributions and can watch the statistics.

MunicipalityHandler

The MunicipalityHandler component stands for the handler that manages the municipality users. This component exposes the IMunicipalityHandler interface that is used by the municipality users (MunicipalitySoftware component) in order to communicate with the SafeStreets server. In particular, the municipality users can send accidents data and can get detected unsafe areas and interventions.

StatisticsHandler

The StatisticsHandler component symbolises the handler that manages and processes the statistics. This component exposes the IStatisticsHandler interface that is used by all the user handlers (AuthorityHandler component, EndUserHandler component, MunicipalityHandler component) in order to take the data regarding the statistics and provide them to the users.

DataBaseHandler

The DataBaseHandler component acts as the handler that manages the access to the database and the operations on it. This component exposes the IDataBaseHandler interface that is used by all the user handlers (AuthorityHandler component, EndUserHandler component, MunicipalityHandler component) in order to take and send data regarding the users, the violations, the unsafeareas, the accidents. Moreover, this interface is used also by the StatisticsHandler to take data regarding the violations in order to create and update the statistics.

2.2.4 Database

The DataBase component represents the third tier, the one which contains all the data and depicts the data layer. This component exposes the IDataBase interface that is used by The DataBaseHandler component in order to take and store data regarding the users, the violations, the unsafe areas and the the accidents.

2.2.5 Class Diagram

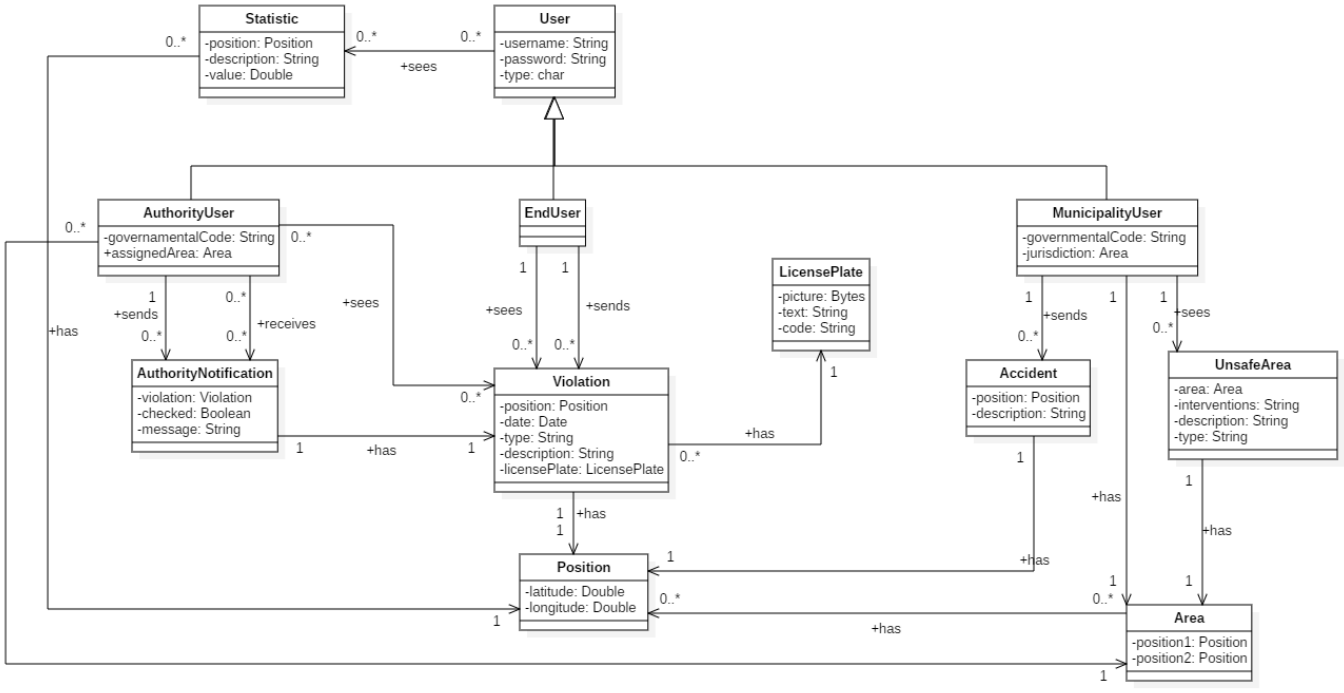

Figure 3: Class Diagram

In this subsection of the component view section, a class diagram is shown with the aim of explaining the data that are exchanged in the system. In order to understand better the diagram it is necessary to describe all the classes.

User

The User class is an abstract class that defines the attributes that are in common between the different types of user of the system. In fact, this class has as attributes an username, a password and a type of user. Because of all the types of user can see the statistics, this class is linked with the Statistics class.

EndUser

The EndUser class represents the end user of the system, so a citizen that uses SafeStreets, and it is linked with a generalisation arrow to the User class. The functionalities that belong to an end user are sending a new traffic violation and seeing his own past contributions (violations sent by an end user in the past). For this reason, the EndUser class is linked twice with the Violation class, in order to divide and highlight the two different functionalities.

AuthorityUser

The Authority class stands for an authority user of the system and it is linked with a generalisation arrow to the User class. An authority can see all the violations sent by the end users so this class is linked with the Violation class. Moreover, an authority can be notified if a new traffic violation is in his assigned area and he can notify other authorities in the same area that he is going to check a traffic violation. These two actions are shown in the diagram with two different arrows that go from the Authority class to the AuthorityNotification class. An authority, for registering as an authority user, must have a governmental code and so it is an attribute of the class. An authority has also a assignedArea attributed that defines the area which is under the control of the authority.

MunicipalityUser

The Municipality class symbolises the municipality user and it is linked with a generalisation arrow to the User class. Also the municipality user must demonstrate his role with a governmental code and so there is the governmentalCode attribute in this class. A municipality user sends the data regarding the accidents to SafeStreets and so this class is linked to the Accident class. Moreover, he receives the unsafe areas under his jurisdiction and for this reason the Municipality class is connected to UnsafeArea class. The jurisdiction under control of the municipality user is defined by an attribute jurisdiction that is an area.

Statistic

The Statistic class represents the statistics regarding the traffic violations. It has a position, a description and a value, and so there are three attributes for defining these three features. (The statistics are shown on a map so there is an attribute referring to the OpenStreetMap class.)

Violation

The Violation class is a traffic violation that is sent by an end user and seen by all the authorities. As attributes this class has a position (and so it is linked to Position class) that represents the location in which the violation occurs, a date, a type and a description. Moreover, the violation refers to a license plate, so there is also the licensePlate attribute.

LicensePlate

The LicensePlate class represents the license plate of the veichle that commits the traffic violation. It has an attribute picture that is the photo of the vehicle taken by the end user, and two attributes of type string, one for the eventual written license plate (if the software does not recognise the photo) and one with the license plate number taken from the photo or from the text.

AuthorityNotification

The AuthorityNotification class stands for the notification that is sent to an authority when there is a new violation in his assigned area or when another authority of the same area is going to check a violation. For this reason there is an attribute with Violation type, a message that explains the notification, and a boolean attribute that specifies if an authority is going to check the violation.

UnsafeArea

The UnsafeArea class symbolises an unsafe area detected by SafeStreets and sent to a municipality user. This class has as attributes an area in order to locate it, and three strings, one referring the suggested interventions, one the type of unsafe area and one a brief description of the danger area.

