SAMUELE MOSCATELLI, ANDREA POZZOLI

SAFESTREETS - DD

Version 2

15th December 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 characterise 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. Also the data exchanged by the components, the association between components and requirements, and the flow of work for implementing, integrating and testing the system are presented. 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 stores it together with the other data provided by the user and the position, date and time. So, in the example previously shown, SafeStreets would memorise 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 contribution 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. In order to maintain the privacy of the end users, the authorities cannot see who sent the violation. The authorities notified can also decide to go and check directly the infringement, so, in order to avoid that more authorities go to verify 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
- DD - Design Document

1.3.3 Abbreviations

- Rn: n-functional requirement.
- rn: n-row of the matrix.

1.4 REVISION HISTORY

The first version of the document has been released on 9th December 2019.

This is the second version of the document. Changes with respect to the first version:

- Ortographic errors corrections.
- Corrections in section 2.5: correspondences between the diagram and the text.

1.5 REFERENCE DOCUMENTS

- SafeStreets Mandatory Project Assignment
- Software Engineering 2 course slides

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 very high level overview of the application, that explains the structure and the architecture of the system. Then the document gets more into architectural details of the system, showing and explaining the architectural components and their distribution with respect to each other and to the hardware resources; also the data exchanged between the components are presented. In order to do so, appropriate UML diagrams 1 (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 in the design 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. So, several mockups are shown and described.

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. All the requirements written in the RASD document are associated to specific components and all the components satisfy specific requirements.

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

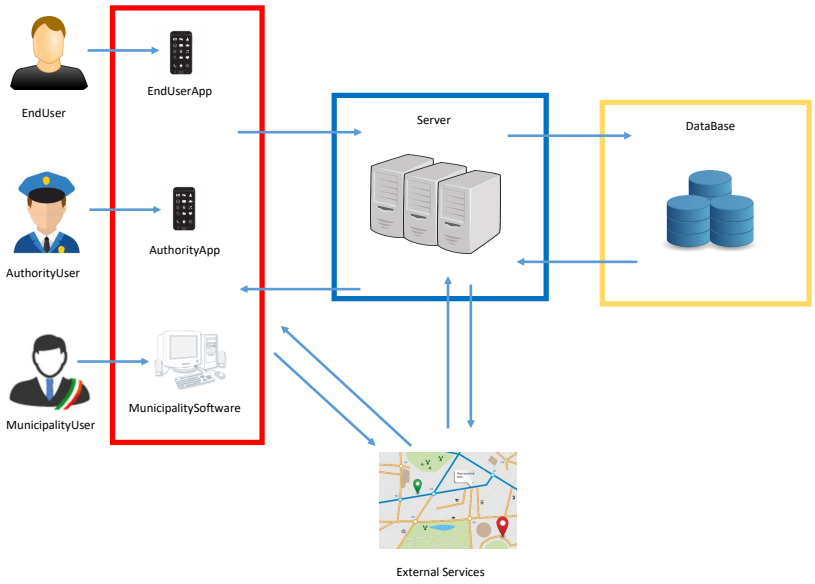

Figure 1: Overview Diagram 2

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 3, 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

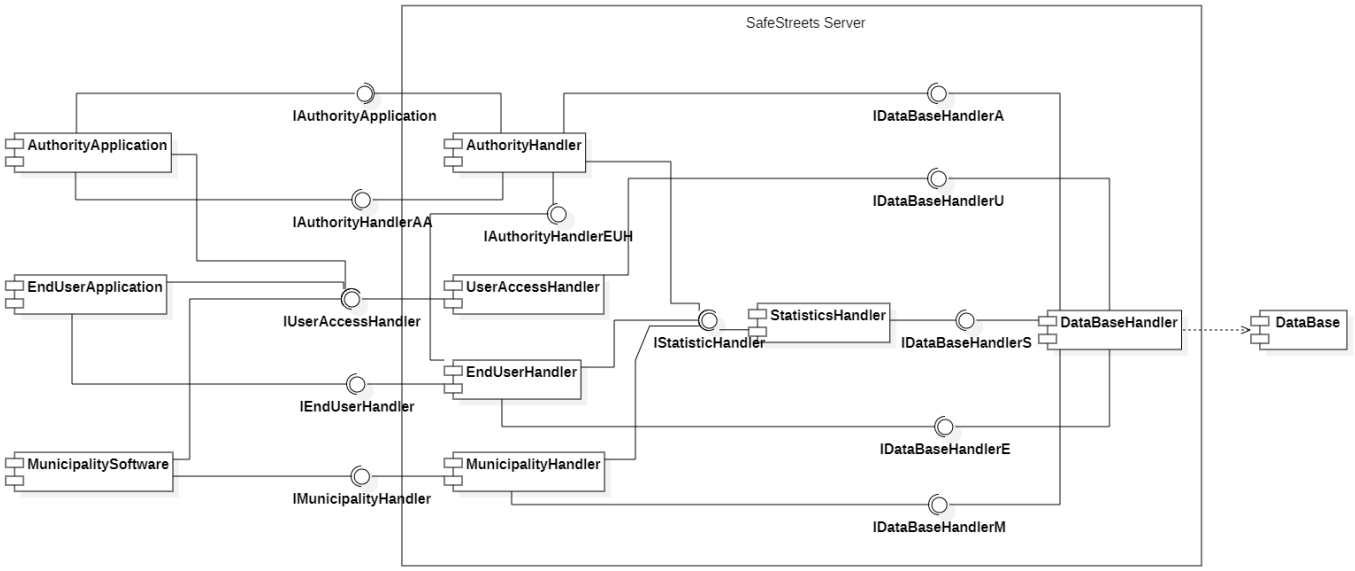

Figure 2: Component Diagram

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 IAuthorityHandlerAA 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. Moreover, this component exposes also the IAuthorityHandlerEUH interface that is used by the EndUserHandler to notify the AuthorityHandler that a new violation has been reported.

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 an interface for each user handler, in particular the IDataBaseHandlerA for the AuthorityHandler component, the IDataBaseHandlerEU for the EndUserHandler component and the IDataBaseHandlerM for the MunicipalityHandler component. Moreover, it also exposes the IDataBaseHanlderU interface for the UserAccessHandler component and the IDataBaseHandlerS for the StatisticsHandler component. In this way it keeps separated the functionalities to which each component can access.

2.2.4 Database

The DataBase component represents the third tier, the one which contains all the data and depicts the data layer.

2.2.5 Class Diagram

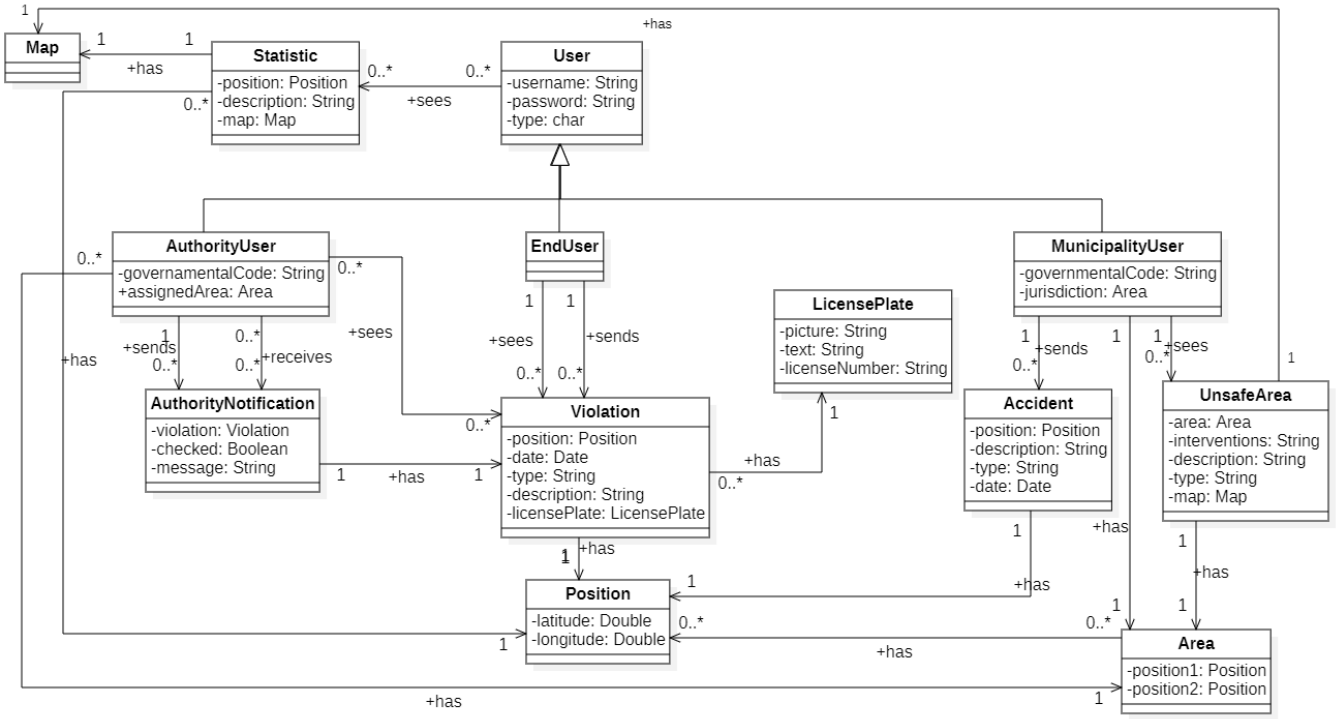

Figure 3: Class Diagram

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 Statistic 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 Map 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 vehicle that commits the traffic violation. It has an attribute picture that is the photo of the vehicle taken by the end user (the photo is encoded and stored as String), and two attributes of type string, one for the eventually 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. Finally, it has an attribute map that is necessary to show the unsafe areas on the OpenStreetMap.

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 some attributes that are the type, the description and the date 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.

Map

The Map class is the representation of the OpenStreetMap map that is necessary to show the statistics to the users and the unsafe areas to the municipality. Details regarding the map aren't shown because OpenStreetMap is an external service.

2.3 DEPLOYMENT VIEW

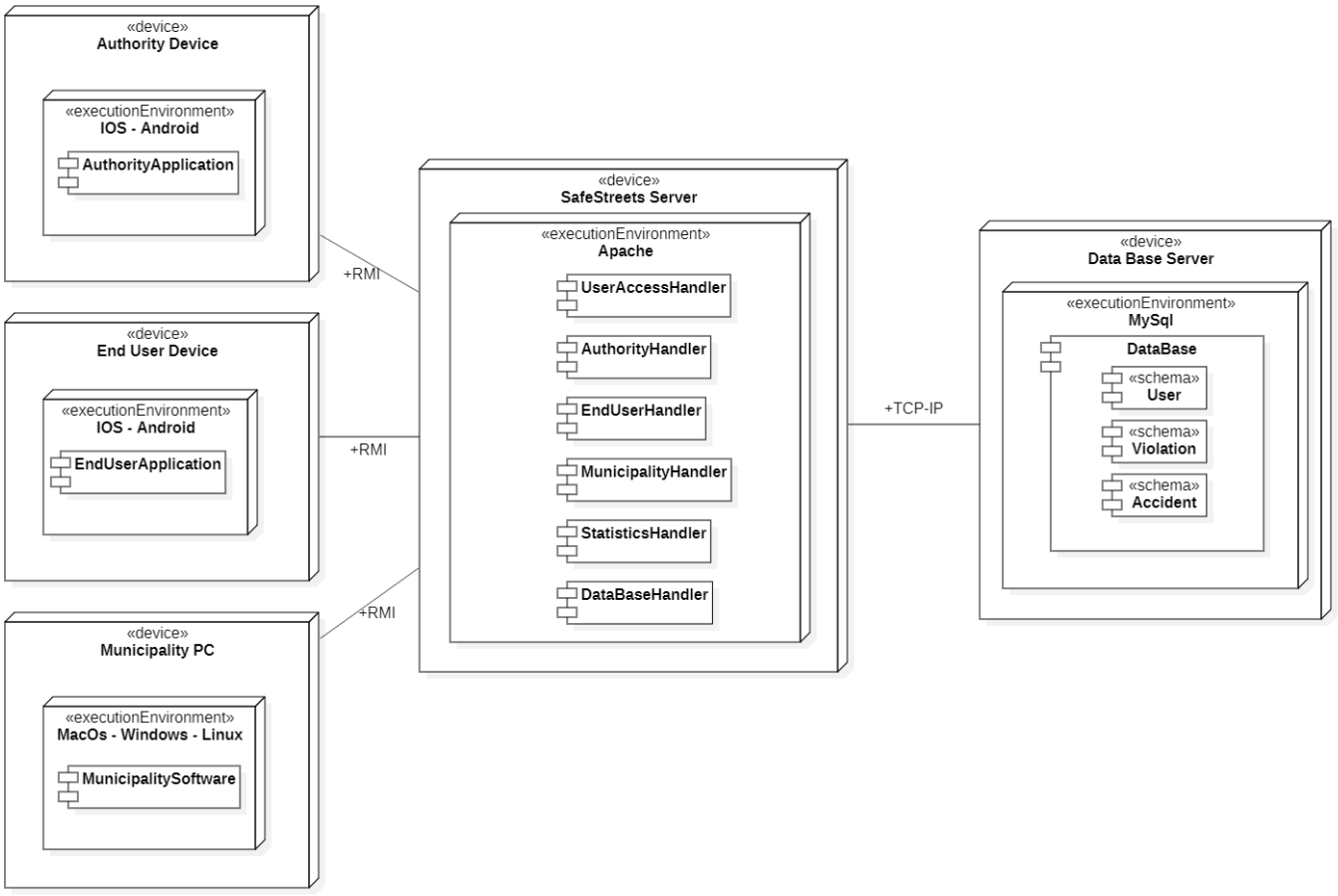

Figure 4: Deployment Diagram

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. In order to work correctly, the device must have a GPS sensor.

