POLITECNICO
MILANO 1863

SafeStreets

REQUIREMENTS ANALYSIS AND SPECIFICATION DOCUMENT

Version 2 – 18/11/2019

Authors:

Giulio A. Abbo 10538950

Gianmarco Accordi 10587213

Massimiliano Bonetti 10560496

Professor:

Elisabetta Di Nitto

Academic year:

2019 – 2020

CONTENTS

1 INTRODUCTION

1.A PURPOSE

In this document will be presented a description of the SafeStreets application – from now on referred to as the *system* – along with the analysis of its goals, its requirements and the assumptions taken.

1.A.1 Problem overview

The system is addressed to two different types of entities: the subscribed user and the municipality.

The system will allow the users to send reports (including a picture, time, date, position and type) about traffic violations. The gathered data will be elaborated (plate recognition if the plate number is not provided, street name) and used to show to the users and the municipality the streets or areas with the highest frequency of violations; in addition the municipality will have access to a list of the plate numbers of the vehicles that have committed the most violations and to the suggestions of possible interventions.

The system will be able to collect data from the municipality about violations on the territory, crossing it with the data from user reports and using it as above. This will be referred to as the *data integration service*[1].

The municipality will have access to a list of suggested interventions, based on the received reports. This will be referred to as the *suggestion service*.

The system will also provide a way for the municipality to access the data from the user reports, to allow the generation of traffic tickets; the data from the generated traffic tickets will be used for building statistics (on the vehicle with most tickets and the trends in the issuing of tickets) accessible by the municipality together with the other insights. Care must be taken to ensure that the chain of custody is never broken. This will be referred to as the *access reports service*[2].

1.A.2 Goals

These are the goals of the SafeStreets system:

- G1: The System accepts valid reports by the users about the parking violations.
- G2: The System suggests possible interventions to the Municipality.
- G3: The System allows the Municipality to retrieve submitted parking violations of its competence area.
- G4: The System gives some statistics to the User about the violations.
- G5: The System can give all the statistics to the Municipality about the violations.
- G6: The System can retrieve the violations verified by the Municipality.

1.B SCOPE

In *Figure 1* is presented the initial distinction between events that belong to the world and those that belong to the machine (the system). The violations and the event *user finds a violation* are considered part of the world. Indeed, the machine is not aware of these events unless the user sends a report notifying the violation and its location, which are shared events. Likewise, are part of the shared phenomena the events in which the users or the municipality access the statistics and the municipality accesses the violations and the suggestions, the systems accesses data on violations from the municipality and vice

[1] This is the *Advanced Function 1* of the *Project Assignment.*
[2] This is the *Advanced Function 2* of the *Project Assignment.*

versa. The machine side of the model contains the data memorized, the process of data analysis and the process of plate recognition.

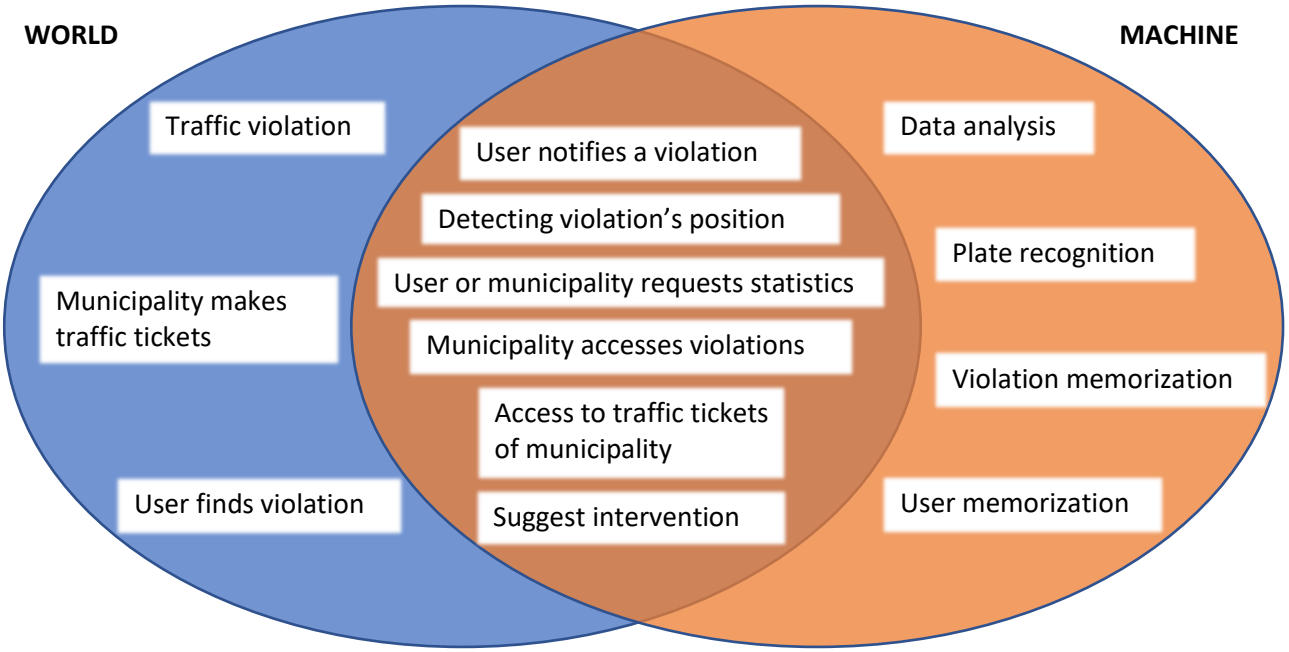

Figure 1 – Distinction between events of the world and of the machine

The next table identifies the most important phenomena, shared and not.

Phenomenon	Shared Y=Yes N=No	Who controls it W=World M=Machine
A car makes a traffic violation	N	W
The User sees the traffic violation	N	W
The User wants to report the violation	N	W
The User reports the violation	Y	W
The User takes a picture and adds it to the report	Y	W
The User insert in the report the position of the violation	Y	W
The User insert in the report the license plate of the car	Y	W
The machine calculates the position of the violation	N	M
The machine identifies the license plate of the car	N	M
The User indicates the type of the violation	Y	W
The User wants to contribute to the safe of the streets	N	W
The User registers himself/herself in the machine	Y	W
The User inserts the copy of his/her identity card	Y	W
The User inserts his/her information (e.g. first name, last name, address)	Y	W
The machine checks the identity card of the User	N	M
The machine accepts or rejects the registration of the User	Y	M
The User makes the login in the machine	Y	W
The User inserts his/her username and password	Y	W
The machine checks the username and the password	N	M