Accident

The Accident class is data sent by municipality to SafeStreets and regarding the accidents that occur under the municipality jurisdiction. In this class there is an attribute that is the position in which the accident occurs and another attribute that is a description of the accident.

Position

The position class represents a geographical position and so it has two double attributes, one for the latitude and one for the longitude.

Area

The area class represents a geographical rectangle individuated between two positions. For this reason, this class has as attributes, two positions.

2.3 DEPLOYMENT VIEW

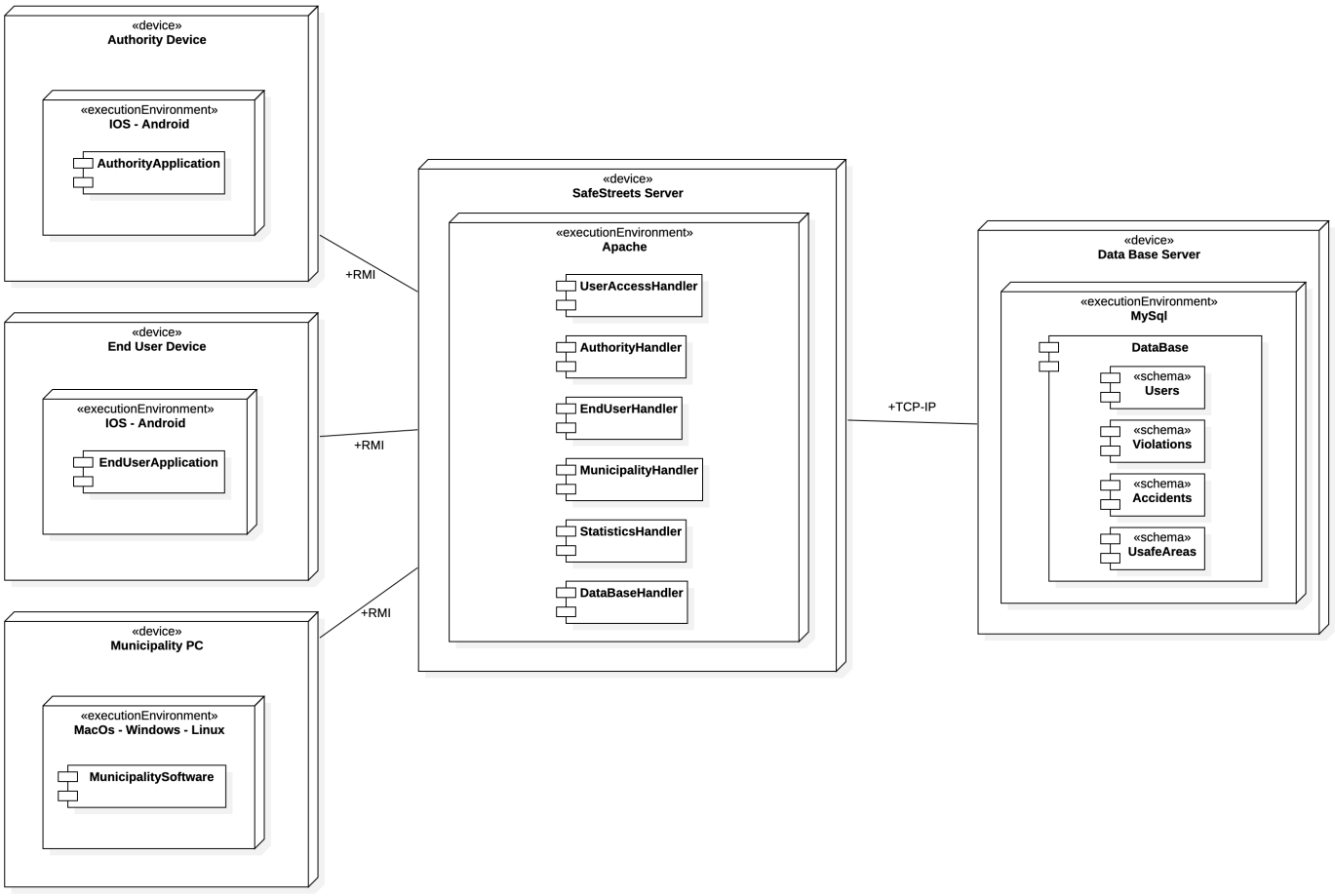

Figure 4: Deployment Diagram

Also the deployment diagram shows clearly the three tiers architecture. On the left there are the three different types of client (one type for each kind of user). In the middle there is the server. On the right there is the database. For explaining this diagram, it is necessary to enter in the details of each node.

Authority Device

The Authority Device is the smartphone or tablet that an authority user uses for participating in the SafeStreets system. As execution environment the device runs on iOS or Android operating system, and in particular the AuthorityApplication is installed on it. The authority device needs also a GPS sensor and so it is displayed as a node inside the device.

End User Device

Similarly to the previous node, the End User Device is the smartphone or tablet that an end user uses for participating to the regulation of the streets. Again, the execution enviroment is iOS or Android as operating systems and

the EndUserApplication is installed on the device. In order to work correctly, the device must have a GPS sensor and a camera.

Municipality PC

The Municiapality PC is the computer from which the municipality user acesses to the system. The MunicipalitySoftware is installed on the device and it runs for MacOS, Windows and Linux (so they are the execution environment).

SafeStreets Sever

The clients and the server communicate each other through RMI protocol. The SafeStreets server runs on Apache as execution environment. The server contains all the handler components previously presented for the component diagram.

Data Base Server

The DataBase and the SafeStreets server communicate each other through TCP/IP protocol. The databse execution enviroment is mySQL and the database contains some schemes, one for the users data, one for the accidents data and one for the unsafe areas data.

2.4 RUNTIME VIEW

After having shown a static view of the system, in this section and in the following one a dynamic view is presented. In particular, here some sequence diagrams are displayed and then described. For each functionality available to the users, a sequence diagram is built.