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 environment 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 Municipality PC is the computer from which the municipality user accesses 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 environment is mySQL and the database contains some schemes, one for the users data, one for the accidents data, one for the violations data (there is also a column referred to the checking by the authorities).

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

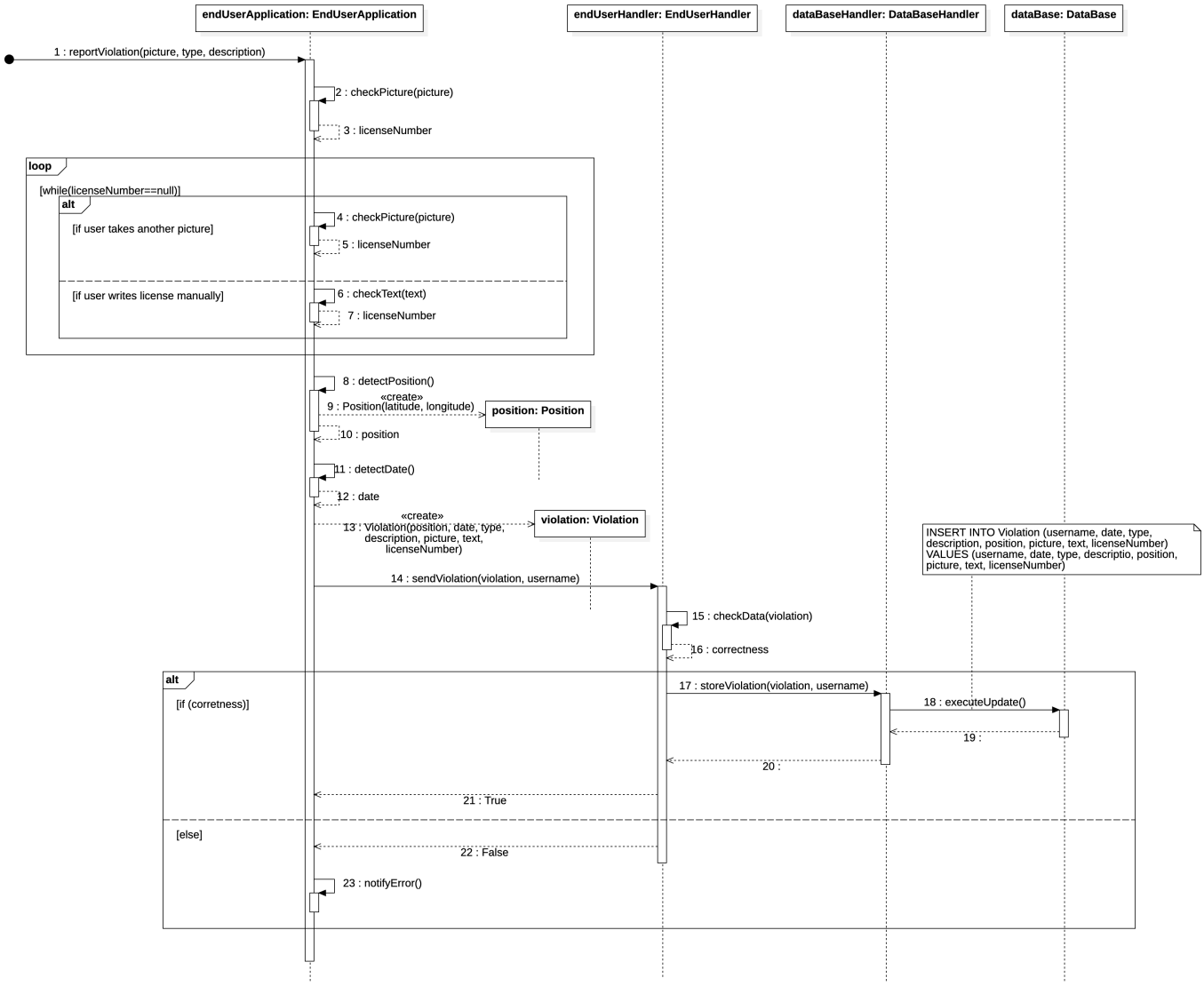

Figure 5: Sequence Diagram 1

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 not readable, the application asks to the end user to insert another picture or to insert the license manually. This is done until the license plate can be read. Instead, 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).

Sequence Diagram 2: An end user sees his past contributions

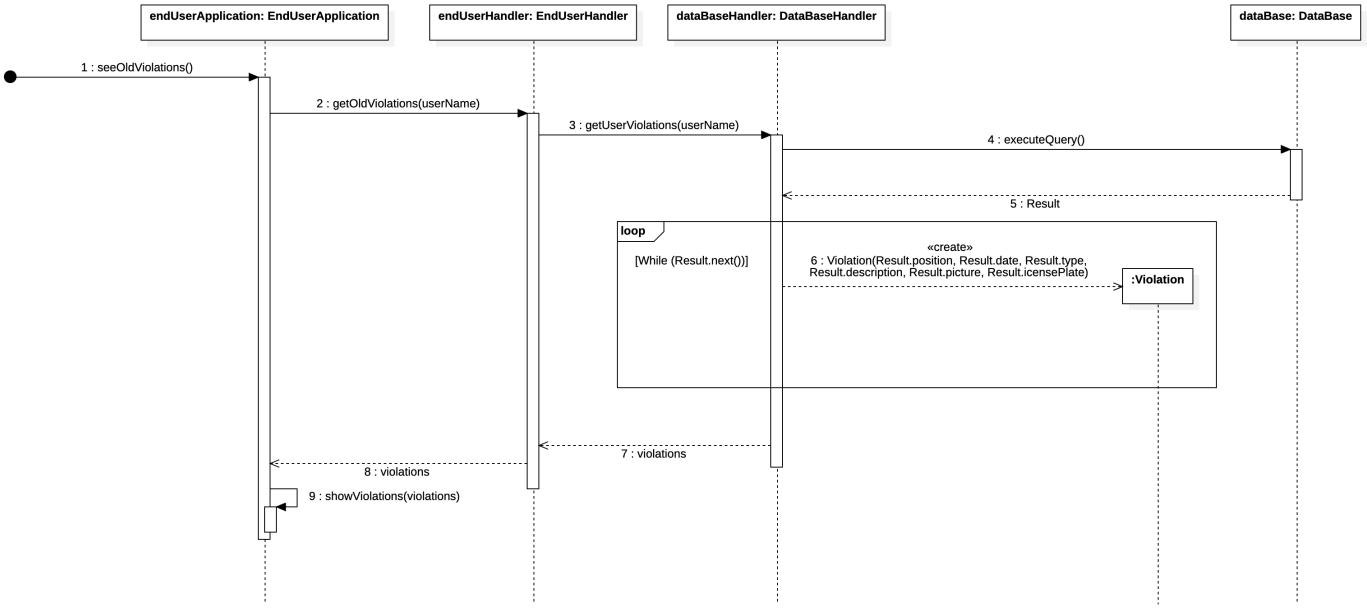

Figure 6: Sequence Diagram 2

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

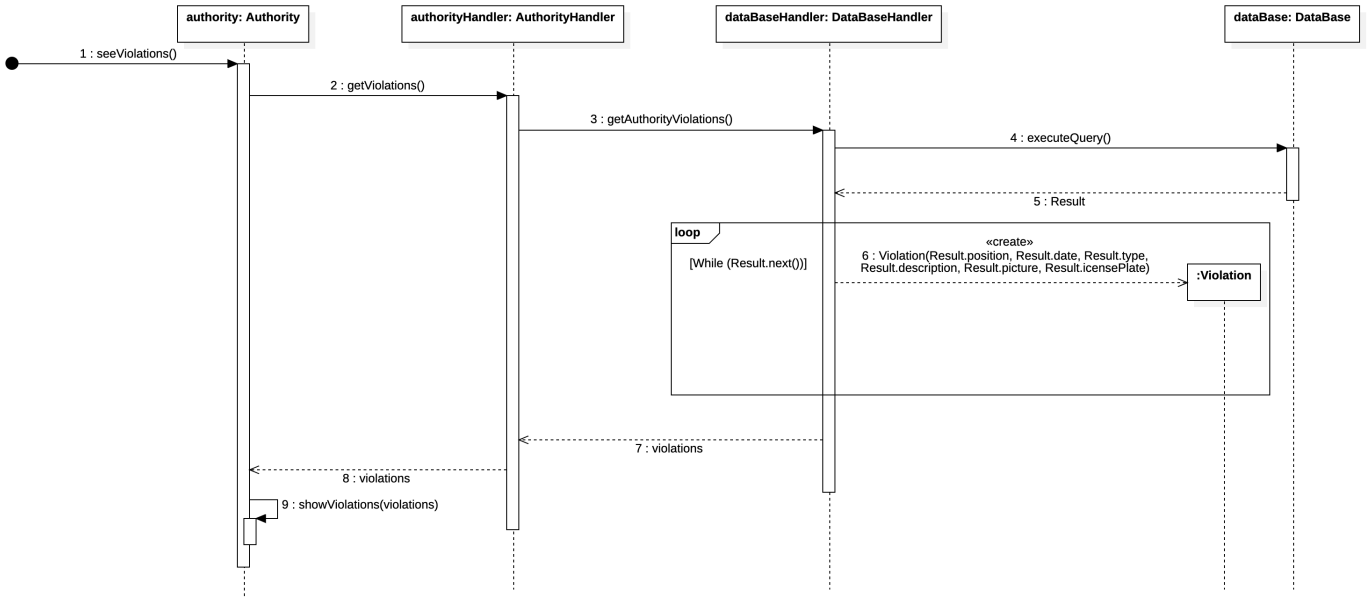

Figure 7: Sequence Diagram 3

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 violation

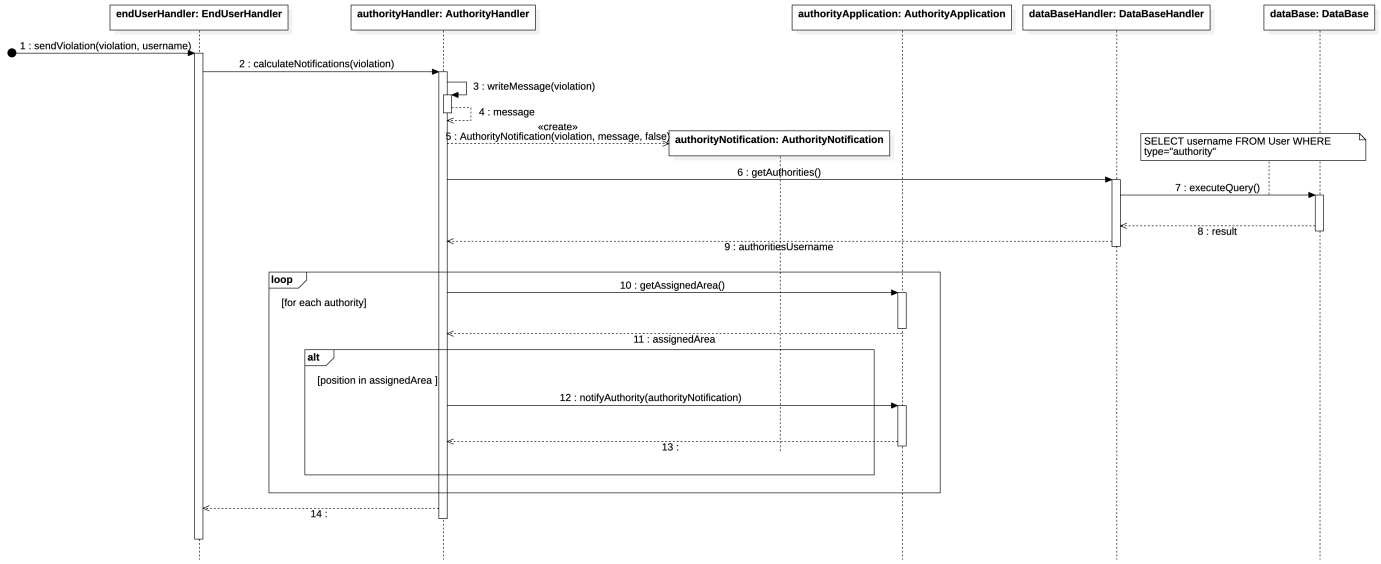

Figure 8: Sequence Diagram 4

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 firstly create the message that has to be associated to that violation (the message refers to the fact that the notification can both be a new violation from an end user or a warning from another authority for the checking of a violation). Then, it creates the AuthorityNotification and requests to the DataBaseHandler the list of all the authorities. Subsequently, it calculates the set of them to notify by asking to each one his assigned area and verifying if the position of the violation is contained in that area. 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