The machine accepts or rejects the username and the password	Y	M
The User wants to see some statistics on the machine	N	W
The User selects some statistics	Y	W
The machine calculates the statistics asked by the User	N	M
The machine shows the calculated statistics to the User	Y	M
The Municipality wants to make safer its streets	N	W
The Municipality asks for the registration to the employees of the SafeStreets company	N	W
The employees of the SafeStreets company register the Municipality	Y	W
The Municipality makes the login in the machine	Y	W
The Municipality inserts its username and password	Y	W
The employees of SafeStreets ask the Municipality to allow the machine to access to its violations	N	W
The Municipality offers to the machine a method to access to its violations	Y	W
The machine accesses the violations of the Municipality	Y	M
The Municipality wants to see some statistics on the machine	N	W
The Municipality selects some statistics	Y	W
The machine calculates the statistics asked by the Municipality	N	M
The machine shows the calculated statistics to the Municipality	Y	M
The Municipality wants to verify the last violations	N	W
The Municipality takes from the machine the last violations	Y	W
The Municipality verifies the violations taken from the machine and it generates traffic tickets from them	N	W
The machine calculates possible interventions	N	M
The machine suggests the Municipality some interventions	Y	M

The world in which the system will work is modelled as follows: all the authorities that oversee the traffic conditions or can generate traffic tickets are considered as one for simplicity and are referred to as *municipality*. The municipality is not a mandatory actor, the system can work fine even without any. The user is a person that is subscribed to the system; his identity is verified, for this reason he is considered trustworthy: he does not send false or wrong reports and SafeStreets will not check the correctness of the report. The users interact with the system mainly through a mobile device.

The system is considered to be supported by an organization of some kind, which handles the infrastructure necessary for the operativity and any contracts with the municipalities.

The data provided by the municipality is given for accurate and timely. All the services provided by third parties are supposed to be trustworthy: if they provide a result without error, then the result is assumed correct.

1.C DEFINITIONS, ACRONYMS, ABBREVIATIONS

1.C.1 Definitions

- (SafeStreets) System, Machine: the software to be and all its components

- User: the end user, a registered and logged in individual who can send reports and access statistics
- Municipality: the authority that oversees the road and traffic conditions in the area and generates tickets; to use the functions it must be registered and logged in
- (Traffic) Violations: all types of traffic violations punishable by law, e.g.: parking on bike lanes or reserved lots, double parking
- Report: an alert about a traffic violation
- Effectiveness of the system: the trend of the number of received reports in a given area with reference to the introduction of the SafeStreets' system.
- License Plate Recognizer or Recognition Plate System: is the third part service that identifies the license plate from the user's photo of the report
- Maps Service: is the third part service that identifies the place and position of the report
- Identity Verifier: is the third part service that verifies the identity of the user

1.C.2 Acronyms

- API: Application Programming Interface
- GPS: Global Positioning System, and any equivalent system such as GALILEO
- UI: User Interface
- S2B: Software to Be
- OS: operative system

1.C.3 Abbreviations

- G*n*: *n*th goal
- D*n*: *n*th domain assumption
- R*n*: *n*th requirement
- U*n*: *n*th use case

1.D REVISION HISTORY

- Version 1 First version of the document.
- Version 2 Minor misprint corrections, rephrasing and style changes.
  Added to the sequence diagrams a reference to the corresponding use case.
  Modified R2 to improve maintainability.
  A few other references have been added to chapter six.
  Changes to use case diagram.
  Added domain assumption D7.

1.E REFERENCE DOCUMENTS

- Project assignment: "Mandatory Project Assignment AY 2019-2020"
- Standard ISO/IEC/IEEE 29148:2018

1.F DOCUMENT STRUCTURE

This document is comprised of six chapters.

The first chapter introduces to the problem, which is then summarized into the goals of the system; what follows is the distinction of the various events in terms of the world and the machine[3], with some

[3] This distinction is the one proposed by M. Jackson in his paper "The World and the Machine".

details specially on the assumption that are taken on the world. The chapter ends with a list of the definitions, acronyms and abbreviations used in this document.

The second chapter provides some details on the modelling of the system, its requirements, details on the actors and the domain assumptions.

The third chapter focuses on the requirements, here is presented a prototype of the user interface and a description of the necessary hardware, software and communication interfaces. Then follow the use cases and some scenarios, the sequence diagrams and the mapping of the requirements and assumptions on the goals. Performance requirements, design constraints and software system attributes close the chapter, giving an overview of the constraints imposed on the system.

Chapter four contains the model – described using the formal language Alloy – of the most critical parts of the system and of the environment. This chapter contains also some worlds obtained from the model and the checks of some assertions.

An account on the number of hours spent by each member of the group to work on each part of the document is presented in chapter five.

Chapter six contains a list of the reference documents used in the writing of this.

2 OVERALL DESCRIPTION

2.A PRODUCT PERSPECTIVE

The system is divided into two products: a software for the user (mainly used on mobile devices) and the municipality, and one for the system's backend.

The report will contain the date and the time of the report, one or more pictures with the main one containing the license plate number, which will be recognized by the system if not inserted by the user, the position (inserted by the user if not retrieved by the location system of the device), the type of the violation and the author. The user will be able to send a report immediately when he finds a violation, but also to do so after some time; in this case he will select the photo from the device, and will manually insert the position (which will override the one from the device) and a time of the violation, which will be different from the time of the report's submission.

Figure 2 contains the class diagram representing the conceptual model of the application domain.