Sequence Diagram 1: An end user create a new traffic violation

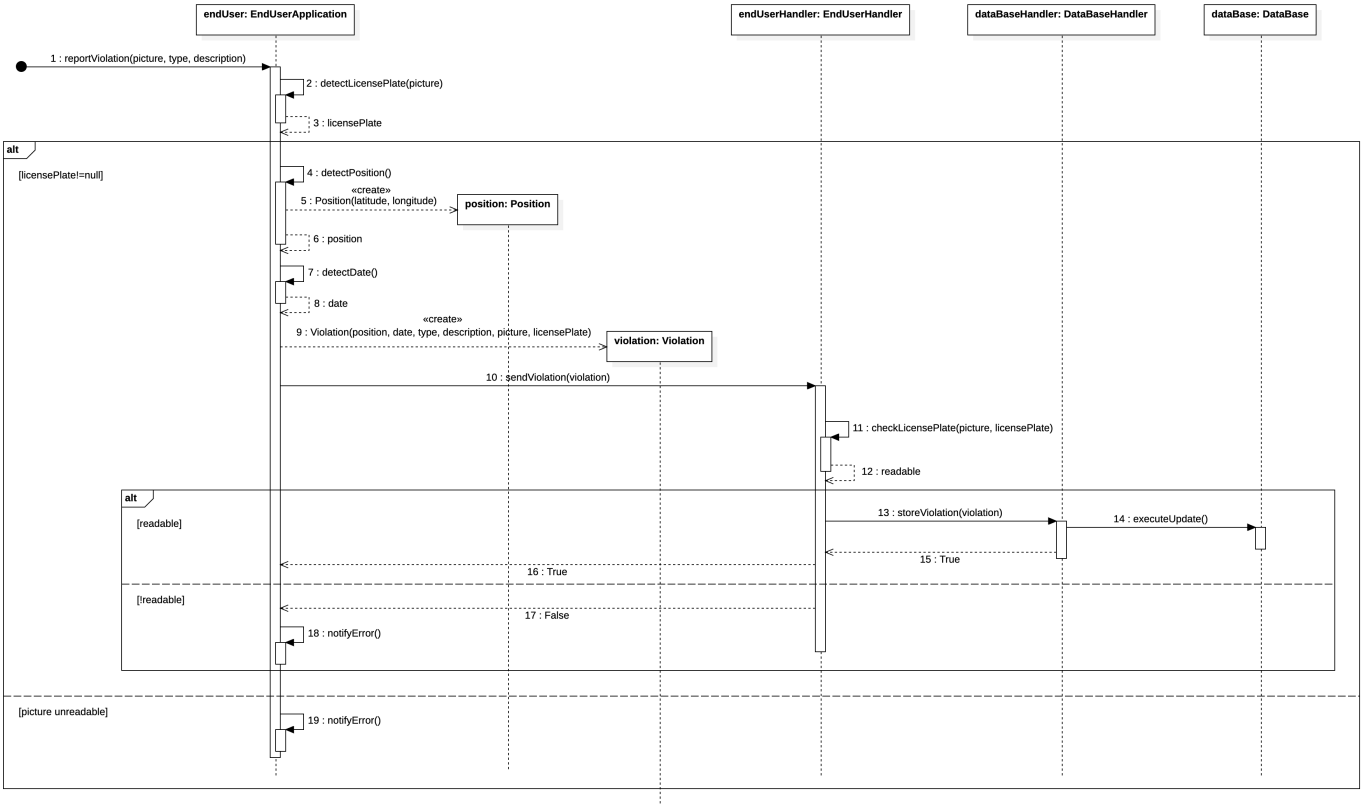

Figure 5: Sequence Diagram 1

In this first sequence diagram is shown the situation in which an end user creates and sends a new traffic violation. Going further in details, as soon as he pushes the "Send" button, the application immediately tries to read the license plate of the vehicle from the picture inserted. If the picture is readable then the position and the date (including time) are detected from the device, so that the violation can be instantiated. At this point, the violation is sent to the EndUserHandler which immediately further checks the correspondence between the license plate read by the application and the picture, so that if the result is positive the violation is sent to the DataBaseHandler which inserts the new violation data in the data base and a "True" message is returned to the application and then the proceeding is concluded; instead if the result is negative a "False" message is sent to the application which, for this reason, notifies the end user of the problem, making him take another picture or insert the license plate manually (so restarting the proceeding). The same result is obtained if the application itself is not able to read the licence plate.

Sequence Diagram 2: An end user sees his past contributions

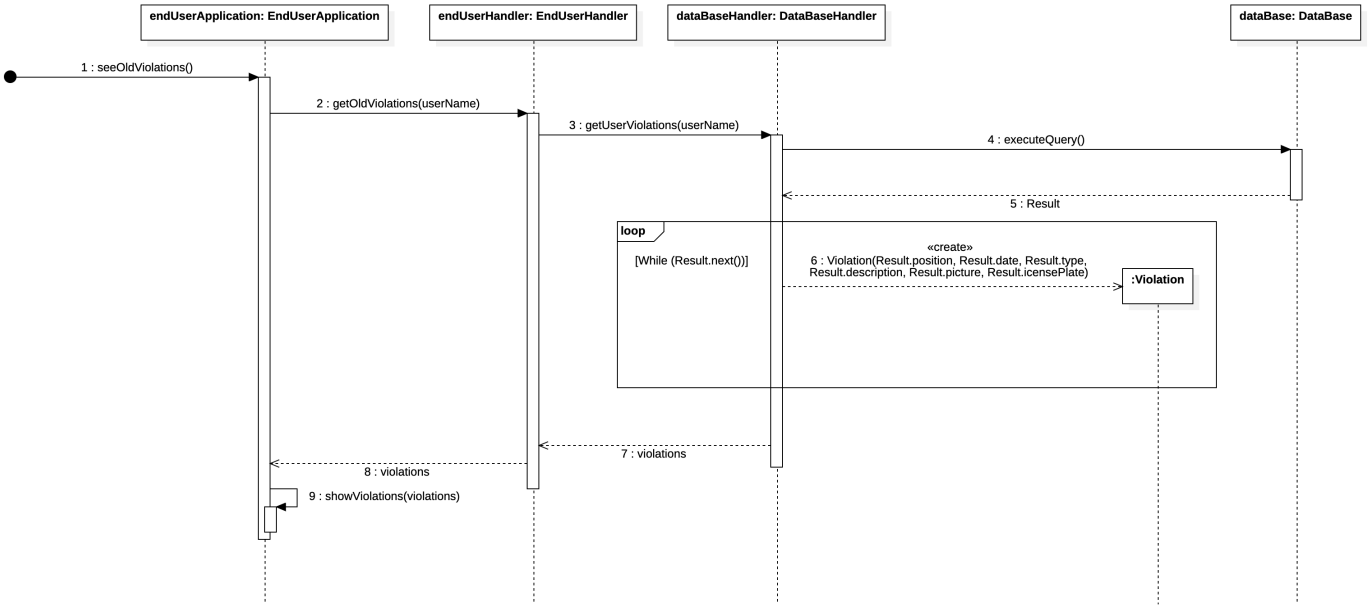

Figure 6: Sequence Diagram 2

In this second sequence diagram is shown the situation in which an end user wants to see the violations sent by himself in the past. In particular, the EndUserApplication asks the end user's past contributions to the EndUserHandler which, in turn, sends a request to the DataBaseHandler. The latter executes the query on the data base and when it gets the results it uses them to create a list of violations which is returned back to the EndUserHandler and then to the EndUserApplication, which can show them to the user.