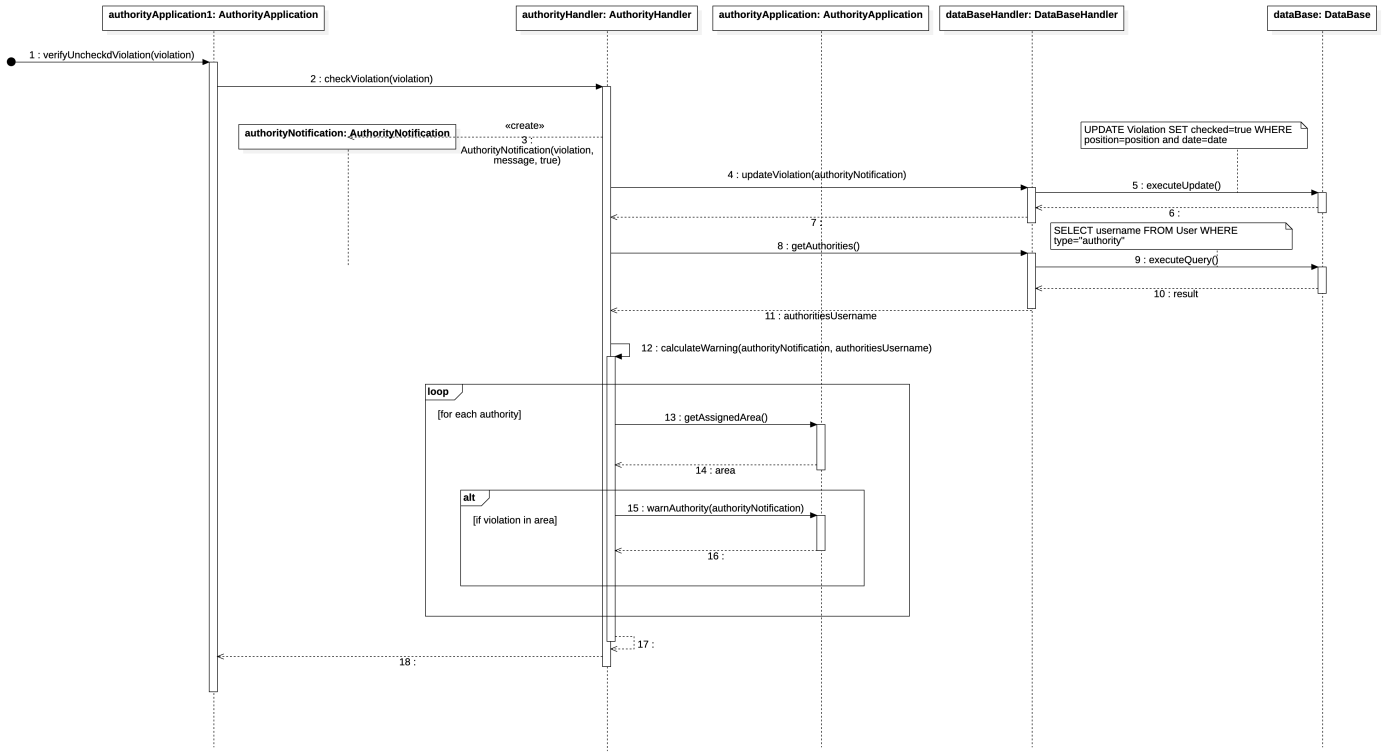

Figure 9: Sequence Diagram 5

This sequence diagram shows the flow of operations that occur when an authority decides to go to verify a traffic violation. The AuthorityApplication communicates to the AuthorityHandler its intention. The latter creates an AuthorityNotification that contains the violation, a message indicating the type of notification and a Boolean that means that the authority is going to verify it. At this point, the AuthorityHandler forwards the just created AuthorityNotification to the DataBaseHandler, which can in this way update the data base, and subsequently asks to it the list of all the registered authorities. In this way, the AuthorityHandler can contact each one of them singularly in order to be able to know which of them have to be notified of the fact that another authority is going to check a violation that has occurred in their assigned area. Each time an interested authority is found it notifies him.

Sequence Diagram 6: A municipality user sends accidents data

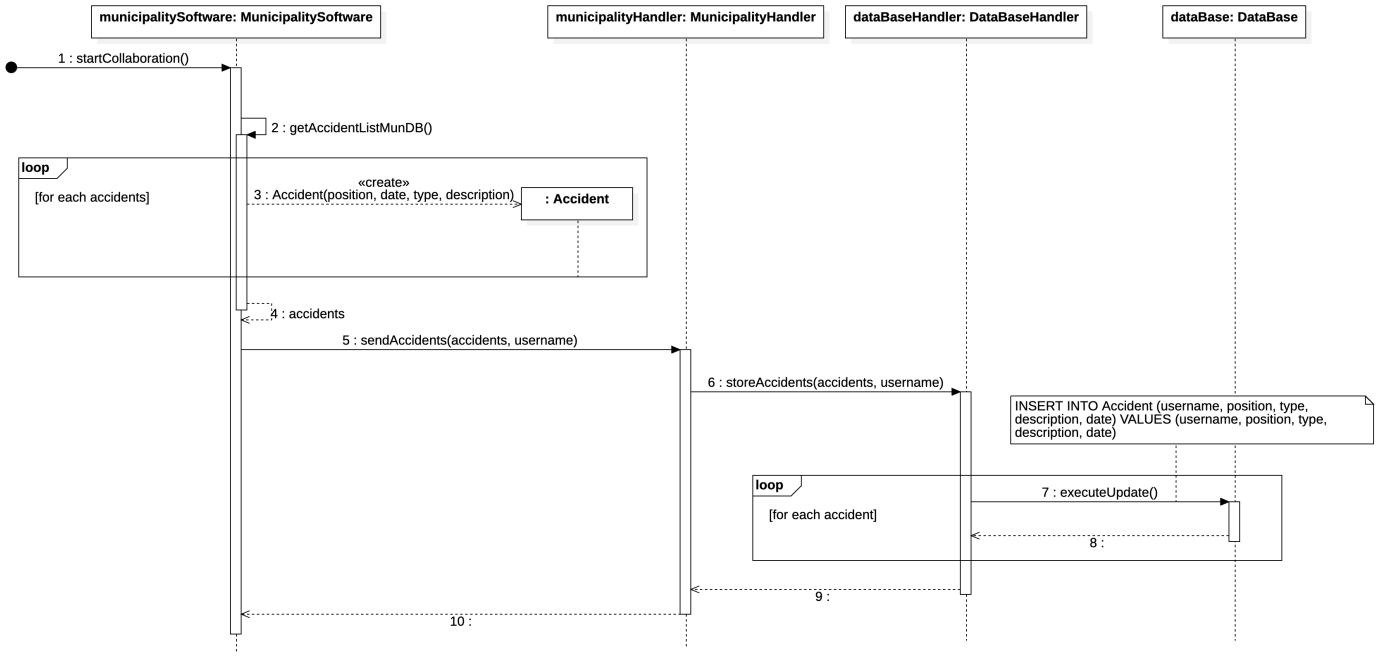

Figure 10: Sequence Diagram 6

The municipality user decides to start to collaborate with SafeStreets. So he inserts all the accidents occurred under his jurisdiction over the years in the application, which creates an array of Accidents and sends it to the MunicipalityHandler. The MunicipalityHandler then sends the accidents data to the DatabaseHandler which inserts them in the database.

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

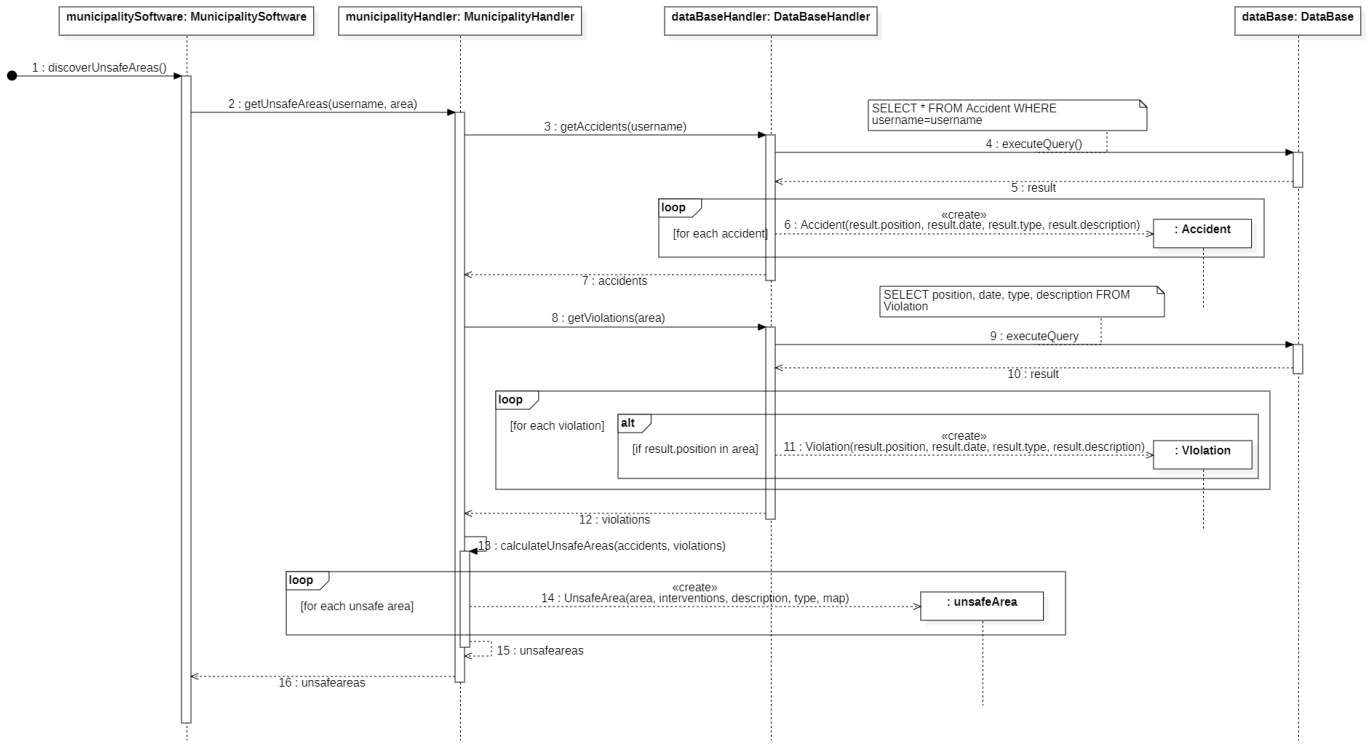

Figure 11: Sequence Diagram 7

When the municipality user wants to see the unsafe areas in his territory, he has to ask them to the system. The MunicipalityHandler firstly asks for the list of accidents occurred in the municipality jurisdiction and then asks for the list of violations occurred in the same territory as before. The DataBaseHandler retrieve these data from the data base, builds a list of Accidents and a list of Violations and send them back to the MunicipalityHandler. At this point, the MunicipalityHandler calculates the unsafe areas and the suggestions related to the territory of the municipality. When it finishes this task it returns the created list of UnsafeAreas to the MunicipalitySoftware, which can then show them to the municipality user.