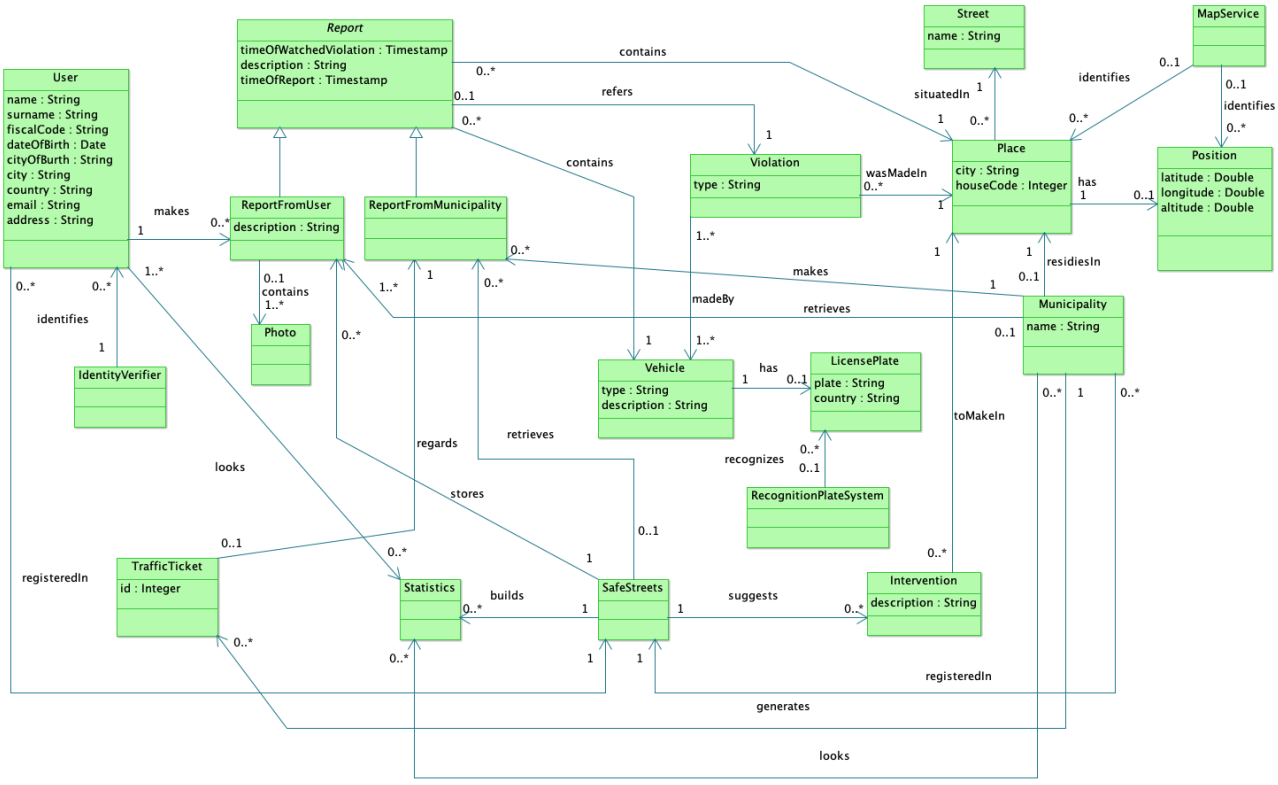

Figure 2 – Class diagram

Figure 3 shows a state chart for the municipality app. After the login, to the municipality is showed a map of its competence area with an overlay representing the streets with the most violations. From this state it can access other statistics, the suggestion provided by the system, and a list of the violations.

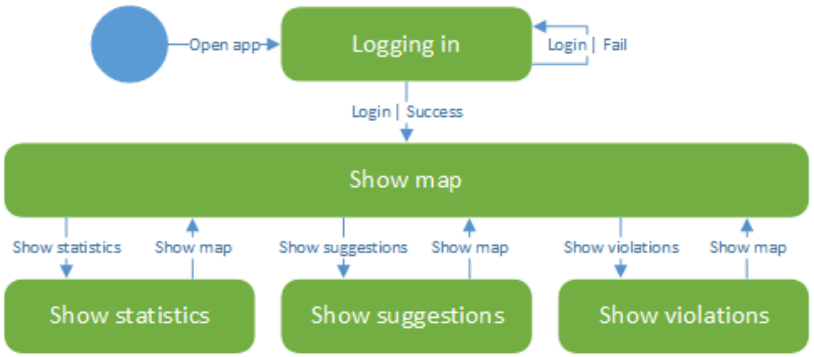

Figure 3 – State chart for the municipality

Figure 4 represents the states of the back-end software. This software awaits input from the municipality or the user and crosses data with the municipality periodically or on demand.

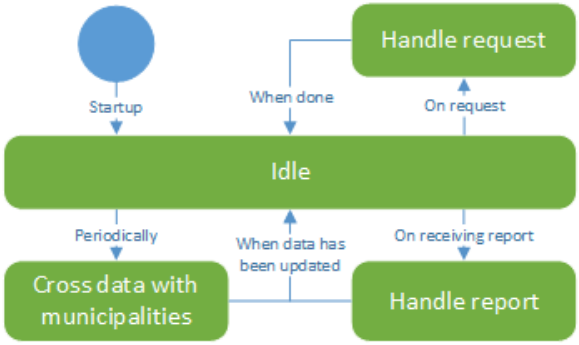

Figure 4 – State chart for the back end

Figure 5 shows the states of the user app. After the login, to the user is presented a map of his surroundings with an overlay representing the streets with the most violations, from there he can access other statistics or send a new report. The new report process is represented in detail.

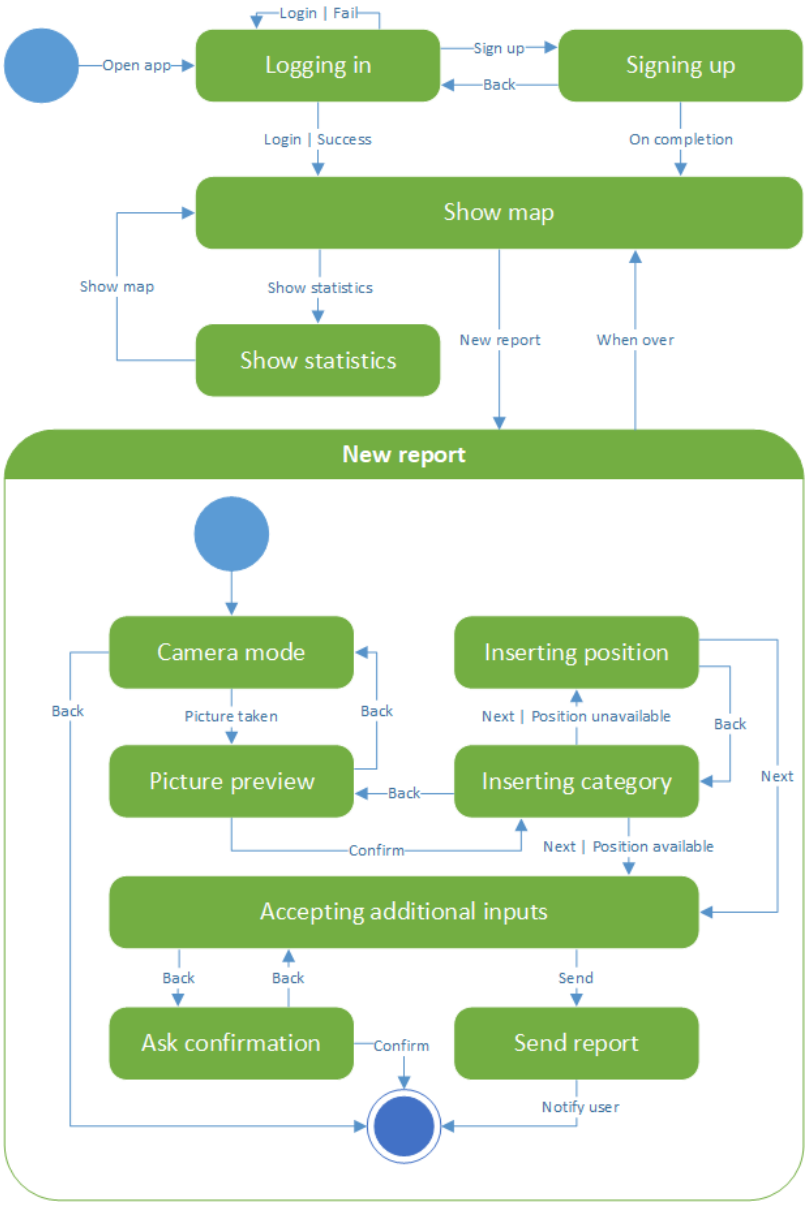

Figure 5 - State chart for the user

2.B PRODUCT FUNCTIONS

The requirements of the SafeStreets system are:

- R1: The reports about the violations are correctly stored.
- R2: The user can view the statistics calculated by the System with some exceptions[4]
  - R2.A: The vehicles that have committed the highest number of violations.
- R3: The Municipality can access only the data of the violations of its competence area.
- R4: Violations registered by the Municipality can be retrieved by the system.
- R5: The system must avoid the manipulation of the violations.
- R6: The system must be able to retrieve the position from the user or from the GPS
- R7: Only the Municipality can access the submitted parking violation of its competence area
- R8: The system must allow the User to take a picture or select one from the device.