Sequence Diagram 3: An authority user sees all the traffic violations

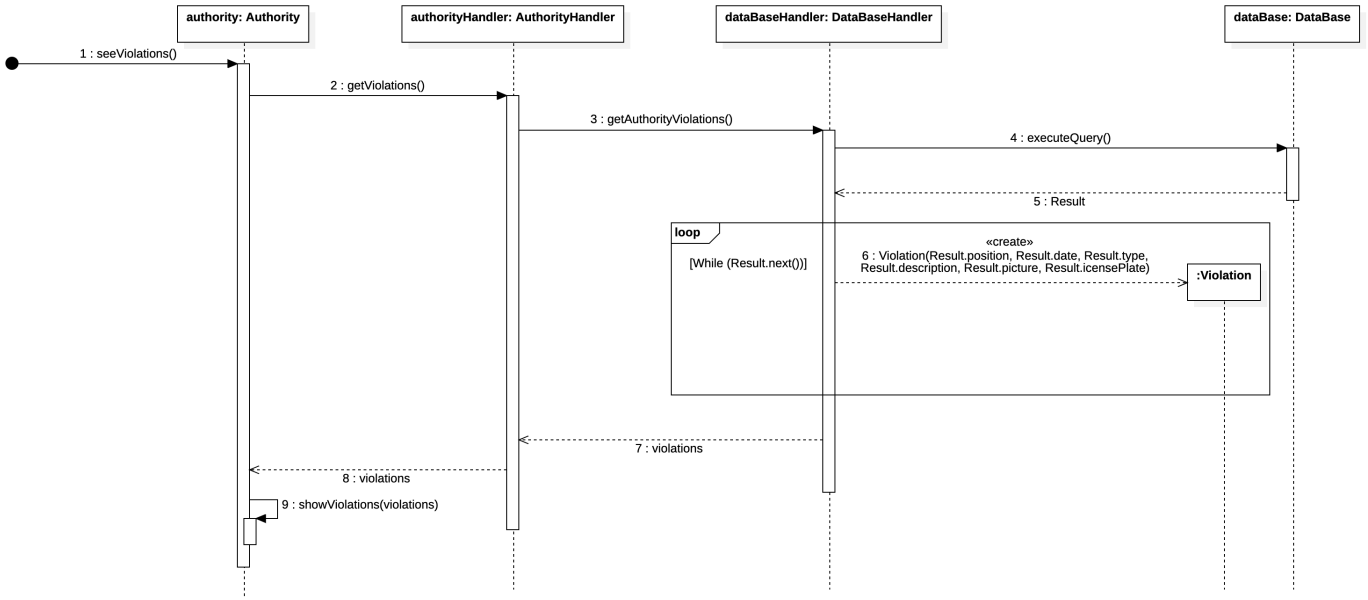

Figure 7: Sequence Diagram 3

In this third sequence diagram is shown the situation in which the authority wants to see all the violations sent by all the end users. In particular, the AuthorityApplication sends the request to the AuthorityHandler, which then asks for them to the DataBaseHandler. The latter executes the query on the data base and when it gets the results it uses them to create a list of violations which is returned back to the AuthorityHandler and then to the AuthorityApplication which is in this way able to show them to the authority.

Sequence Diagram 4: An authority user is notified for a near violation

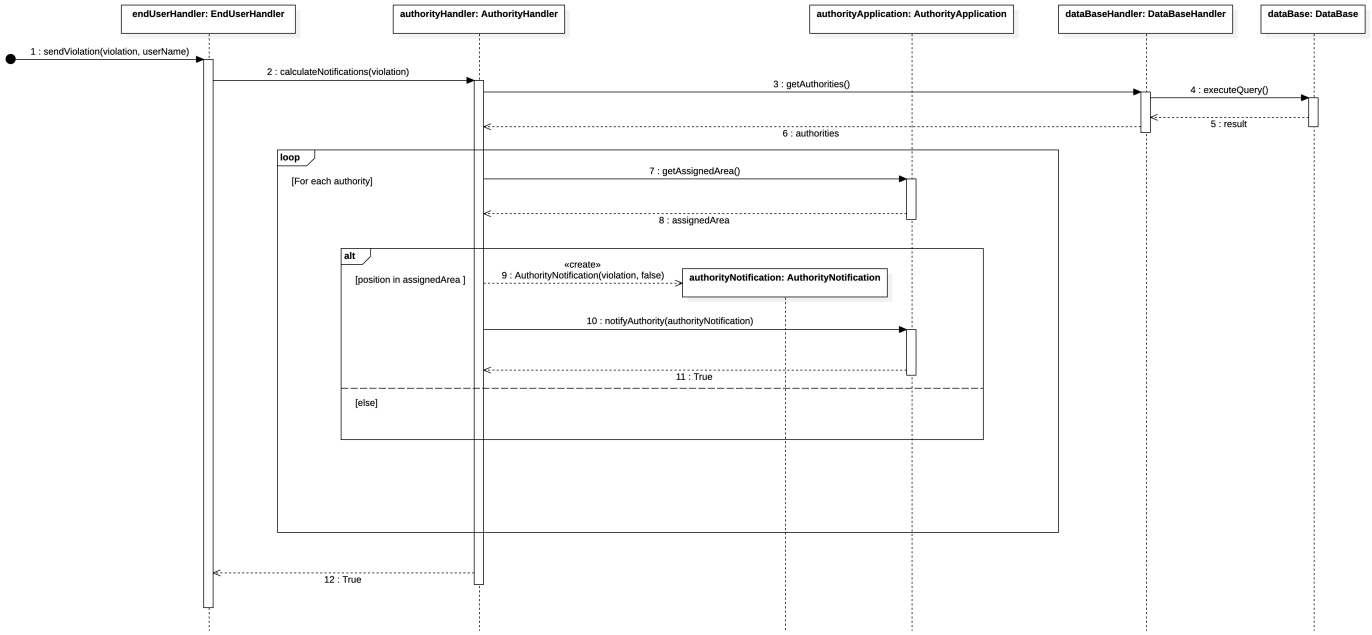

Figure 8: Sequence Diagram 4

In this fourth sequence diagram is shown the situation in which SafeStreets receives a new violation and consequently it notifies the authorities to which is assigned the area in which the violation has occurred. In particular, as soon as the violation is sent to the DataBaseHandler, the EndUserHandler asks to the AuthorityHandler to calculate the set of authorities which have to be notified for that violation by checking if the position of it is in their assigned area. So, the AuthorityHandler requests to the DataBaseHandler the list of all the authorities, and then it calculates the set of them to notify. Finally, the AuthorityHandler notifies all the AuthorityApplications in the calculated set, which can then show the received data directly to the user.

Sequence Diagram 5: An authority user notifies other authorities

This sequence diagram shows the flow of operations that occur when an authority decides to go to verify a traffic violation. Firstly, the authority creates an AuthorityNotification that contains the violation and a boolean that means that the authority is going to verify it. Then, the authority sends to the AuhtorityHandler his intention. So, the AuthorityHandler starts to contact each authority, ask them their assigned area, and, if the violation is in an authority's area, it warns that authority that the violation is going to be checked.

Sequence Diagram 6: A municipality user sends accidents data

The municipality user decides to start to collaborate with SafeStreets. So he takes all the accidents occured under his jurisdiction over the years and sends them to the MunicipalityUserHandler. The MunicipalityHandler then

sends the accidents data to the DatabaseJandler and they are inserted in the database.

Sequence Diagram 7: A municipality user requests the unsafe areas and interventions

When the municipality user wants to see the unsafe areas in his territory, he has to ask them to the system. The MunicipalityHandler takes all data referring to accidents and violations in the municipality jurisdiction and then elaborates the unsafe areas. Finally, it calculates also the possible interventions and it sends unsafe areas and suggestions to the municipality user.

Sequence Diagram 8: An user sees the statistics

This sequence diagram shows how an end user can ask the statistics. This diagram is valid also for the authority user and the municipality user (obviously changing "endUser" with the type of user). The user asks the statistics to the system. The EndUserHandler sends the request to the StatisticsHandler, who takes the data regarding the violations from the database (thanks to the DataBaseHandler). Then, the statisticsHandler elaborates the statistics and create a statistic. This is sent to the EndUserHandler and then to the EndUser.