Sequence Diagram 8: An user sees the statistics

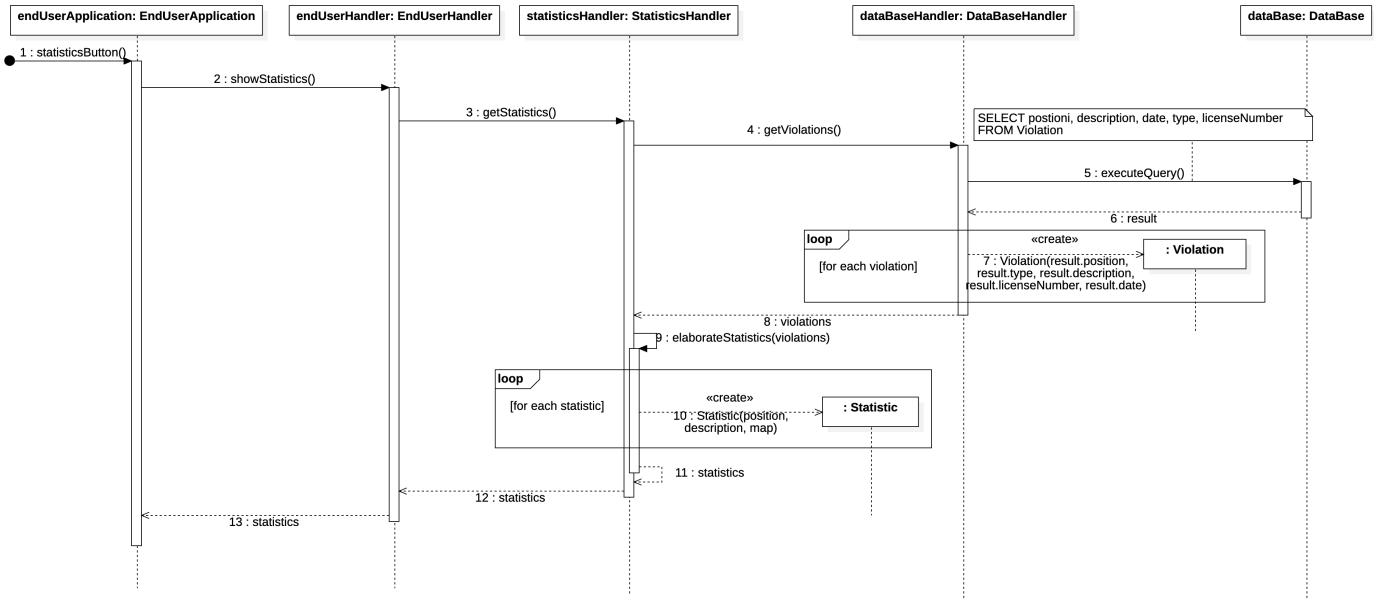

Figure 12: Sequence Diagram 8

This sequence diagram shows how an end user can ask the statistics. Since the mechanism for requesting them is the same also for the authority users and for the municipality users, this situation is shown only once for the end users, but can be easily adapted for all the types of user. The user asks the statistics to the EndUserHandler, which forwards the request to the StatisticsHandler. The StatisticsHandler asks to the DataBaseHandler for the list of all the violations stored in the data base. The DataBaseHandler retrieves them from the data base and returns them to the StatisticsHandler, which then elaborates the statistics and sends them to the EndUserHandler, which can then show them to the end user.

Sequence Diagram 9: An End User logs in

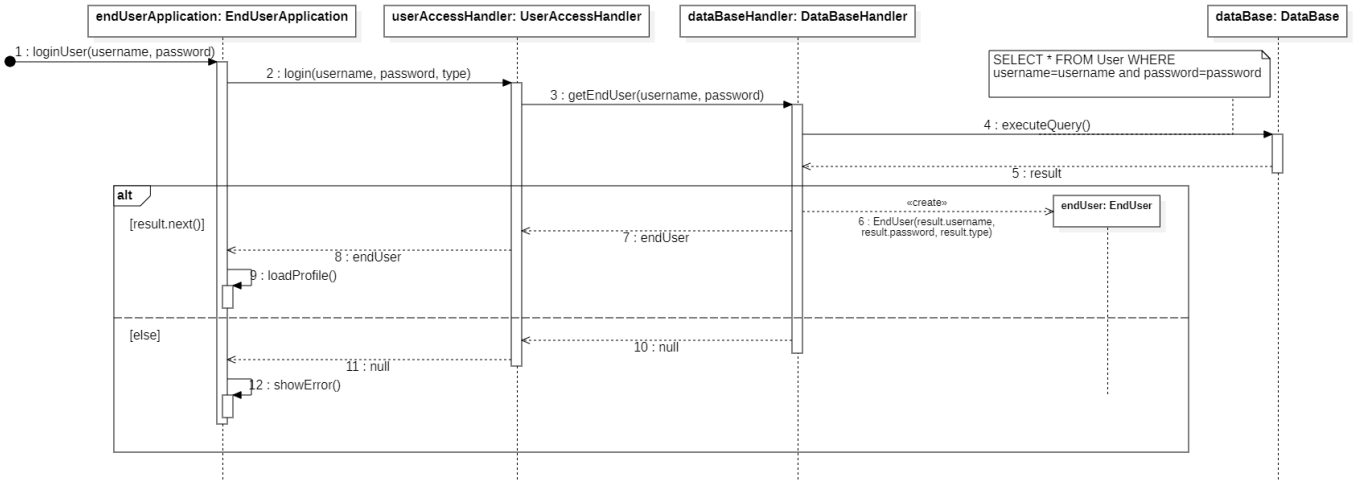

Figure 13: Sequence Diagram 9

For completeness, in this ninth sequence diagram is represented the procedure through which an end user logs in to the application (the same procedure is adopted in case of authority user and municipality user). As soon as the end user enters his credentials and pushes the "Login" button the EndUserApplication sends the inserted credentials to the EndUserHandler, which then asks to the DataBaseHandler to check if the inserted credentials are correct. The DataBaseHandler queries the data base and if a tuple is found, then a EndUser is created and is returned to the EndUserHandler and then to the EndUserApplication, which can in this way load the profile. Otherwise, a null object is returned and the procedure is restarted.

Sequence Diagram 10: An authority signs up

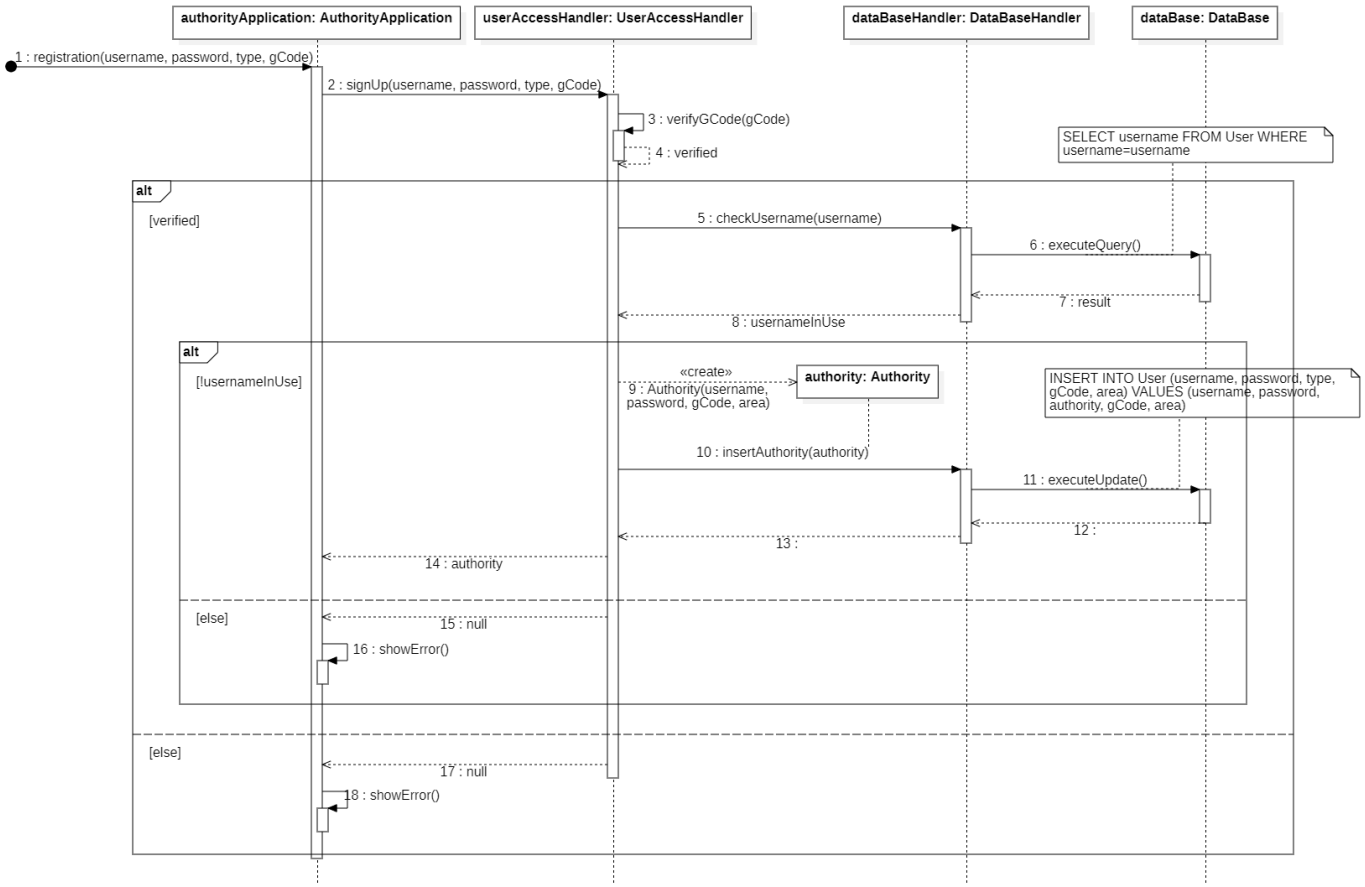

Figure 14: Sequence Diagram 10

Also the sign up procedure is shown through the tenth sequence diagram (the same procedure is adopted for the municipality user and also for the end user, but without the part concerning the governmental code). As soon as an authority fills the form with the chosen credentials and his governmental code and pushes the "Signup" button, the AuthorityApplication sends these information to the UserAccessHandler. The UserAccessHandler then verifies the inserted governmental code, and only if it is correct it proceeds (the governmental code is linked to an assigned area, so from the code is possible to know the assigned area). Then, if the code is right, it checks that the chosen username doesn't already exists in the database by asking it to the DataBaseHandler. In case of negative response, the UserAccessHandler creates an Authority with the credentials, the governmental code and the area and passes it to the DataBaseHandler which inserts these data into the data base. Then the UserAccessHandler returns the Authority also to the AuthorityApplication. In case of positive response, so in case the username is already in use, an error message is shown to the user which has then to choose another username and the procedure is restarted.

2.5 COMPONENT INTERFACES

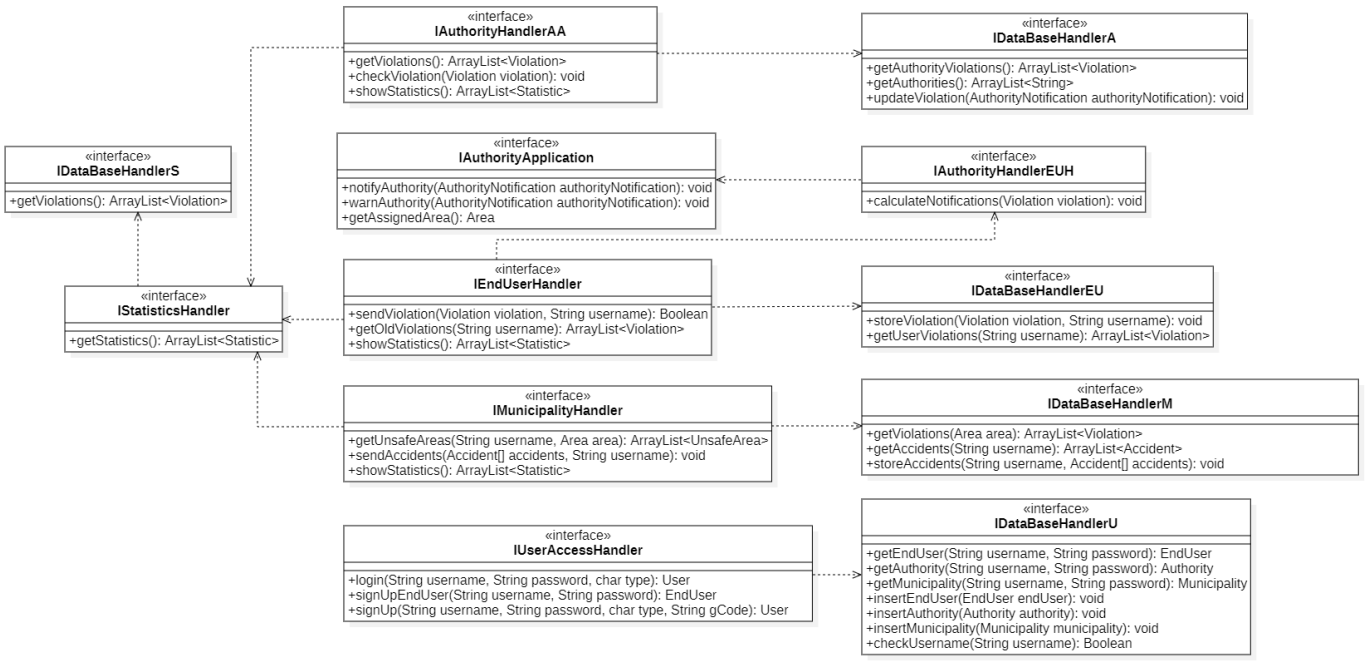

Figure 15: Component Interfaces

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 interested 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 IAuthorityHandlerAA

This interface is exposed by the AuthorityHandler component for the usage of the AuthorityApplication. In particular, it is needed for two main reasons, the first is to mine information from SafeStreets and the second is to warn the system that an authority checked a violation. This interface 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 way through which the authority user can mine information.

checkViolation(Violation violation): void

This method takes as parameter a Violation 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(): ArrayList<Statistic>

This method does not take parameters and returns an array of Statistics. The aim is to get the statistics from 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 shows 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 IAuthorityHandlerEUH

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): void

This method takes as parameter a Violation and does not return any value. 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 three methods:

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<String>

This method does not take parameters and returns an array of String. 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 then 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 that some authority has checked a violation.

updateViolation(AuthorityNotification authorityNotification): void

This method takes as parameter an AuthorityNotification and does not return any value. The aim is to update the violations saved in the data base, marking them as checked when an authority checks one of them.