[4] The user can not view the statistics specified.

- R9: The system accepts reports from the User.
- R10: The System must calculate some statistics
    - R10.A: The system must calculate the streets with the highest and the lowest number of violations.
    - R10.B: The system must calculate the effectiveness of the service.
    - R10.C: The system must calculate the vehicles (identified by the traffic plate) that have committed the highest number of violations.
    - R10.D: The system must calculate the most common violations of a given area
- R11: The municipality can view all the statistics calculated by the system.
- R12: The system must suggest interventions to the Municipality.
    - R12.A: Inspect an area
    - R12.B: New cycle lane
    - R12.C: New sidewalk
    - R12.D: New pedestrian crossing
    - R12.E: New parking
    - R12.F: New speed detector
- R13: The system accepts only reports with a valid plate number and position.
- R14: The system must allow the user to perform the registration and the login.
- R15: The system must allow the Municipality to perform the registration and the login.
- R16: The system must ask the User the non-mandatory attributes of the report.
- R17: The system must communicate with the Identity Verifier.
- R18: The system must communicate with the Plate Recognizer Service.
- R19: The System must communicate with the Maps Service.

The User can report only Parking violations. The User can select from this type of Parking violations:

- Parking on bike lanes
- Parking on reserved stall
- Double parking
- Parking on pedestrian crossing
- Parking on sidewalk
- Parking on traffic island
- Parking not payed
- Parking on red zone

SafeStreets can take from the Municipality also these types of violations:

- Traffic light violations
- Incident between vehicles
- Speed violations
- Against traffic violations
- Other violations

2.C USER CHARACTERISTICS

2.C.1 The user

The user is an individual who can send reports about traffic violations and access statistics, he must be registered and logged in to access the functionalities of the system. He will access the system mainly through a mobile device.

On registration, his identity is verified by a third party; he is accountable for the sent reports, and it is assumed that he will not send false reports.

2.C.2 The municipality

The municipality is the authority that oversees the road and traffic conditions in the area and generates tickets. To use the system, it must be registered and logged in. The registration process for municipalities is not handled directly by the system.

2.D ASSUMPTIONS, DEPENDENCIES AND CONSTRAINTS

The domain assumptions of the SafeStreets system are:

- D1: If the license plate recognizer recognises the license plate, then the result is correct.
- D2: If the map service recognises the street name from the coordinates, then the result is correct.
- D3: The identity card is correctly verified.
- D4: The Municipality possesses only real violations.
- D5: The data retrieved by the smartphone's GPS is correct.
- D6: The time reported by the user's smartphone is correct.
- D7: The user does not send false reports.

3 SPECIFIC REQUIREMENTS

3.A EXTERNAL INTERFACE REQUIREMENTS

3.A.1 User interfaces

The software will provide different functionalities for the user and the municipality. In Figure 6 are shown some prototype mock-ups of what the user UI might look like.

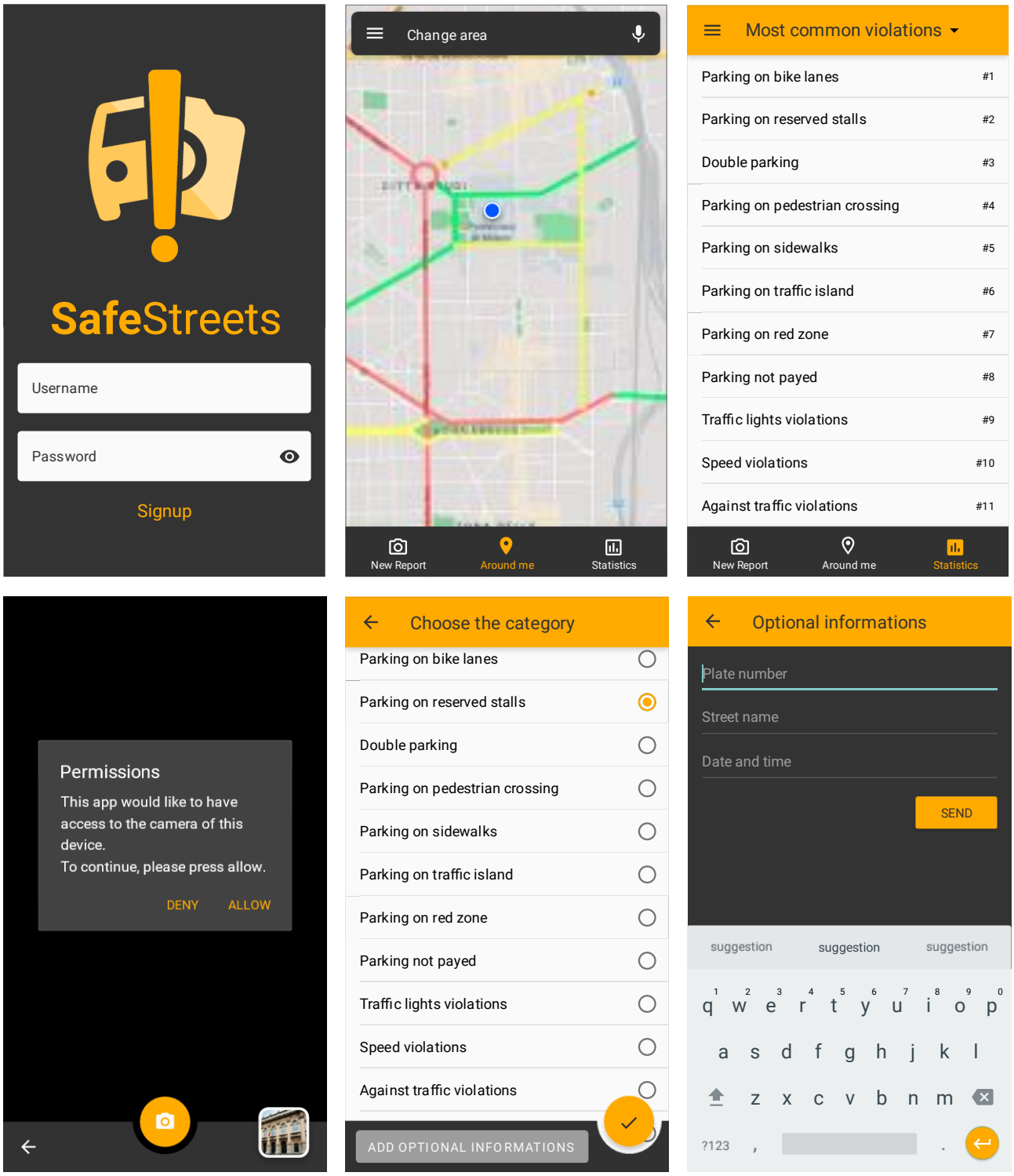

Figure 6 – User mock-ups. From the top left: login, show map, show statistics and three screens of the new report process