2.5 COMPONENT INTERFACES

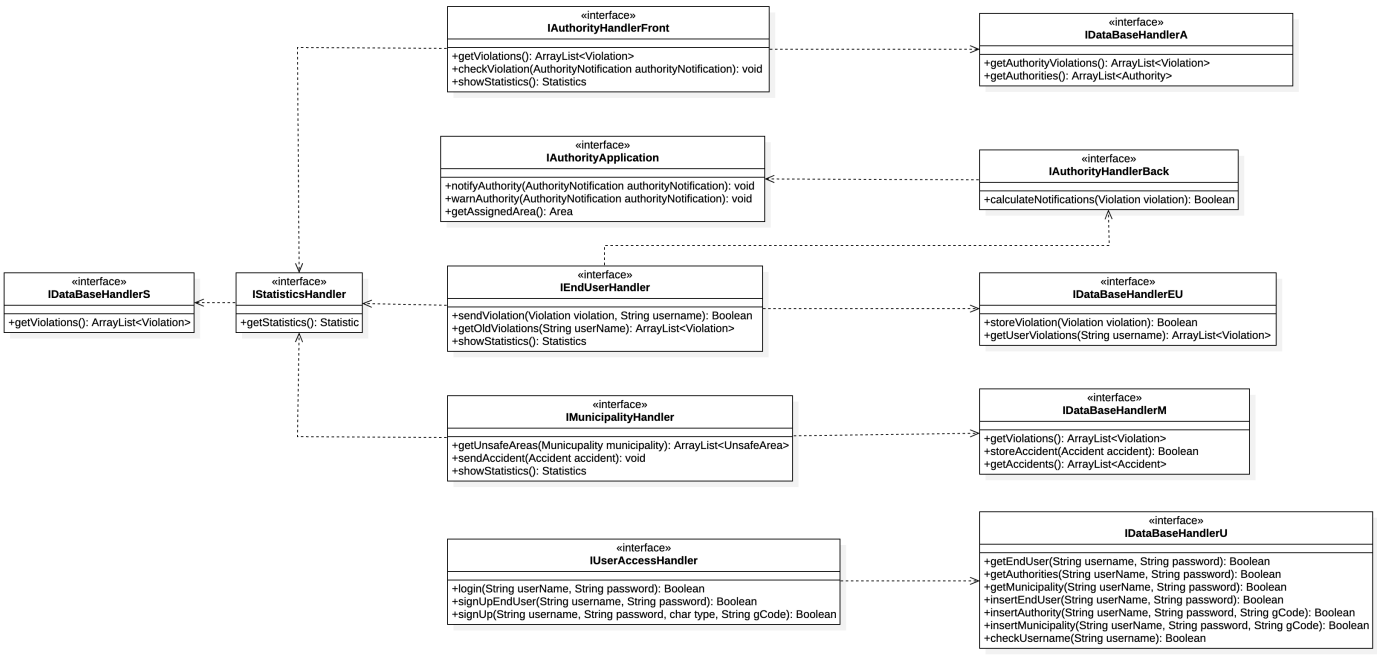

Figure 9: Component Interfaces

Here the interfaces shown before in the component diagram are explained in details. In particular, each interface is treated individually, exposing

which methods it exports and why it exports those methods. Some components expose more than a single interface in order to make available the right functions to the right components, and at the same time avoid that someone uses a function to which it shouldn't have acces. The dashed arrows represent the dependencies between the interfaces.

2.5.1 IAuthorityApplication

The AuthorityApplication is the only client component which exposes an interface. This is necessary because SafeStreets must be able to notify authorities when a violation occurs in their assigned area, when another authority checks a violation and also to get the assigned area of the authority. In particular, it exports three methods:

notifyAuthority(AuthorityNotification authorityNotification): void

This method takes as parameter an AuthorityNotification and does not return any value. The aim, as the name suggests, is to notify the authority of the occurrence of a traffic violation in his assigned area.

warnAuthority(AuthorityNotification authorityNotification): void

This method takes as parameter an AuthorityNotification and does not return any value. The aim is similar of that of the function explained above, in fact also in this case it is a matter of notifying the authority, but this time the reason is that another authority who has the same assigned area checks the violation, so the system has to warn the other interestd authorities in order to avoid that too much of them are engaged in the same problem.

getAssignedArea(): Area

This method does not take parameters and returns an Area. The aim is to provide to the System the assigned area of the interested authority such that it can calculate the authorities that it has to notify in case of the occurrence of a new violation and those it has to warn in case another authority checks a violation.

2.5.2 IAuthorityHandlerFront

This interface is exposed by the AuthorityHandler component for the usage of the AuthorityApplication. In particular, it is needed for two main reason, the first is to mine information from SafeStreets and the second is to warn the system that an authority checked a violation. This interfaces exports three methods:

getViolations(): ArrayList<Violation>

This method does not take parameters and returns an array of Violations. The aim is to request to the system all the violations that have been sent till

that moment, and so it is a first through which the authority user can mine information.

checkViolation(AuthorityNotification authorityNotification): void

This method takes as parameter an AuthorityNotification and does not return any value. The aim is to warn the system that some authority is going to check a violation, such that subsequently it can calculate the set of authorities that are interested by the same violation because the position of the violation falls within their assigned area and inform them of it.

showStatistics(): Statistics

This method does not take parameters and returns a Statistic. The aim is to get the statistics from the SafeStreets and so it represents another way to mine information from the system. In particular, it is provided to the requesting authority a map complete of markers which show all the violations sent till that moment, thus giving a visual idea of the situation of the territory. Moreover, more precise data are given in the form of real statistics.

2.5.3 IAuthorityHandlerBack

This interface is exposed by the Authorityhandler component in order to allow the EndUserHandler component to warn it when an end user sends a new violation to the system. It exports only one method:

calculateNotifications(Violation violation): Boolean

This method takes as parameter a Violation and returns a Boolean. The aim is to calculate the set of authorities that have to be notified for the violation received in input by the function. In particular, after having requested and received the assigned areas of all the authorities it selects only those whose assigned area contains the position of the violation.

2.5.4 IDataBaseHandlerA

In order to complete the description of the interfaces related to the authority user, here is explained the interface exposed by the DataBaseHandler component for the AuthorityHandler component. The task of this interface is to make available to this last mentioned component all the methods that are necessary to retrieve data useful for the authorities. This interface exports two mothods:

getAuthorityViolations(): ArrayList<Violation>

This method does not take parameters and returns an array of Violations. The aim is to get from the data base all the violations received until that moment such that the AuthorityHandler component can then return them to the AuthorityApplication which is in this way able to show them to the authority user.

getAuthorities(): ArrayList<Authority>

This method does not take parameters and returns an array of Authorities. The aim is to get from the data base the list of all the authorities currently registered to the service such that the AuthorityHandler component can the calculate both the set of authorities that have to be notified for a new violation and the set of authorities that have to be warned of the fact some authority has checked a violation.

2.5.5 IEndUserhandler

This interface is exposed by the EndUserHandler for the EndUserApplication in order to provide to the end user the three main functionalities to which he can have access: send a new violation, see the his own old violations and see statistics. The IEndUserhandler interface exports three methods:

sendViolation(Violation violation, String username): Boolean

This method takes as parameters a Violation and a String representing the username of the end user and returns a Boolean. The aim is to allow the user to send a new violation to the system, such that the system can notify the authorities of it and store it in order to calculate statistics.

getOldViolations(String userName): ArrayList<Violation>

This method takes as parameter a String representing the username of the end user and returns an array of Violations. The aim is to make possible to the end user to request all the old violations that he has sent in the past.

showStatistics(): Statistics

This method does not take parameters and returns a Statistic. The aim is to get the statistics from the SafeStreets and so it represents a way to mine information from the system. In particular, it is provided to the requesting end user a map complete of markers which show all the violations sent till that moment, thus giving a visual idea of the situation of the territory. Moreover, more precise data are given in the form of real statistics.