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 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(): ArrayList<Statistic>

This method does not take parameters and returns an array of Statistics. 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 shows 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 violations sent by an end user. This interface exports two methods:

storeViolation(Violation violation, String username): void

This method takes as parameters a Violation and the username of the end user and does not return any value. 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(String username, Area area): ArrayList<UnsafeArea>

This method takes as parameters the username of the municipality and the Area under the jurisdiction of it and returns an array of UnsafeAreas. The aim is to calculate the set of unsafe areas that characterise 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[] accidents, String username): void

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

showStatistics(): ArrayList<Statistic>

This method does not take parameters and returns an array of Statistics. 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 shows 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 MunicipalityHandler in order to make possible to get violations and to store data about accidents. In particular, it exports three methods:

getViolations(Area area): ArrayList<Violation>

This method takes as parameter the Area representing the territory of the municipality 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 is able to use them to calculate the most unsafe areas of the territory of the municipality and also show them to the municipality user.

storeAccident(String username, Accident[] accidents): void

This method takes as parameters the username of the municipality and an array of Accidents and does not return any value. 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.

getAccidents(String username): ArrayList<Accident>

This method takes as parameter the username of the municipality 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 is 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 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, char type): User

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 a char defining the type of user, and then returns a User. The aim is to make possible to each user to access its account everywhere.

signUpEndUser(String username, String password): EndUser

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 EndUser. 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 is not already in use by another user. In case the chosen username is already in use the function returns null object, thus making the user select another username.

signUp(String username, String password, char type, String gCode): User

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 User. 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 null object, thus making the user to reenter it.

2.5.10 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 seven methods:

getEndUser(String username, String password): EndUser

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 EndUser. 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 the EndUser, otherwise a null object.

getAuthority(String username, String password): Authority

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 Authority. 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 the Authority, otherwise a null object.

getMunicipality(String username, String password): Municipality

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 Municipality. 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 the Municipality, otherwise a null object.

insertEndUser(EndUser endUser): void

This method takes as parameter an EndUser and then returns void. The aim is to insert the credentials of the new end user in the data base.

insertAuthority(Authority authority): void

This method takes as parameter an Authority and then returns void. The aim is to insert the credentials of the new authority user in the data base.

insertMunicipality(Municipality municipality): void

This method takes as parameter a Municipality and then returns void. 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.

2.5.11 IStatisticsHandler

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

getStatistics(): ArrayList<Statistic>

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

2.5.12 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 parameters 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 layer and part of the application layer (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 allowed functionalities. 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 and by the DataBaseHandler to create the Violation, by the StatisticsHandler to create the Statistic, by the AuthorityHandler to create the AuthorityNotification, by the MunicipalitySoftware and by the DataBaseHandler to create the accidents data (Accident), by the MunicipaltyHandler to create the UnsafeAreas and by the UserAccessHandler and by the DataBaseHandler to create the Users (Authority, EndUser, Municipality).
- Observer: the AuthorityApplications 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.
- It allows to have reduced data redundancy and high data consistency.
- It supports a quantitative data processing.
- It has a unified language for queries.

3 USER INTERFACE DESIGN

Here the discourse made in the RASD document regarding the user interfaces 4 is resumed and a more polish and precise description is given, thus giving the idea of how the user interface will actually be implemented.

It is immediately necessary to underline the distinction between three different user interfaces that must be developed, one per user type. They are well explained in the next three sections.

3.1 AUTHORITY INTERFACE

The user interface that is available for the authorities is composed of three different pages that can be navigated via a navigation bar at the bottom of the screen. The first page is the "Traffic Violations" page (shown in the picture below), which lists all the traffic violations that have been sent by the end user till that moment. This page also contains the violations that has been received as a notification by the authority because they have occurred in a position contained into the assigned area of the authority, and they are identified by a blue dot on the left side of the violation itself.

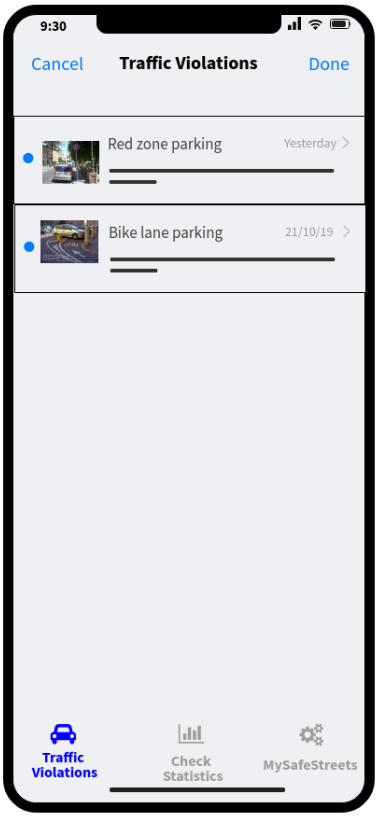

Figure 16: "Traffic Violations" page

The second page is the "Check Statistics" page (shown in the picture below), which is the same for all the three types of user and which allows them to see the traffic violations statistics elaborated by the system. In particular, it shows a marked map containing a coloured dot for each violation sent. The dot is placed in the exact position in which the violation has been occurred and its colour indicates the type of the infringement (green for bike lane parking, red for parking on the sidewalk and yellow for the reserved parking). Below the map there is a legend that explains what each colour corresponds to and a text that details the statistics calculated by the system in a more systematic way.

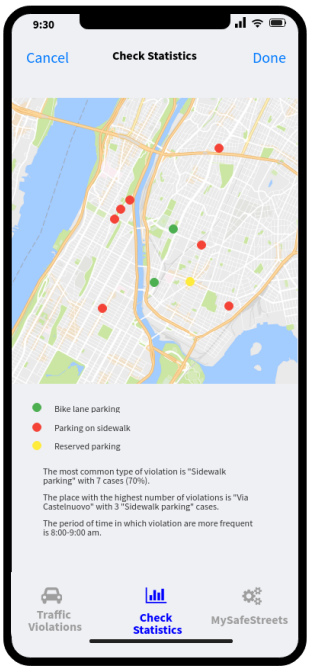

Figure 17: "Check Statistics" page

The third page is the "MySafeStreets" page (shown in the picture below), which again is also part of the interface of the other two types of users and which represents the page through which is possible to handle data about the account of the user. This page was not discussed in the previous chapters as it has a rather marginal task, namely to allow the user to change the username or the password.

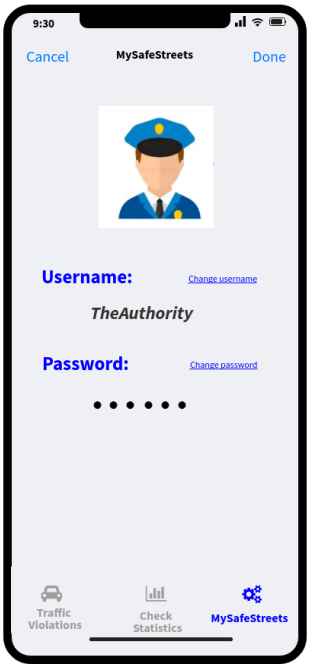

Figure 18: "MySafeStreetsPage" page

3.2 END USER INTERFACE

Also the user interface available for the end user is composed of three different pages, again navigable through the navigation bar at the bottom of the screen. Since the pages "Check Statistics" and "MySafeStreets" are exactly the same as those explained in the previous section, here it is exposed only the "Create Violation" page (shown in the picture below), which is the only one that differs from the others with respect to the interface provided to the authority user. In particular, through this page the end user can create a new violation and send it to SafeStreets. It is in fact possible to insert the picture of the infringement, select the traffic violation type and write down a description of the situation. Then pushing the "Send" button the information just created is sent to the system.

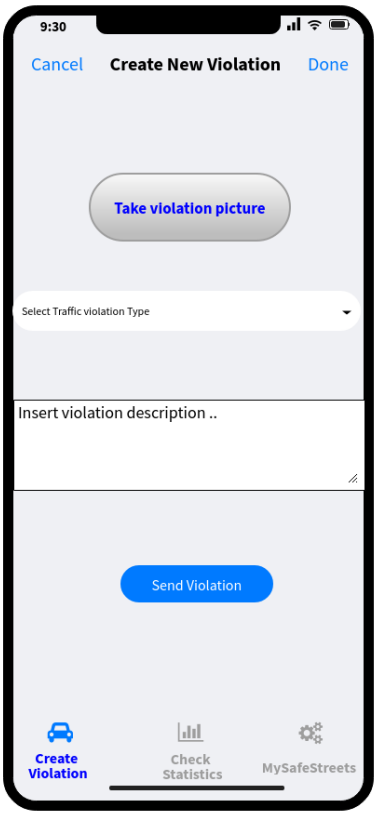

Figure 19: "Create Violation" page

3.3 MUNICIPALITY INTERFACE

The user interface available for the municipality user is composed of four pages, which in this case can be navigated through the navigation bar at the top of the screen. The first page is the "Insert Accidents" page (shown in the picture below), which allows the municipality user to insert data regarding the accidents happened in their jurisdiction. In particular, it provides a button that allows to upload the file containing accidents information, a text section that shows the data already inserted and a "Send your accident data" button for sending the inserted data to SafeStreets.

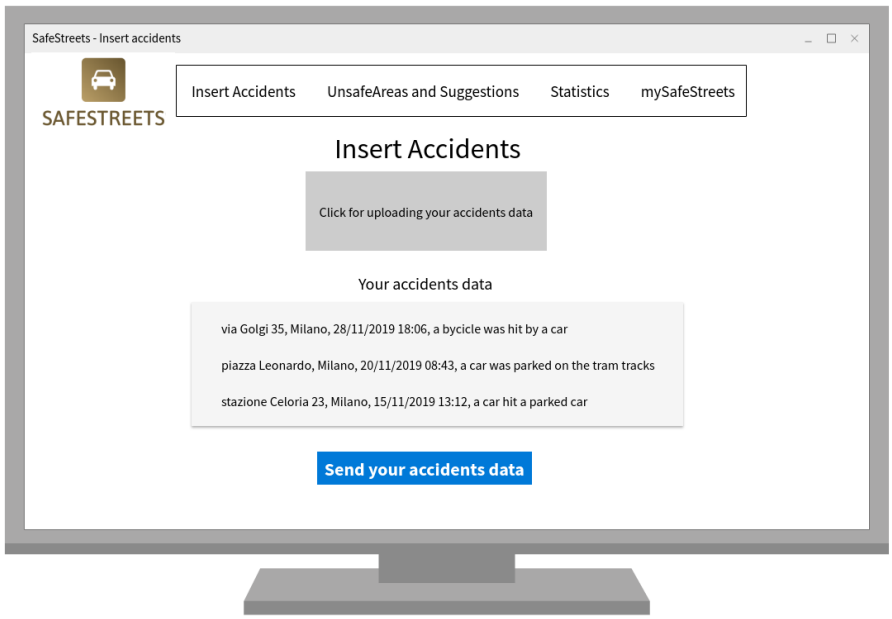

Figure 20: "Insert Accidents" page

The second page is the "Unsafe Area and Suggestions" page (shown in the picture below), which provides a map showing the most unsafe areas that characterize the territory under the jurisdiction of the municipality (the zones coloured by red). Selecting on a specific unsafe area in the map, the page also presents on the right of the map a text giving the right position of the unsafe area and the interventions elaborated by SafeStreets in order to make the area safer.