When the user opens the app the login screen is displayed, here the user can complete the login operation or the registration. When the user is correctly logged in, the system displays the homepage, where the user can see the area around him if the GPS is available, and some statistics about the most important roads around him. The bottom navigation menu allows the User to send a new report or to ask for some statistics. In the statistics page the user can choose from a menu which type of statistic he wants. In the new report page the user can compile a report of a violation.

The next images show the mock-ups for the municipality.

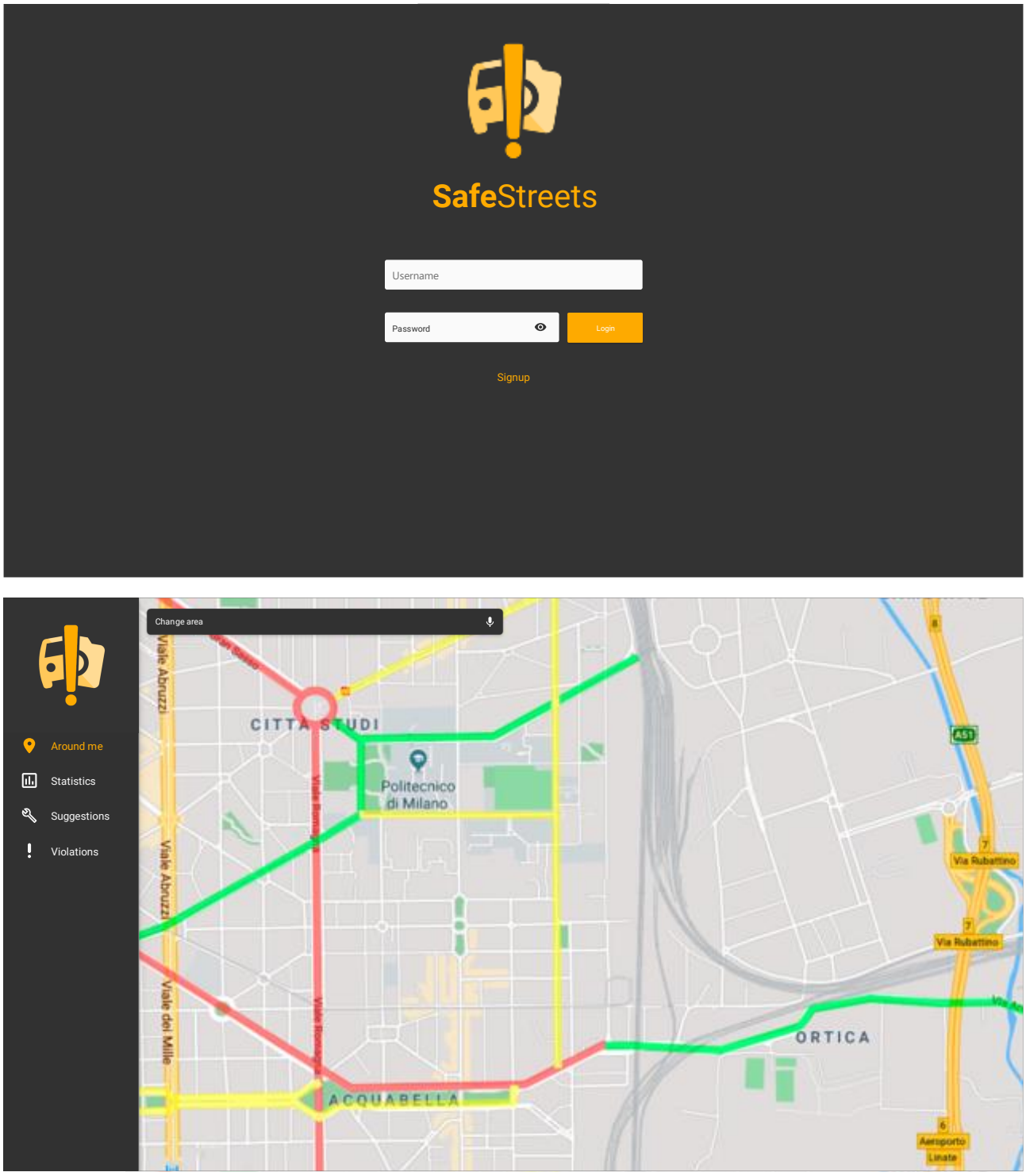

Figure 7 – Municipality mock-ups: login and main screen

3.A.2 Hardware interfaces

The system does not directly access any piece of hardware; however, it will require for the user a mobile device with a camera and possibly GPS capabilities.

3.A.3 Software interfaces

A maps service will be used to handle the maps in the user interface and to translate between coordinates and street names.
An external service will be used to recognize the plate number in the submitted photos.
The user identity verification is left to an external service.

The system will also need interfaces to handle the communication with the municipalities for the data integration, suggestions, and access reports services.

3.A.4 Communication interfaces

The communication is done over the Internet, thus the devices used will need to be connected to the Internet. The handling of the connection is left to the underlying OS.