2.5.6 IDataBaseHandlerEU

This interface is exposed by the DataBaseHandler for the Enduserhandler in order to make it possible to store violations sent by the end users in the data base and to get from the data base all the violation sent by an end user. This interface exports two methods:

storeViolation(Violation violation): Boolean

This method takes as parameter a Violation and returns a Boolean. The aim is to store a violation just sent by an end user in the data base, such that then it is possible to retrieve it and also to use it to calculate statistics.

getUserViolations(String username): ArrayList<Violation>

This method takes as parameter a String representing the username of the end user and returns an array of Violations. The aim is to retrieve all the violations sent by an end user in order to make it possible to show them to him.

2.5.7 IMunicipalityHandler

This interface is exposed by the MunicipalityHandler component for the MunicipalitySoftware component. It allows to get the unsafe areas, to send accidents data and to get the statistics. It exposes three methods:

getUnsafeAreas(Municupality municipality): ArrayList<UnsafeArea>

This method takes as parameter a Municipality and returns an array of UnsafeAreas. The aim is to calculate the set of unsafe areas that characterize the territory under the jurisdiction of the municipality. It also calculates the best interventions that can be applied in order to improve the situation, thus giving suggestions.

sendAccident(Accident accident): void

This method takes as parameter an Accident and does not return anything. The aim is to make it possible to the MunicipalitySoftware component to sent data about accidents, such that then the system can save them for the elaboration of the most unsafe areas of the territory.

showStatistics(): Statistics

This method does not take parameters and returns a Statistic. The aim is to get the statistics from the SafeStreets and so it represents a way to mine information from the system. In particular, it is provided to the requesting municipality a map complete of markers which show all the violations sent till that moment, thus giving a visual idea of the situation of the territory. Moreover, more precise data are given in the form of real statistics.

2.5.8 IDataBaseHandlerM

This interface is exposed by the DataBaseHandler for the Municipality-Handler in order to make possible to get violations and to store data about accidents. In particular, it exports three methods:

getViolations(): ArrayList<Violation>

This method does not take parameters and returns an array of Violations. The aim is to get from the data base all the violations received until that moment such that the MunicipalityHandler component which is in this way able to use them to calculate the most unsafe areas of the territory of the municipality and also show them to the municipality user.

2.5.9 storeAccident(Accident accident): Boolean

This method takes as parameter an Accident and returns a Boolean. The aim is to store in the data base data about accidents sent by a municipality such that then they can be used to calculate the most unsafe areas and also the suggestion related to them.

2.5.10 getAccidents(): ArrayList<Accident>

This method does not take parameters and returns an array of Accidents. The aim is to get from the data base all the accidents data received until that moment such that the MunicipalityHandler component which is in this way able to use them to calculate the most unsafe areas of the territory of the municipality and also show them to the municipality user.

2.5.11 IUserAccessHandler

This interface is exposed by The UserAccessHandler component in order to make possible to each user of each type to log in and register to the application. In particular, it exports three methods:

login(String userName, String password): Boolean

This method takes as parameters two Strings, one defining the username of the user and the second defining its password to access its account and returns a Boolean. The aim is to make possible to each user to access its account everywhere.

signUpEndUser(String username, String password): Boolean

This method takes as parameters two Strings, one defining the username of a new user and the second defining its chosen password to register to the service and returns a Boolean. The aim is to make it possible to a new user to register himself to SafeStreets in the role of end user. The method also checks that the chosen username in not already in use by another user. In case the chosen username is already in use the function returns false, thus making the user select another usrname.

2.5.12 signUp(String username, String password, char type, String gCode): Boolean

This method takes as parameters three Strings, one defining the username of a new user, the second defining its chosen password to register to the service and the third defining a governmental code. Then it takes as parameter also a character defining the role with which the user wants to register and then returns a Boolean. The aim is to make possible to a new user to register to the SafeStreets service in the role of authority or municipality. In order to prove that the new user really covers the chosen role he must provide the

governmental code which will then be verified by SafeStreets. If the code is not correct the function returns false, thus making the user to reenter it.

2.5.13 IDataBaseHandlerU

This interface is exposed by the DataBaseHandler component in order to make possible to the UserAccessHandler component to verify the identity of a user and to register a new user. In particular, it exports six methods:

getEndUser(String username, String password): Boolean

This method takes as parameters two Strings, one defining the username of an end user and the second defining its password and then returns a Boolean. The aim is to verify the identity of a registered end user in order to allow him to log in to his account by verifying the presence of a tuple with the inserted credentials in the data base. If the credentials are correct the function returns true, otherwise false.

getAuthorities(String userName, String password): Boolean

This method takes as parameters two Strings, one defining the username of an authority user and the second defining its password and then returns a Boolean. The aim is to verify the identity of a registered authority user in order to allow him to log in to his account by verifying the presence of a tuple with the inserted credentials in the data base. If the credentials are correct the function returns true, otherwise false.

getMunicipality(String userName, String password): Boolean

This method takes as parameters two Strings, one defining the username of a municipality user and the second defining its password and then returns a Boolean. The aim is to verify the identity of a registered municipality user in order to allow him to log in to his account by verifying the presence of a tuple with the inserted credentials in the data base. If the credentials are correct the function returns true, otherwise false.

insertEndUser(String userName, String password): Boolean

This method takes as parameters two Strings, one defining the username of a municipality user and the second defining its password and then returns a Boolean. The aim is to insert the credentials of the new end user in the data base.

insertAuthority(String userName, String password, String gCode): Boolean

This method takes as parameters two Strings, one defining the username of a municipality user, the second defining its password and the third defining a governmental code, and then returns a Boolean. The aim is to insert the credentials of the new authority user in the data base.

insertMunicipality(String userName, String password, String gCode): Boolean

This method takes as parameters two Strings, one defining the username of a municipality user, the second defining its password and the third defining a governmental code, and then returns a Boolean. The aim is to insert the credentials of the new municipality user in the data base.

checkUsername(String username): Boolean

This method takes as parameters a String and returns a Boolean. The aim is to verify if the username chosen by a new user in not already in use. In particular, if the username is in the data base the function returns true thus making the user choose another username, otherwise it returns false.

2.5.14 IStatisticsHandler

This interface is exposed by the StatisticsHandler component in order to meke possible to all the users to get the statistics calculated by SafeStreets. In particular, it exports only one method:

getStatistics(): Statistic

This method does not take parameter and returns a Statistic. The aim is to make possible for all the user to mine data from SafeStreets, in particular to retrieve information as statistics.

2.5.15 IDataBaseHAndlerS

This interface is exposed by the DataBaseHandler component for the StatisticsHandler component in order to retrieve data to build the statistics. In particular, it exports only one method:

getViolations(): ArrayList<Violation>

This method does not take parameter and returns an array of Violations. The aim is to make possible to retrieve all the violations from the data base and return them to the StatisticsHandler component such that it can calculate the statistics.