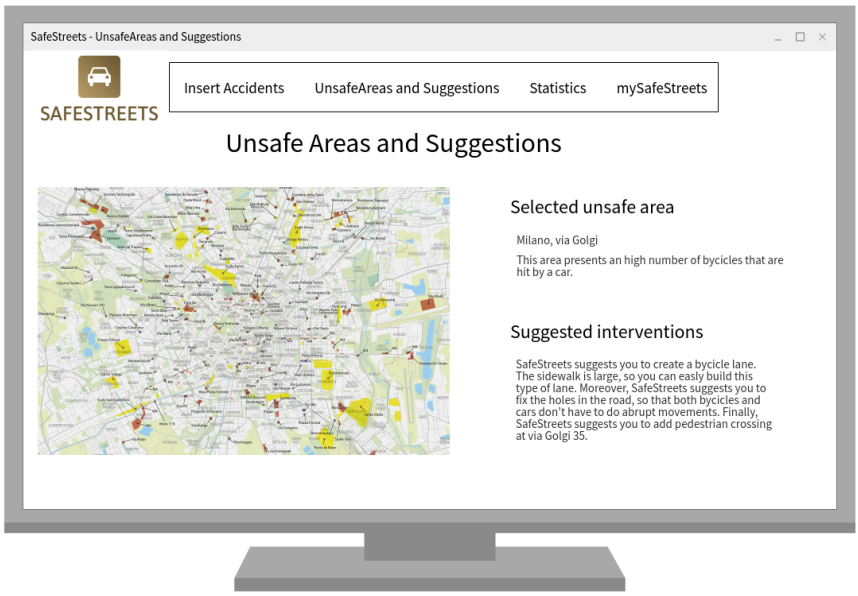

Figure 21: "Unsafe Area and Suggestions" page

The third page is the "Statistics" page (shown in the picture below), which allows them to see the traffic violations statistics elaborated by the system. In particular, it shows a marked map containing a coloured dot for each violation sent. The dot is placed in the exact position in which the violation has been occurred and its colour indicates the type of the infringement (green for bike lane parking, red for parking on the sidewalk and yellow for the reserved parking). On the right side of the map there is a legend that explains what each colour corresponds to and a text that details the statistics calculated by the system in a more systematic way.

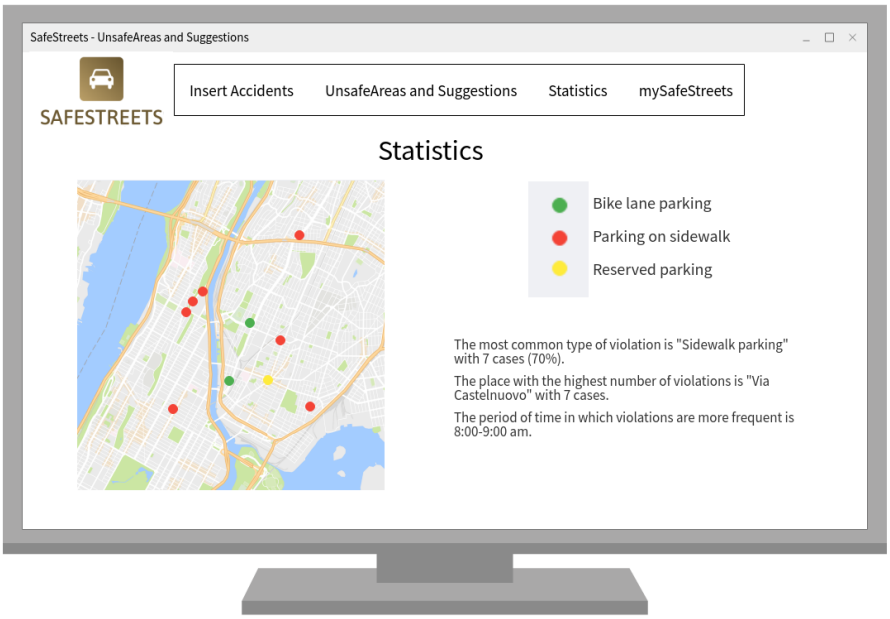

Figure 22: "Statistics" page

The fourth and last page is the "MySafeStreets" page, which, as for the other two user types, represents the page through which the user can see information about his account and change his credentials.

Figure 23: "MySafeStreets" page

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 requirements.

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, DataBaseHandler
r8	R8	EndUserApplication, EndUserHandler, DataBaseHandler, MunicipalitySoftware
r9	R9	AuthorityHandler, AuthorityApplication, DataBaseHandler
r10	R10	EndUserApplication, EndUserHandler, AuthorityHandler, AuthorityApplication, MunicipalitySoftware, MunicipalityHandler, StatisticsHandler, DataBaseHandler
r11	R11	StatisticsHandler, DataBaseHandler, MunicipalitySoftware
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	EndUserHandler, AuthorityHandler, AuthorityApplication, DataBaseHandler
r19	R19	AuthorityHandler, AuthorityApplication, DataBaseHandler
r20	R20	AuthorityHandler, AuthorityApplication
r21	R21	AuthorityHandler, DataBaseHandler
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 requests 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.

DataBaseHandler It retrieves data about violations from the data base without getting the identity of the user who reported 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.

DataBaseHandler It retrieves data only about violations sent by the user who requested them without getting violations sent by other users.

MunicipalitySoftware It does not provide to the user the functionality that allows to retrieve violations.

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.

DataBaseHandler It retrieves data about violations sent till that moment from the data base.

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.

MunicipalitySoftware 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 user accesses to statistics or a municipality user accesses to unsafe areas, SafeStreets must provide him the traffic violation statistics and the unsafe areas information updated to the last violation sent.

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

StatisticsHandler It recalculates the statistics every time it receives a request for them.

MunicipalityHandler It recalculates the unsafe areas every time it receives a request for them.

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.

EndUserHandler It forwards the reported violation to the AuthorityHandler, triggering the notification mechanism.

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 assigned 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 assigned 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 each time it is warned about a new checking.

AuthorityHandler When an authority wants to check a violation it requests the list of all the authorities and calculates the set of them who have to be warned of the event.

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

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

DataBaseHandler It retrieves the list of all the violations sent till that moment that have already been 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 allows to the end user to see all the violations he has sent in the past.

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

Project Plan

First Phase: EndUser

Activity	Beginning of the activity	Duration of the activity
DataBaseHandler Implementation	1	4
UserAccessHandler Implementation	2	3
EndUserHandler Implementation	2	3
EndUserApplication Implementation	3	6
DataBaseHandler and DataBase service Integration	5	2
UserAccessHandler, EndUserHandler, DataBaseHandler Integration	5	3
EndUserApplication, UserAccessHandler, EndUserHandler Integration	8	3
Single units testing	5	7
First Phase Testing	11	5

Second Phase: Authority

Activity	Beginning of	Duration of t
DataBaseHandler Implementation	1	2
UserAccessHandler Implementation	2	2
AuthorityHandler Implementation	2	4
EndUserHandler Implementation	5	2
AuthorityApplication Implementation	2	7
DataBaseHandler and DataBase service Integration	3	1
UserAccessHandler, AuthorityHandler, DataBaseHandler Integration	6	3
AuthorityHandler and EndUserHandler inegration	7	2
AuthorityApplication, UserAccessHandler, AuthorityrHandler Integration	9	3
Single units testing	3	9
Second Phase Testing	12	5

Third Phase: Statistics

Activity	Beginning of	Duration of t
DataBaseHandler Implementation	1	2
StatisticsHandler Implementation	2	5
AuthorityHandler Implementation	4	2
EndUserHandler Implementation	4	2
AuthorityApplication Implementation	6	4
EndUserApplication Implementation	8	2
DataBaseHandler and DataBase service Integration	3	2
StatisticsHandler, DataBaseHandler Integration	7	2
AuthorityHandler, EndUserHandler, StatisticsHandler Integration	7	3
StatisticsHandler and Map services integration	7	5
AuthorityApplication and AuthorityrHandler Integration	10	2
EndUserApplication and EndUserHandler Integration	10	2
Single units testing	3	9
Third Phase Testing	12	5

Fourth Phase: Municipality

Activity	Beginning of	Duration of t
DataBaseHandler Implementation	1	2
UserAccessHandler Implementation	2	2
MunicipalityHandler Implementation	3	4
MunicipalitySoftware Implementation	2	7
DataBaseHandler and DataBase service Integration	3	2
UserAccessHandler, DataBaseHandler Integration	4	3
StatisticsHandler, MunicipalityHandler, DataBaseHandler Integration	7	3
MunicipalitySoftware and MunicipalityHandler Integration	9	3
Single units testing	3	9
Fourth Phase Testing	12	5

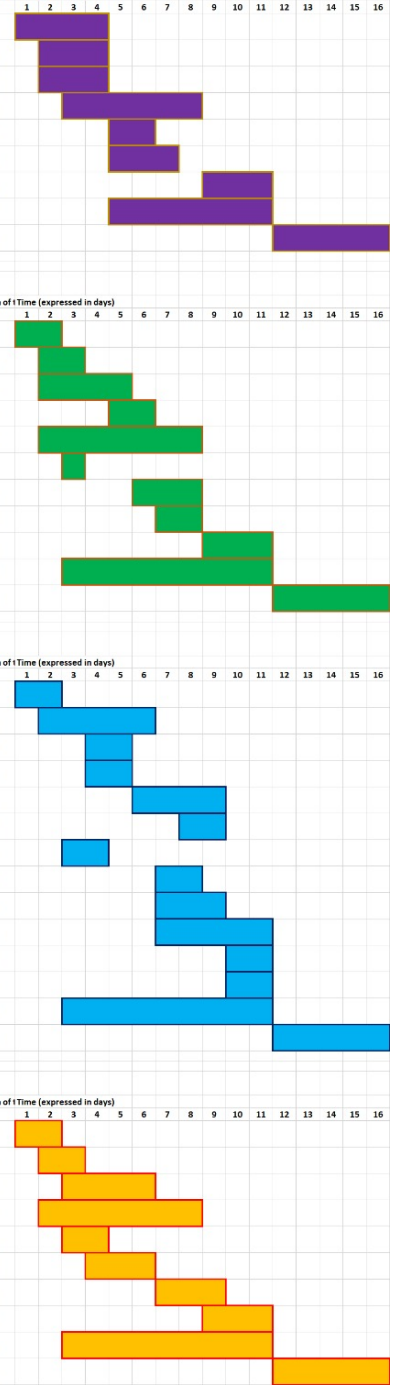

Figure 24: Gantt

The entire SafeStreets system will be implemented, integrated and tested along four successive phases. The Gantt diagram 5 above shows in a precise way how the four phases are oganized. In the following sections each phase is treated separately and detailed from the point of view of the implementation, of the integration and finally of the testing.

5.1 PHASE 1

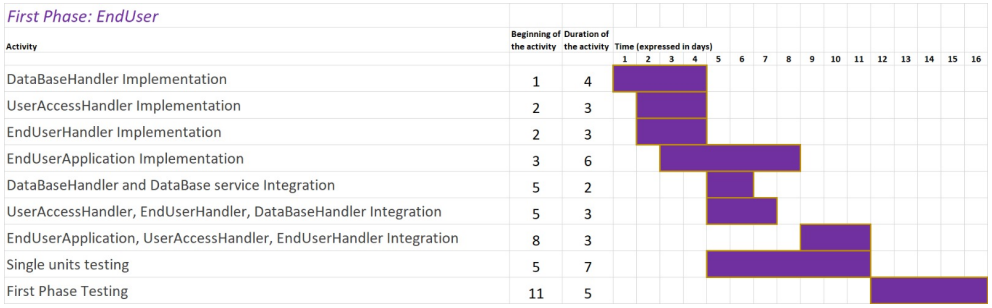

Figure 25: Gantt - first phase

5.1.1 Phase 1 - Implementation plan

In the first phase will be implemented the core functionality of SafeStreets, which is that of giving the possibility to the end users to report a violation which occurs in front of them. In order to do so the first component to implement is the DataBaseHandler, in particular will be implemented the functionalities offered by it that allow to store and retrieve the violations into and from the data base. However, it is necessary for the end users to be able to register and log into the system, so also the functionalities that permit to store and retrieve data about end users have to be implemented in this phase. Once this is done it is possible to implement the UserAccessHandler which manages the log in and registration procedures and also to begin the development of the EndUserHandler. Finally, in order to reach the goal of the phase, the EndUserApplication will be implemented, thus completing the chain that permits to the end users to register, log in, send violations and retrieve their own reported infringements (together with the EndUserApplication component will be implemented also the user interface shown in the chapter 3 related to the end user).

5.1.2 Phase 1 - Integration plan

During the completion of the implementation of the first phase, the components are integrated with each other. In particular, as soon as the DataBaseHandler component is implemented, it is integrated with the data base, such that it can really store and retrieve data into and from it.

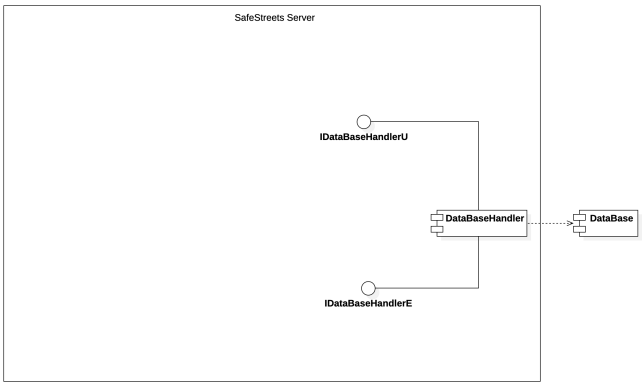

Figure 26: First integration

Then the UserAccessHandler component is integrated with both the EndUserHandler component and the DataBaseHandler, thus completing the sign in and sign up functions.