3.B FUNCTIONAL REQUIREMENTS

3.B.1 Use Case Diagram

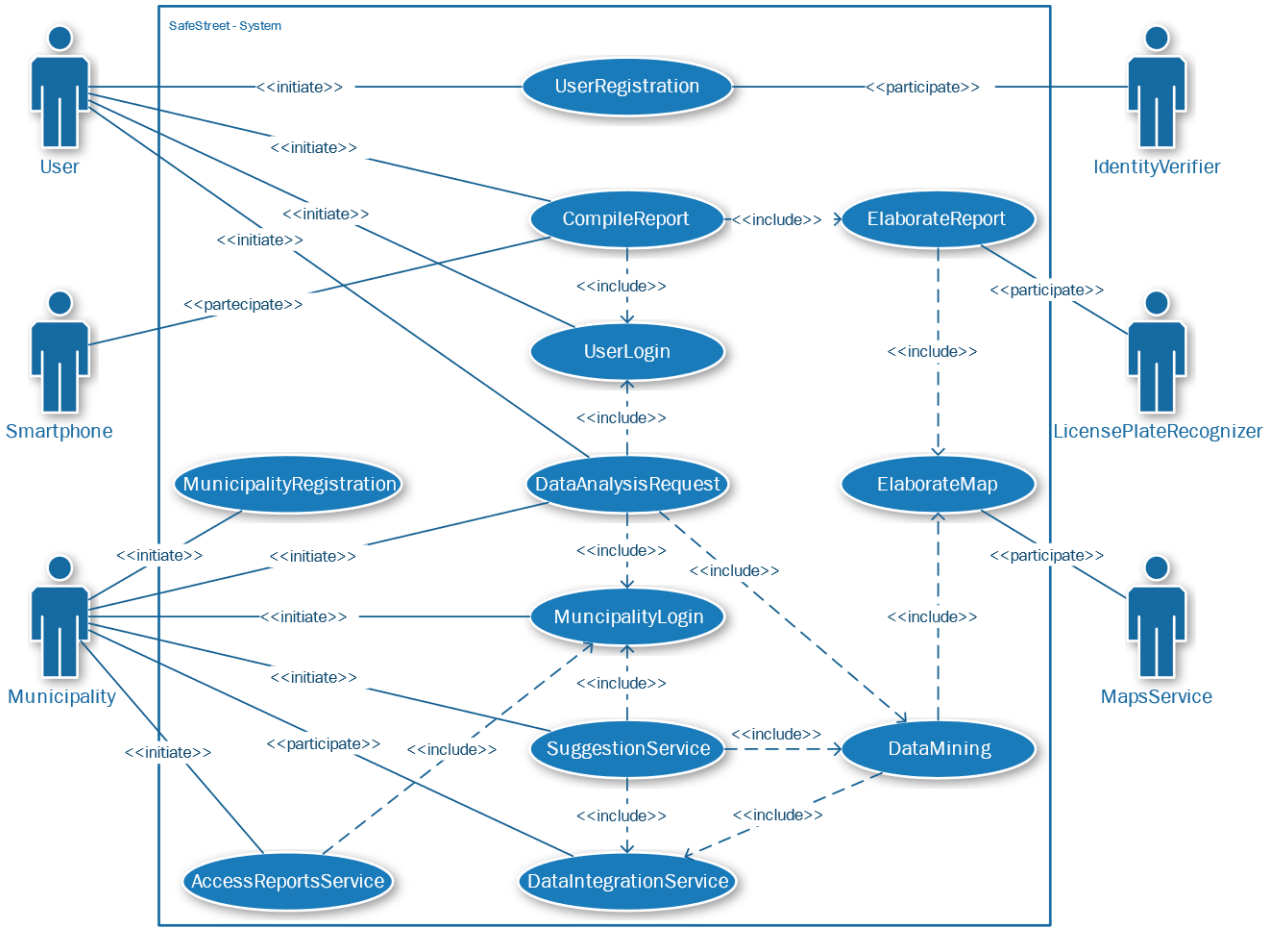

Figure 8 – Use case diagram

The smartphone is considered as an actor because it allows to retrieve GPS coordinates and to take a picture with the camera. The smartphone is also used to retrieve the timestamp of the notification.

In all the use cases that follow it is implicit that the connection can be lost during the flow of events. In this case what happens is that the System will discard the operation and if the User wants to do it again, he must return online and restart the whole operation.

3.B.2 Scenarios

ReportViolation
Bob is moving in the city and he discovers a traffic violation, so he wants to report this violation to SafeStreets. He downloads SafeStreets' app on his smartphone and he registers himself by providing a copy of his identity card and his personal details. After the System has authenticated the User and it has verified Bob's identity, Bob starts to fill the report of the violation (he has to take a photo with his smartphone, acquire the position, …) and when he has finished the report, he sends it to the SafeStreets' System. When the SafeStreets System receives the report, it elaborates it: it sends the picture to the License Plate Recognizer, and it elaborates the position through the Maps Service, then if everything is all right it stores the notification. Otherwise if some errors occur, the System notifies Bob about the error and asks him to redo the operation.

MunicipalityRegistration
The municipality of Monza wants to increase its effectiveness in discovering new violations. Monza's municipality also discovers that the number of people that are using SafeStreets' app in the province is growing very fast. Monza's municipality decides to use the services of SafeStreets, so it contacts SafeStreets' organization to elaborate a contract. In the end it receives a code. In the registration screen, Monza's municipality sends the code to the System, which verifies it and then asks the Municipality the username and the password that will be used for the future login operations. The Municipality specifies its preferences and can now access the violations reported to the SafeStreets' System that are under its authority area. The Municipality also sends to SafeStreets the credential that can be used to access the information about the violations stored by them, if the Municipality uses the data integration service.

UserDataAnalysis
Alice is interested in finding out how the local Municipality of Monza is working on the traffic regulation in her living road. Some of her friends told her about the new possibility to use SafeStreets' app to analyse the violations in the province. She accesses with her account into the app and then she queries the System to find out the most frequent violations that have happened in her living road. The System, after receiving the request, evaluates whether Alice can retrieve the requested data and it mines the information available in his storage; if Monza's municipality is available, the System tries to retrieve the latest violations registered by the Municipality. The presentation of the result is sent to Alice, who can see on a map the result of her request.

MunicipalityTicket
Monza's municipality, after the registration, wants to access the violations, about its authority area, that have been reported to the SafeStreets System (access reports service). In order to do so, the Municipality signs into the System and after it requests for a pull of the violations, the System will check the request of the Municipality and will send only the violations that are under the authority of the Municipality. The Municipality then retrieves the requested violations, it can check them and, if necessary, it will emit a traffic ticket.

RetrieveMunicipalityViolations
Monza's Municipality offers also the possibility to retrieve information about the violations that occurred on its territory and that have been registered by them (data integration service). The SafeStreets System uses the credentials provided by the Municipality to authenticate for the service.

Then the System can access this information when needed to provide the result of some data analysis, that come from the User or from the same Municipality.

ElaborateReport
The System receives a report from Alice. The System then starts to elaborate the report. It first sends a message containing the report's picture with the license plate of the vehicle committing the violation that needs to be recognized to the Plate Recognizer Service. After the response from the Plate Recognizer Service with the retrieved license plate the System contacts also the Maps Service to retrieve the exact position of the violations. When the Maps Service responds with the requested address, the System stores the violations and it informs Alice of the success of the operation.