2.6 SELECTED ARCHITECTURAL STYLES AND PATTERNS

2.6.1 Architectural styles

The entire system is a client-server application, in which the clients are represented by the users' device, while the server is represented by the handlers. The clients are fat clients because they have both the presentation and the application layers (the municipality user also the data layer). The architectural structure is a three tier architecture. The three tiers are: clients, server and

database. The database is separated from the server for having a better management of data, for answering more requests and for guaranteeing a more scalability over the time. The server is composed by different handlers, each of which manages one of the features that characterize the system this way keeping the different functionalities separated and independent. This choice is motivated by the fact that in this way is easier to grant maintainability, scalability and security. The DataBaseHandler exposes different interfaces for each type of user in order to give a first layer of protection against malicious attacks, thus contributing to the security, because in this way each user handler can access only to the functionalities to which he is allowed to access. The end user and the municipality user don't expose any interface because they have only to send something to the system or asking the system for receiving data. The authority user instead exposes an interface because he has to receive from the system notifications without asking for them. Another characteristic is the portability of the system, in fact it works both for iOS and Android execution environment and it can run in the main operating systems.

2.6.2 Design patterns

The main design patterns used are:

- Singleton: all the handlers inside the server are singletons. In this way, for each handler there is only one instance.
- Factory method: this design pattern is used by the EndUserApplication to create the Violation, by the StatisticsHandler to create the Statistic, by the AuthorityApplication to create the AuthorityNotification, by the municipality to create the accidents data and by the MunicipaltyHandler to create the UnsafeAreas.
- Observer: the AuthoritieApplications are the observers of the AuthorityHandler. So each time the authorityHandler needs to warn the authorities, they can be notified.

2.7 OTHER DESIGN DECISIONS

Regarding the communication protocol between the clients and the server the decision fell on adopting the RMI protocol (Remote Method Invocation). This because RMI allows to exploit the advantages given by the object oriented programming also in a distributed setting. By using it, in fact, object references can be passed through the various nodes and moreover the separation between the objects' interfaces and their implementations can be easily kept.

Another design decision that has to be outlined is the choice of using a relational data base. There are many reasons that justify this pick, and the main ones are:

- It is a simple data model which is easier to manage and to implement.

- Reduced data redundancy and high data consistency.
- Quantitative data processing,
- Unified language for queries.

3 USER INTERFACE DESIGN

3.1

4 REQUIREMENTS TRACEABILITY

The previously presented design choices have been made taking into account that the system that has to be built has to fulfill the requirements exposed in the RASD document in order to reach the individuated goals under the assumptions that have been made at first. In order to better explain how the requirements are covered by the design picks, in this chapter is shown primarily a traceability matrix which gives a first visual and fast idea of which requirement is satisfied by which component, and secondly a bulleted list which goes further into details giving also an explanation of how the component satisfies the requirement.

4.1 TRACEABILITY MATRIX

Row ID	Requirements ID	Components
r1	R1	EndUserApplication, EndUserHandler, AuthorityHandler, AuthorityApplication
r2	R2	EndUserApplication
r3	R3	EndUserApplication
r4	R4	EndUserApplication, EndUserHandler
r5	R5	EndUserHandler, AuthorityHandler, AuthorityApplication
r6	R6	EndUserApplication
r7	R7	AuthorityHandler, EnduserHandler
r8	R8	EndUserApplication, EndUserHandler
r9	R9	AuthorityHandler, AuthorityApplication, DataBaseHandler
r10	R10	EndUserApplication, EndUserHandler, AuthorityHandler, AuthorityApplication, MunicipalitySoftware, MunicipalityHandler, StatisticsHandler, DataBaseHandler
r11	R11	StatisticsHandler, DataBaseHandler (forse aggiungere sequence su questo)
r12	R12	MunicipalitySoftware, MunicipalityHandler, DataBAseHandler
r13	R13	MunicipalitySoftware, MunicipalityHandler, DataBaseHandler
r14	R14	EndUserApplication, AuthorityApplication, MunicipalitySoftware, UserAccessHandler, DataBaseHandler
r15	R15	EndUserApplication, AuthorityApplication, MunicipalitySoftware, UserAccessHandler, DataBaseHandler
r16	R16	EndUserApplication, EndUserHandler, AuthorityHandler, AuthorityApplication, MunicipalitySoftware, MunicipalityHandler, StatisticsHandler, DataBaseHandler
r17	R17	MunicipalitySoftware, MunicipalityHandler, DataBaseHandler
r18	R18	AuthorityHandler, AuthorityApplication, DataBaseHandler
r19	R19	AuthorityHandler, AuthorityApplication, DataBaseHandler
r20	R20	AuthorityHandler, AuthorityApplication
r21	R21	AuthorityHandler, AuthorityApplication
r22	R22	EndUserApplication, EndUserHandler, DataBaseHandler
r23	R23	MunicipalitySoftware, MunicipalityHandler, DataBaseHandler

4.2 TRACEABILITY IN DETAILS

Here is shown the list of all the requirements and for each one of them it is explained how design components cover it.

R1 When a user sees a traffic violation, SafeStreets application must allow him to take a picture of it, insert a description and immediately send the information to authorities.

EndUserApplication It allows to take the picture, to insert the description and send the information.

EndUserHandler It allows to send the violation the the Authorityhandler.

AuthorityHandler It calculates the set of authorities that are interested by the sent violation and notifies them.

AuthorityApplication It allows to show the violation to the authority user.

R2 When an end user sends a violation report, SafeStreets application must detect automatically the date, the time and the position from the device. The position is taken from the GPS of the user's device.

EndUserApplication It allows to detect automatically the date, the time and the position from the device.

R3 When detecting the date, the time and the position from the device, if it is not able to take one of these information, SafeStreets application must notify the user telling him which is the problem.

EndUserApplication It allows to tell to the user that it is not possible to detect automatically the date, the time or the position from the device.

R4 When the picture of the violation inserted by the end user is not readable by the application, the system must ask him to insert it again or to write manually the license plate number.

EndUserApplication It tries to read the license plate from the picture and if it fails it asks to the user to insert another picture or to insert the license plate manually.

EndUserHandler It performs a second control on the correctness of the license plate read by the EndUserApplication, and if it finds inconsistencies it request to the end user to redo the process.

R5 When a violation is sent, SafeStreets dispatching software must find the nearest authority users and notify them.

EndUserHandler It forwards the violation to the AuthorityHandler.

AuthorityHandler It first requests the list of all the authorities, then asks to each one of them their assigned area, calculates the set of them whose assigned area contains the position of the violation and finally notifies them.

AuthorityApplication It shows the violation to the authority user as soon as the AuthorityHandler notifies it.

R6 When an end user wants to report a traffic violation and there is no internet connection, SafeStreets software must allow him to save it and send it when internet connection has been restored.

EndUserApplication It stores the violation data in the local device storage and then as soon as the internet connection is restored it sends the violation to the EndUserHandler.

R7 When a violation is reported, SafeStreets must not show the identity of the end user that created it, so that to guarantee anonymity.

EnduserHandler It forwards the violation to the AuthorityHandler without including the identity of the end user which sent it.

AuthorityHandler It shows only data concerning the violation, without the identity of the end user who sent it.

R8 When an end user or a municipality user logs in, SafeStreets must not allow him to see the traffic violations sent by the other end users.

EndUserApplication It does not allow the end user to retrieve all the violations but only those sent by himself.

EndUserHandler It cannot request all the violations to the DataBaseHandler.

R9 When an authority logs in from his device, SafeStreets must allow him to see information about the traffic violations sent by the end users.

AuthorityHandler It provides the method that allows to request to the DataBaseHandler the list of all the violations sent till that moment.

AuthorityApplication It allows the authority user to access to the violations page, thus asking to the AuthorityHandler to retrieve all the violations.