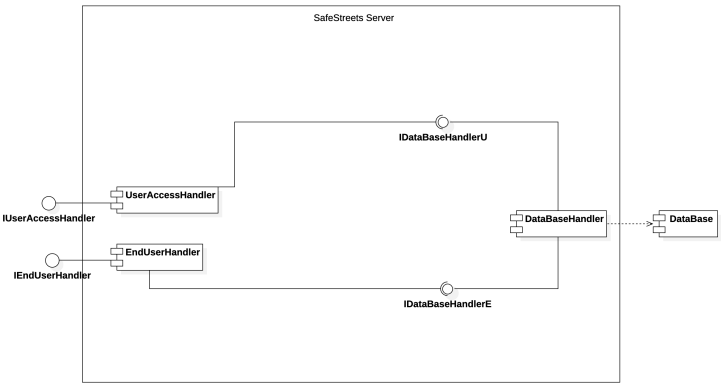

Figure 27: Second integration

Finally, the EndUserApplication component is integrated with the UserAccessHandler component and the EndUserHandler component, in this way making available also the core functionality of SafeStreets, which is that of granting to the end user the possibility of reporting new violations and retrieving the old ones that he has sent.

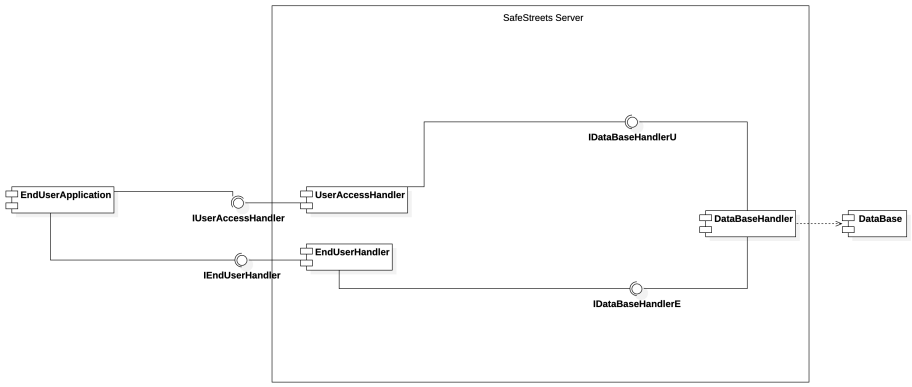

Figure 28: Third integration

5.1.3 Phase 1 - Test plan

As soon as components are implemented, unit testing is performed on each component, being sure that all the implemented functionalities work in the right way. The following functionalities are tested:

DataBaseHandler The end user log in and sign up, the insertion of a violation and finally the retrieval violations reported by the end user.

UserAccessHandler The sign up and sign in mechanisms related to the end users.

AuthorityHandler The sign up and sign in mechanisms related to the authority users, the retrieving of all the violations, the report of the violations occurred in the assigned area of the authorities and the notification for the checking of a violation by another authority.

EndUserHandler The sign in and sign up functionalities and the reporting and the retrieving of violations mechanisms.

EndUserApplication The sign in and sign up functionalities and the reporting and the retrieving of violations mechanisms.

Once this is done, an incremental integration testing is performed, adopting a top-down technique, such that it is possible to exercise the interfaces and the modules interactions. So, once the UserAccessHandler component has been implemented and integrated with the DataBaseHandler component, the sign in and sign up functionalities are tested again. The same mechanism is adopted as soon as the EndUserHandler and the EndUserApplication components are integrated with the rest of the components. In this case, however, in addition to sign up and sign in testing, also the insertion of a new violation and the retrieval of the set of all the reported violations are tested.

5.2 PHASE 2

5.2.1 Phase 2

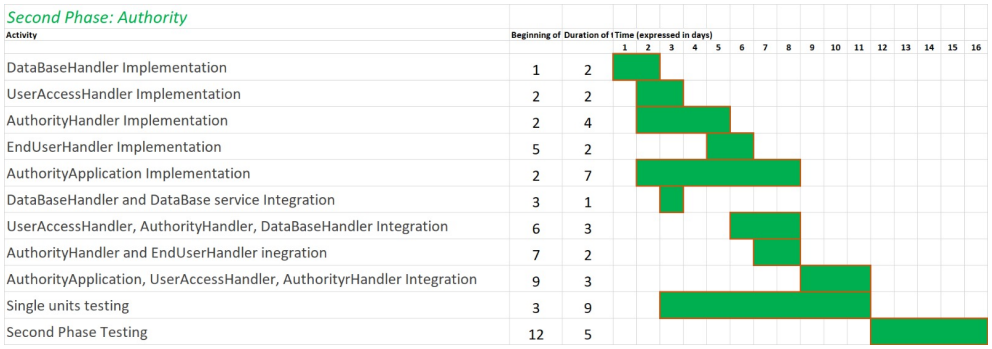

Figure 29: Gantt - second phase

5.2.2 Phase 2 - Implementation plan

In the second phase will be implemented the functionality that permits to the authorities to be notified about the violations that occurs in their assigned area and to see all the violations that have been reported. In order to do so, the DataBaseHandler component has to be enriched with the functionality that allows to retrieve all the violations from the data base and with that one which permits to get and store authorities data. Then also the UserAccessHandler has to be enriched with the functionalities that allow to register and log in the authority user. Once done, the AuthorityHandler can be implemented with all its features and it has to be added to the EndUserHandler the functionality that allows it to forward a violation report to the AuthorityHandler such that it can calculate the authorities that have to be notified and notify them. Finally, the AuthorityApplication component can be implemented, thus completing also the front end part of the application which regards the authority users (together with the AuthorityApplication component will be implemented also the user interface shown in chapter 3 related to the authority user).

5.2.3 Phase 2 - Integration plan

Also during the implementation scheduled in phase two, as components are completed, they are integrated with each other. In particular, the DataBaseHandler component is further integrated with the data base in order to also make possible to store and retrieve data regarding authorities.

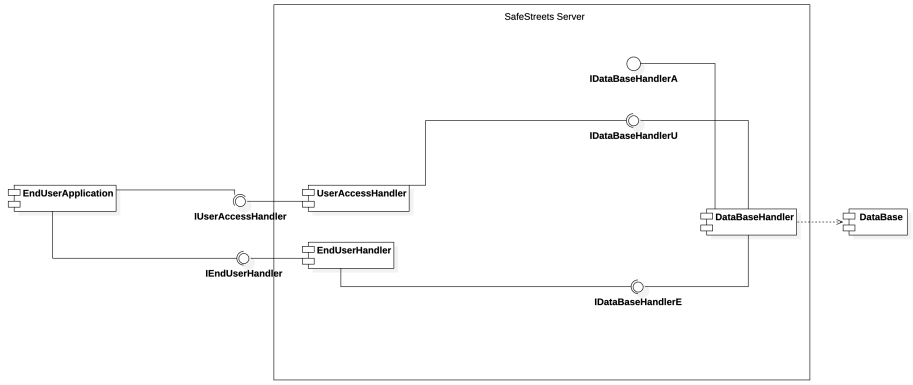

Figure 30: Fourth integration

Once this is done, the UserAccessHandler component can be integrated with the AuthorityHandler and the DataBaseHandler, granting the sign in and sign up functionalities for the authority users.

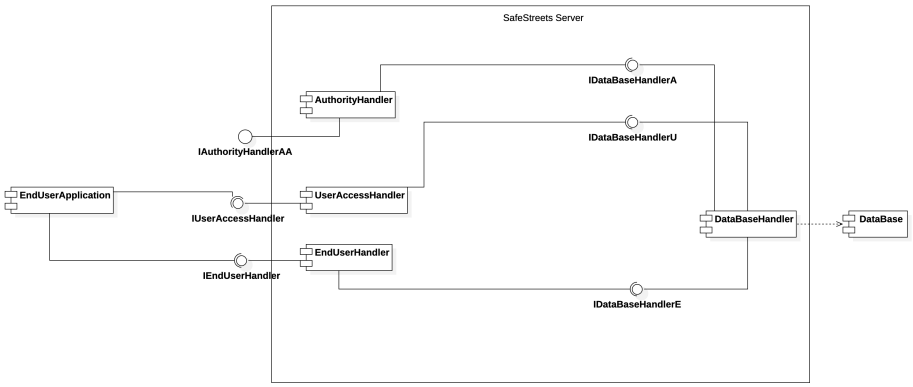

Figure 31: Fifth integration

Finally, the AuthorityHandler is integrated with the EndUserHandler, such that each time a violation is reported the former can notify the authorities, and as soon as the AuthorityApplication component is ready, it is integrated with the UserAccessHandler component and the AuthorityHandler component.

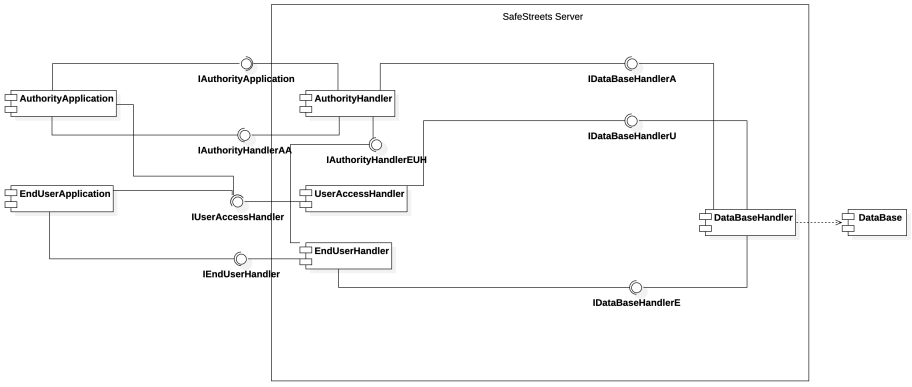

Figure 32: Sixth integration

5.2.4 Phase 2 - Test plan

Also in the second phase testing, each component is firstly tested singularly. So, testing units are created for all the components as soon as they are implemented in order to check the following functionalities:

DataBaseHandler The sign up and sign in functionalities related to the authority users and the mechanism for retrieving all the violations.

UserAccessHandler The sign up and sign in mechanisms related to the authority users.

AuthorityHandler The sign up and sign in mechanisms related to the authority users, the retrieving of all the violations, the report of the violations occurred in the assigned area of the authorities and the notification for the checking of a violation by another authority.

EndUserHandler The forwarding of a violation to the AuthorityHandler as soon as it has been reported.

AuthorityApplication The sign up and sign in mechanisms related to the authority users, the retrieving of all the violations, the report of the violations occurred in the assigned area of the authorities and the notification for the checking of a violation by another authority.

Once this is done, the same functionalities tested with the unit testing on the single components are now checked incrementally on more components at the same time with the integration testing. So, going in the same order defined by the integration plan, functionalities are tested firstly on the DataBaseHandler, UserAccessHandler and AuthorityHandler. Then, when the EndUserHandler is integrated a test is done again. And a last one when the AuthorityApplication component is added.

5.3 PHASE 3

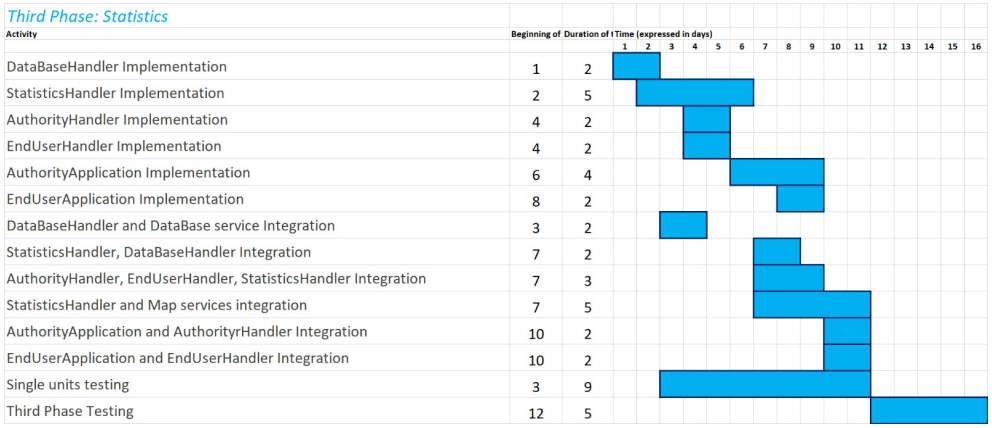

Figure 33: Gantt - third phase

5.3.1 Phase 3 - Implementation plan

In the third phase the calculation of statistics is made possible. In particular, here the DataBaseHandler adds the interface IDataBaseHandlerS that allows to the StatisticsHandler to retrieve data necessary for building the statistics. Subsequently, the StatisticsHandler component can be entirely implemented and the EndUserHandler component, the AuthorityHandler component, the EndUserApplication component and the AuthorityApplication component can be enriched with the functionality that permits to the end users and to the authorities to request for the traffic violation statistics. This phase is quite faster then the others because the procedure is the same for both the authority user and for the end user, so once implemented the functionality the first time then the code can be duplicated.