3.B.3 Use case analysis

U1: User Registration
Actors: User, Identity Verifier.
Entry conditions: The User wants to register himself in the system.
Flow of events:
1. The User sends the request of registration
2. The User starts to fill the registration forms
   2.1. A picture of the User is registered
   2.2. The User inserts his personal details
   2.3. The User provides also his email, username and the password he will use
   2.4. The User inserts a copy of his identity card
3. The user sends the form compiled and the system will take care of it
4. The system validates the user identity and verifies if another user already exists with the same personal details
5. The system contacts the Identity Verifier in order to find out if the document provided by the User is valid
6. When the Identity Verifier has correctly replied with the validated identity of the User, the System sends a notification to the User that the registration has been carried out correctly

Exit conditions: The User has been correctly registered.
Exceptions: Errors occurs if the Identity Document provided by the user cannot be validated by the Identity Verifier, or if there exists another User with the same personal details, so the User receives an error message and he has to retry.

U2: Login
Actors: User or Municipality.
Entry conditions: The User or the Municipality wants to access his account.
Flow of events:
1. The User or the Municipality fills the form with his personal details and tries to sign in
2. The System receives this request and searches the requested account
3. The System verifies if the password is correct for the requested account
4. The System responds to the User or the Municipality and gives it the access to the system's service

Exit conditions: The User is correctly signed into his account.
Exceptions: If the User provides a username that doesn't appears inside the User list, if the provided password is wrong, the User will receive an error messages and he has to retry the login with a different email or password.

U3: Compile report
Actors: User, Smartphone.
Entry conditions: The User wants to notify a violation.

Flow of events:
1. The User signs into the System
2. The User starts to fill a form for the violation notification
3. The System asks the User for a picture of the violation with the license plate of the vehicle
4. The User can decide to select a picture from the gallery or to take one with his camera
5. The System receives the picture inserted by the User and ask the User to fill some required options
   5.1. The System asks the User to input the position of the violations
   5.2. The User can use the position retrieved through his smartphone or he can insert the position manually
   5.3. The System then asks the User to select the violation type
   5.4. The User inserts the violation type
   5.5. Then the System asks the user for some details that are not mandatory
   5.6. The User can respond with the non-mandatory options
6. The System receives the report and the report will be elaborated

Exit conditions: The report has been correctly received by the System.
Exceptions: The User cannot sign into the system, so he will receive an error message and he has to retry the login.

U4: Data Analysis Request
Comprehends also data mining and elaborate map.

Actors: User, Maps Service, Municipality.
Entry conditions: The User or the Municipality ask for an Analysis of the data.
Flow of events:
1. The User or the Municipality signs in
2. The User or the Municipality asks for an analysis of the data
3. The System receives the request and verifies if the User or the Municipality can access the requested data
4. The System retrieves the information and asks the Municipality for its latest data about the violations, if the Municipality of interest uses the data integration service
5. The System mines the information from the retrieved data
6. The System elaborates a graphical representation of the data using the map provided by Maps Service
7. The System sends the result to the User or the Municipality
8. The result is displayed to the terminal of the User or of the Municipality

Exit conditions: The data is correctly visualized by the User or the Municipality.
Exceptions: The System cannot understand the analysis request, the System cannot retrieve enough data for the Data Analysis, the System cannot access its data on the violations or the data of the Municipality, the system cannot interpret the data that it has retrieved (for example Maps Service cannot understand the position), the User or the Municipality has the wrong access rights for the requested data analysis, in all of this case the operation is aborted by the System and the User or the Municipality needs to redo the request.

U5: Access Reports Service
Actors: Municipality.
Entry conditions: The Municipality wants to retrieve the violations notified to the System (access reports service).
Flow of events:
1. The Municipality signs in
2. The Municipality asks for an update of the latest notified violations to the System or it asks to retrieve some violations

3. The System elaborates the request, retrieves all the latest violations concerning the Municipality, and verifies the access rights to the violations
4. The System sends the latest violations to the Municipality
5. The Municipality accesses the update sent by the System

Exit conditions: The Municipality gets the latest violations concerning its authority.
Exceptions: The Municipality cannot sign in.

U6: Municipality Registration
Actors: Municipality.
Entry conditions: The Municipality wants to perform a registration.
Flow of events:

1. The Municipality requests a registration operation
2. The System asks the Municipality to insert its contract code
3. The System verifies the contract code of the Municipality
4. The System asks the Municipality to select the credential for the next login operations
5. The Municipality sends the required credentials
6. The System memorizes the credentials chosen by the Municipality
7. The System asks the Municipality if they offer a service to retrieve the violations (data integration service)
   7.1. If so, the Municipality sends the credential that the System can use to access this service
   7.2. Then the System verifies the credentials provided by the Municipality
8. The Municipality is informed of the operation's success

Exit conditions: The Municipality has been correctly registered and can access the services of the System.
Exceptions: If the Municipality provides a wrong code it is asked again; if the Municipality provides wrong credentials to access their service (if they provide the data integration service), the Municipality receives an error messages, and the operation needs to be redone.
Before the registration: The Municipality contacts an employee of SafeStreets in order to get a contract code for the registration, which is provided when the two parts sign a contract. This operation is not managed by the system, so it is not included in the use case and sequence diagram.

U7: Elaborate Report
Actors: Maps Service, License Plate Recognizer
Entry conditions: The system wants to elaborate a report.
Flow of events:

1. The System starts the elaboration of a report
2. The System sends the report's picture with the plate of the vehicle committing the infraction to the License Plate Recognizer.
3. The License Plate Recognizer sends back the required plate to the System
4. The System asks to the Maps Service to provide the address of the violation or street name
5. The Maps Service sends back the response to the System
6. The System then stores the violation

Exit conditions: The violation has been correctly stored.
Exceptions: The License Plate Recognizer or the Maps Service are unavailable, the System will try to contact them in future to validate the Report.

U8: Suggestion Service
Actors: Municipality.
Entry conditions: The System notifies the Municipality about possible interventions (suggestion service).
Flow of events:

1. The Municipality logs into the system
2. The Municipality wants to get some suggestions
3. The system calculates if some suggestions are available
   3.1. The system verifies the competence area of the Municipality
   3.2. The system finds the most common violations in a certain area
   3.3. The system tries to find a possible solution to the problem
4. The Municipality receives the suggestions and can ask to get more of them
   4.1. If the Municipality asks for more suggestions, then system calculates them, and sends the suggestions to the Municipality

Exit conditions: The Municipality doesn't want other suggestions, or there are no other suggestions.
Exceptions: The Municipality cannot sign in.