R10 When a user logs in from his device, SafeStreets must allow him to see statistics about the traffic violations.

EndUserApplication It allows the end user to access to the statistics page.

AuthorityApplication It allows the authority user to access to the statistics page.

MunicipalityASoftware It allows the municipality user to access to the statistics page.

EndUserHandler It allows the EndUserApplication to request statistics.

AuthorityHandler It allows the AuthorityApplication to request statistics.

MunicipalityHandler It allows the MunicipalitySoftware to request statistics.

StatisticsHandler It calculates the statistics using data retrieved from the data base.

DataBaseHandler It allows the StatisticsHandler to retireve data from the data base.

R11 When a violation is reported, SafeStreets must update the traffic violation statistics and the unsafe areas information.

DataBaseHandler It allows the StatisticsHandler to retrieve the list of all the violations sent till that moment from the data base.

StatisticsHandler It always keeps the statistics updated. (da riguardare)

R12 **When a municipality user logs in, SafeStreets must allow him to provide information about accidents occurred in its territory.**

MunicipalitySoftware It allows the municipality user to send data about accidents occurred in his territory.

MunicipalityHandler It forwards the accident data sent to the DataBaseHandler.

DataBaseHandler It store accidents data in the data base.

R13 When a municipality user logs in from his device, SafeStreets must allow him to see unsafe areas and possible interventions.

MunicipalitySoftware It allows the municipality user to access to the "Unsafe areas and Suggestions" page.

MunicipalityHandler It calculates the most unsafe areas and possible suggestions in the territory of the requesting municipality and provides these information to the MunicipalitySoftware.

DataBaseHandler It retrieves data regarding accidents and violations from the data base.

R14 When a user registers to SafeStreets application, the system must ask him to select the role he held (end user, authority or municipality), such that it is possible to distinguish him and provide him the features associated with his role. In particular, if the user select “Authority” or “Municipality” the system must ask him also the governmental code and verify their identity through the provided interface.

EndUserApplication It allows a new end user to register to the service.

AuthorityApplication It allows a new authority user to register to the service.

MunicipalitySoftware It allows a new municipality user to register to the service.

UserAccessHandler It checks if the username is not already in use, checks the governmental code in case the user wants to register as authority or municipality and then in case all the information are correct it forward them to the DataBaseHAndler.

DataBaseHandler It stores the credentials of the new user in the data base.

R15 When a user logs in, SafeStreets must recognise him and his role (end user, authority or municipality), such that to provide him the right features.

EndUserApplication It allows an end user to log in to the service.

AuthorityApplication It allows an authority user to log in to the service.

MunicipalitySoftware It allows a municipality user to log in to the service.

UserAccessHandler It checks if the inserted credentials are correct sending a request to the DataBaseHandler.

DataBaseHandler It queries the data base in order to verify if the inserted credentials are present in the data base.

R16 When a user requests to see traffic violations statistics, but there are too few traffic violation reports, the System must notify him of that.

EndUserApplication It tells to the end user that there are too few information in order to elaborate statistics.

AuthorityApplication It tells to the authority user that there are too few information in order to elaborate statistics.

MunicipalitySoftware It tells to the municipality user that there are too few information in order to elaborate statistics.

EndUserHandler It requests the statistics to the StatisticsHandler for the EndUserApplication.

AuthorityHandler It request the statistics to the StatisticsHandler for the AuthorityApplication.

MunicipalityHandler It request the statistics to the StatisticsHandler for the MunicipalitySoftware.

StatisticsHandler It requests to the DataBaseHandler the list of all the violations and checks if they are sufficient in order to elaborate accurate violations statistics.

DataBaseHandler It retrieves the list of all the violations sent till that moment from the data base.

R17 When a Municipality User requests to see the most unsafe areas and the possible interventions, but there are too few traffic violation data and too few accident data, the System must notify him.

MunicipalitySoftware It tells to the municipality user that there are too few information in order to elaborate unsafe areas and suggestions.

MunicipalityHandler It request the list of all the violations and all the accident data for the territory of the municipality to the DataBaseHandler.

DataBaseHandler It retrieves the list of all the violations and the list of all the accidents data sent till that moment from the data base.

R18 When a violation is reported, SafeStreets must detect the position of all the authorities from their device in order to know who can be interested in knowing the occurrence of the violation.

AuthorityApplication It provides the assigned area of the authority to the AuthorityHandler and then if it receives the violation, it shows it to the authority user.

AuthorityHandler It requests the list of all the authorities to the DataBaseHandler and to each one of them asks for their assigner area. Then it calculates the list of all the authorities that have to be notified by checking if the position of the violation is inside the assigned area of the authority.

DataBaseHandler It retrieves the list of all the authorities from the data base.

R19 When an authority user is notified, SafeStreets software must allow him to warn other authorities that have received the same notification that he is going to check the violation so that not too many authorities deal with the same violation.

AuthorityApplication It provides the assigned area of the authority to the AuthorityHandler and then if it receives the warning for the violation being checked, it notifies the authority user.

AuthorityHandler It requests the list of all the authorities to the DataBaseHandler and to each one of them asks for their assigner area. Then it calculates the list of all the authorities that have to be warned by checking if the position of the violation is inside the assigned area of the authority.

DataBaseHandler It retrieves the list of all the authorities from the data base.

R20 When an authority checks a violation, the System must not allow other authorities to check it and authorities must no longer be able to see it as a notification (with "check a violation" it is meant that the authority expresses to the System the willingness of going verify it in person).

AuthorityApplication It deletes all the AuthorityNotifications that have been checked by other authorities.

AuthorityHandler When an authority wants to check a violation it requests the list of all the checked violations and verifies if that violation is not already checked.

R21 When an authority is warned about a violation checked by another authority, the System must not allow him to check the violation again.

AuthorityApplication It deletes all the AuthorityNotifications that have been checked by other authorities.

AuthorityHandler When an authority wants to check a violation it requests the list of all the checked violations and verifies if that violation is not already checked.

R22 SafeStreets must store all the traffic violations sent by each end user, so that it can show to an end user (only) own contributions.

EndUserApplication It alllws to the end user to access the page "My Violations".

EndUserHandler It requests the list of all the violations sent by a single end user to the dataBaseHandler.

DataBaseHandler It retrieves the list of all the violations sent till that moment by that end user from the data base.

R23 SafeStreets must not allow a municipality to see the unsafe areas information of other municipalities.

MunicipalitySoftware It allows the user to access the page "Unsafe Areas and Suggestions".

MunicipalityHandler It request the list of all the violations and the list of all the accidents data which concern the territory of the municipality.

DataBaseHandler It retrieves the list of all the violations and the list of all the accidents data sent till that moment from the data base concerning the territory under the jurisdiction of the municipality.

5 IMPLEMENTATION, INTEGRATION AND TEST PLAN

6 EFFORT SPENT

6.1 SAMUELE MOSCATELLI

- 14/11/2019: 2h
- 16/11/2019: 4h 30min
- 17/11/2019: 3h
- 18/11/2019: 30min
- 19/11/2019: 2h 15min
- 20/11/2019: 1h 15min
- 28/11/2019: 4h 30min

Total: 18h

6.2 ANDREA POZZOLI

- 11/11/2019: 1h
- 15/11/2019: 2h 30min
- 16/11/2019: 4h 30min
- 17/11/2019: 5h 15min
- 19/11/2019: 1h

Total: 14h 15min

7 REFERENCES