5.3.2 Phase 3 - Integration plan

Regarding the integration of phase three, once enriched the DataBaseHandler component with the functionalities needed for getting data necessary to calculate statistics, it can be further integrated with the data base in order to really grant this feature.

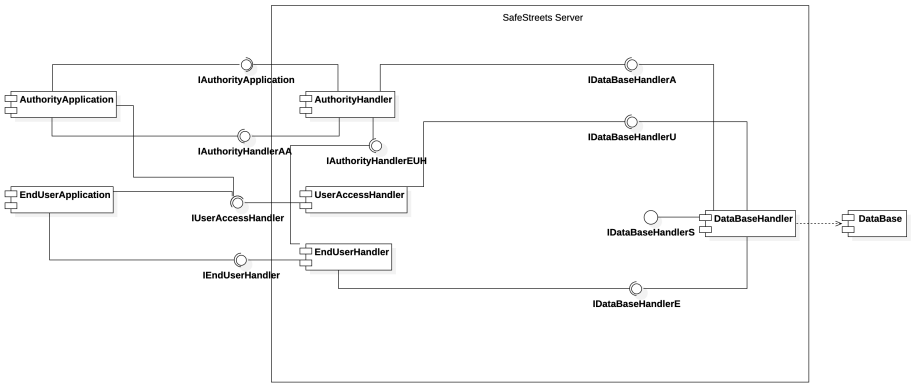

Figure 34: Seventh integration

Then, when the StatisticsHandler is completed, it can be integrated with the DataBaseHandler component.

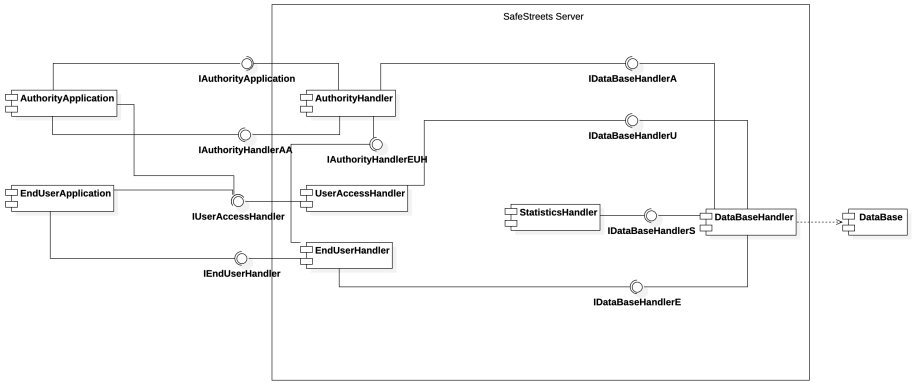

Figure 35: Eighth integration

Subsequently, the EndUserHandler component and the AuthorityHandler component are integrated with the StatisticsHandler and afterwards the former is integrated with the EndUserApplication component and the second with the AuthorityApplication. Before this two last integrations however, the StatisticsHandler component is integrated with the external map service called OpenStreetMap, such that it can then provide the map on which it shows the statistics.

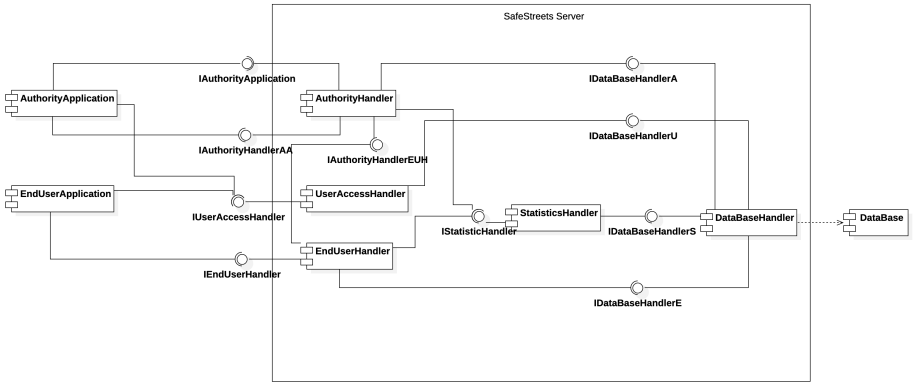

Figure 36: Ninth integration

5.3.3 Phase 3 - Test plan

Since the third phase is focused on the development of the statistics elaboration, with the unit testing this functionality is tested in each component singularly. Once done, the statistics elaboration is also tested through an incremental integration testing, adopting a top-down technique. So, each time a new component is integrated with the others the same functionality is tested again, thus being sure of the correct behaviour of them together.

5.4 PHASE 4

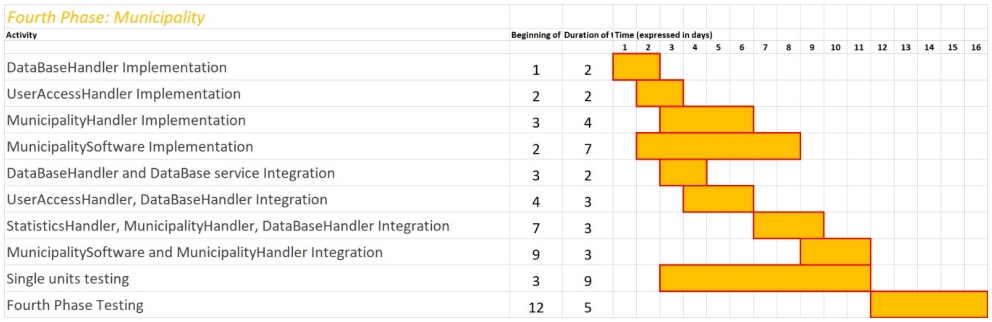

Figure 37: Gantt - fourth phase

5.4.1 Phase 4 - Implementation plan

In the fourth and last phase is implemented the part of SafeStreets that allows to include the municipality users in the system. In particular, the DataBaseHandler component is enriched with the ability of storing and retrieving data about municipality users. Then also the UserAccessHandler component has to be expanded, giving it the ability of logging in municipalities. Subsequently, the MunicipalityHandler component can be implemented, with the functionalities that allow to send accidents data, to calcu-

late unsafe areas and suggestions and to get statistics. Finally, the MunicipalitySoftware component can be implemented, thus completing the front end which regards the municipality users and in this way concluding the implementation (together with the MunicipalitySoftware component will be implemented also the user interface shown in the chapter 3 related to the municipality user).

5.4.2 Phase 4 - Integration plan

In phase four, the DataBaseHandler is again integrated with the data base in order to provide the sign in and sign up functionalities to the municipality users.

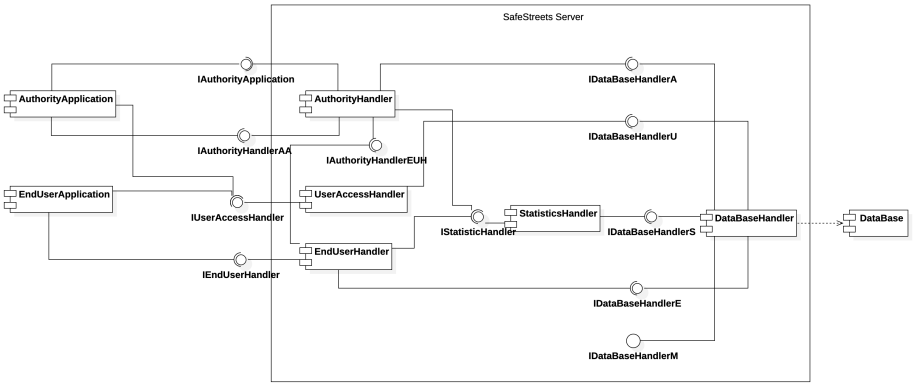

Figure 38: Tenth integration

Then, the just created MunicipalityHandler component is integrated with UserAccessHandler and with the DataBaseHandler and at the same time also with the StatisticsHandler.

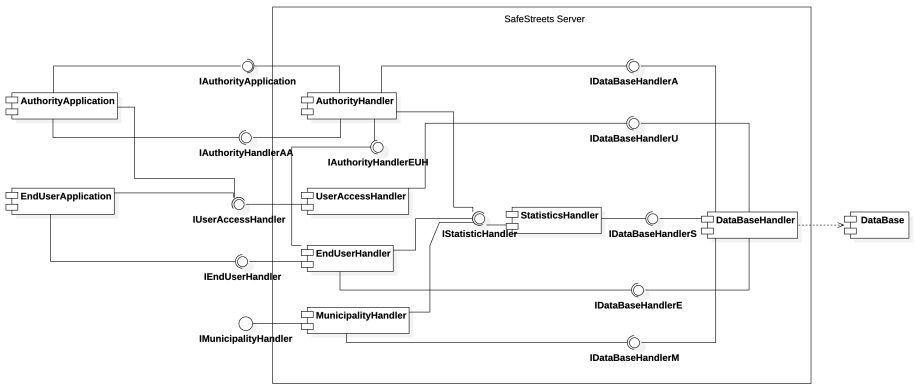

Figure 39: Eleventh integration

Finally, the MunicipalitySoftware component is integrated with the MunicipalityHandler component, thus completing the functionalities related to municipality users, and completing the integration of the entire system.

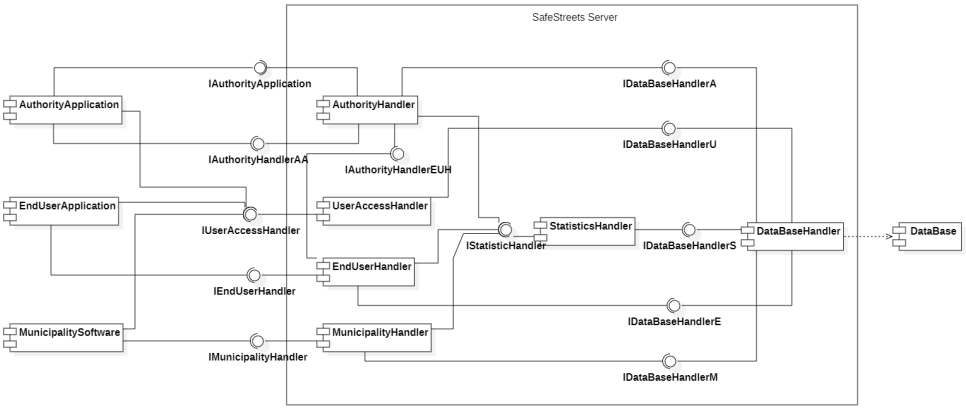

Figure 40: Twelfth integration

5.4.3 Phase 4 - Test plan

In the fourth phase, functionalities related to the municipality users are implemented, so the unit testing is performed on each component, checking:

DataBaseHandler The sign up and sign in functionalities related to the municipality users, the mechanism for retrieving all the violations and the mechanism for storing and retrieving the accidents data.

UserAccessHandler The sign up and sign in mechanisms related to the municipality users.

MunicipalityHandler The sign up and sign in mechanisms related to the municipality users, the retrieving of all the violations and the accidents data and the report of the accidents data occurred in the area under the jurisdiction of the municipality.

MunicipalitySoftware The sign up and sign in mechanisms related to the municipality users, the mechanism for reporting the accidents data, the mechanism for seeing statistics and the mechanism for seeing unsafe areas.

Once done, as for all the other phases, an incremental integration testing through a top-down approach is performed. So, each time a new component is integrated with all the others functionalities are tested again, exercising the interfaces and the modules interactions.

Since this is the last phase, at the end of it the entire system is completed, so it is possible to perform a system testing. In particular, since the system has to handle a quite huge quantity of data it is performed a load testing, bringing SafeStreets to its limits in order to identify them. Then, also a performance testing is carried out in order to be sure that the system guarantees a good level of performances and to identify inefficient algorithms that may affect them.

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
- 29/11/2019: 6h
- 30/11/2019: 1h 15min
- 01/12/2019: 4h 30min
- 02/12/2019: 45min
- 07/12/2019: 5h
- 08/12/2019: 5h 30min
- 15/12/2019: 2h

Total: 43h

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
- 28/11/2019: 3h 30min
- 29/11/2019: 2h 30min
- 01/12/2019: 4h

- 04/12/2019: 3h 15min
- 06/12/2019: 8h
- 07/12/2019: 5h
- 08/12/2019: 45min
- 09/12/2019: 2h 45min
- 15/12/2019: 1h

Total: 44h

7 REFERENCES

1. For UML diagrams we used StarUML software.
2. For the overview diagram we used Microsoft Word.
3. For the map external services we used OpenStreetMap, www.openstreetmap.org
4. For the mockups we used mockflow.com.
5. For the gantt diagram we used Microsoft Excel.
6. For the coordination and organization of the work we used a repository of github.com.