3.B.4 Sequence Diagrams

Login (U2)

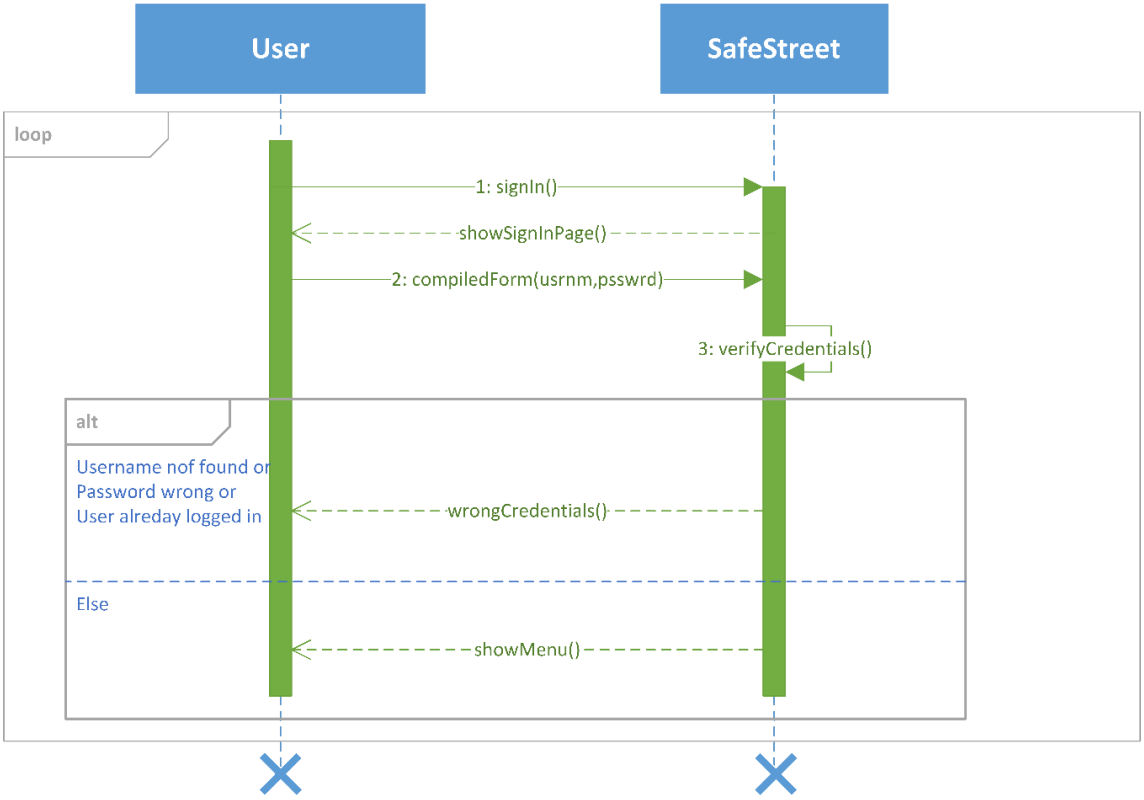

Figure 9 – Login sequence diagram

This sequence diagram represents the login operation: in this case the operation is carried out by the User, but the Municipality can access the System in the same way. After this operation, the User and the Municipality will be authenticated, and they can use the services exposed by SafeStreets.

Report (U3)

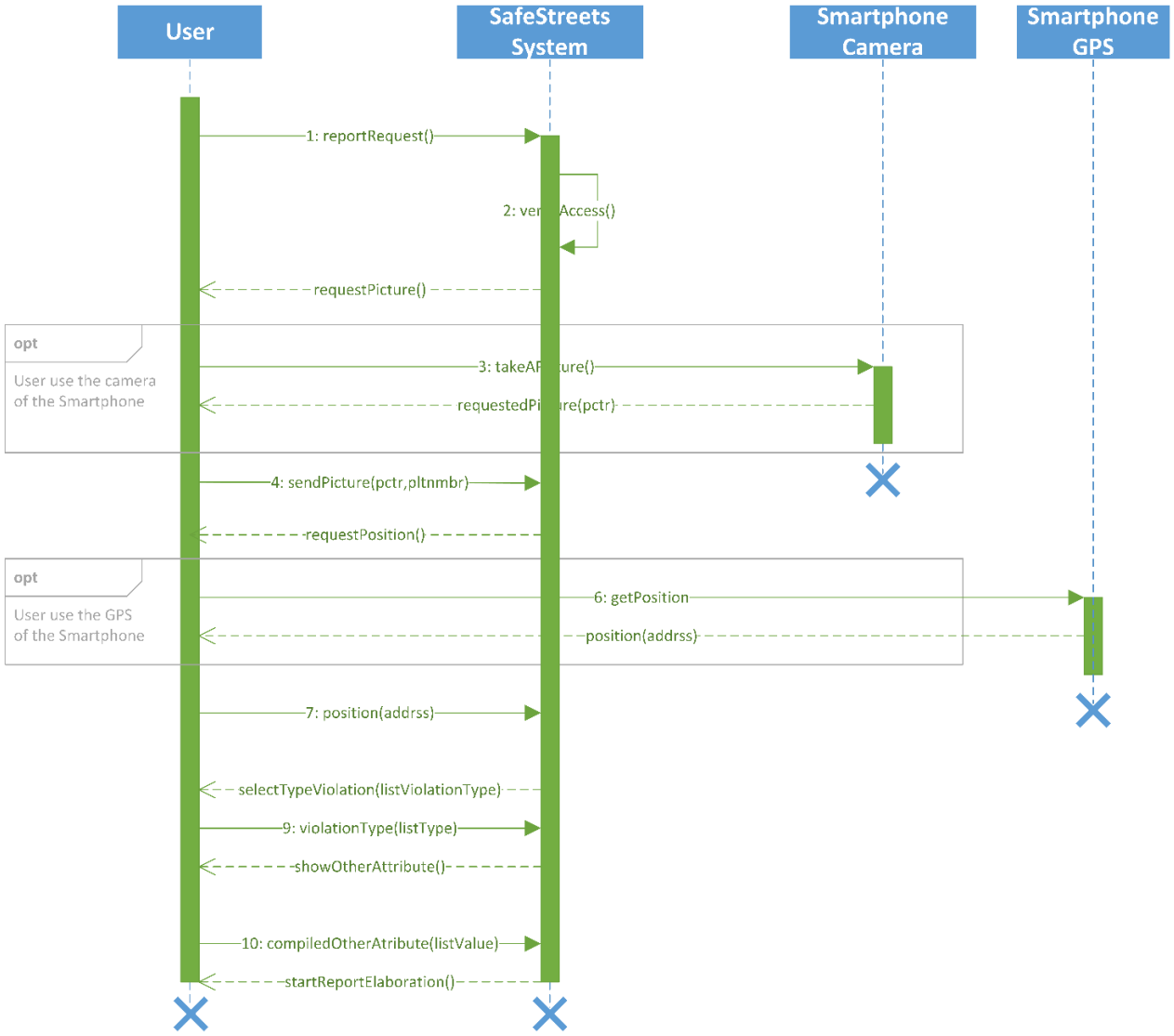

Figure 10 – Report sequence diagram

This sequence diagram represents the sequence of operations carried out by the User to fill the form of a report. The smartphone is represented as an actor because is used to collect the data about the User's position and to take a picture about the violations. This operation can be performed after the *Login* operation that is presented above.
Also the last response message indicates that the SafeStreets' System will take care of the elaboration of the report that is presented in the sequence diagram *Elaboration* below.

Elaboration (U7)

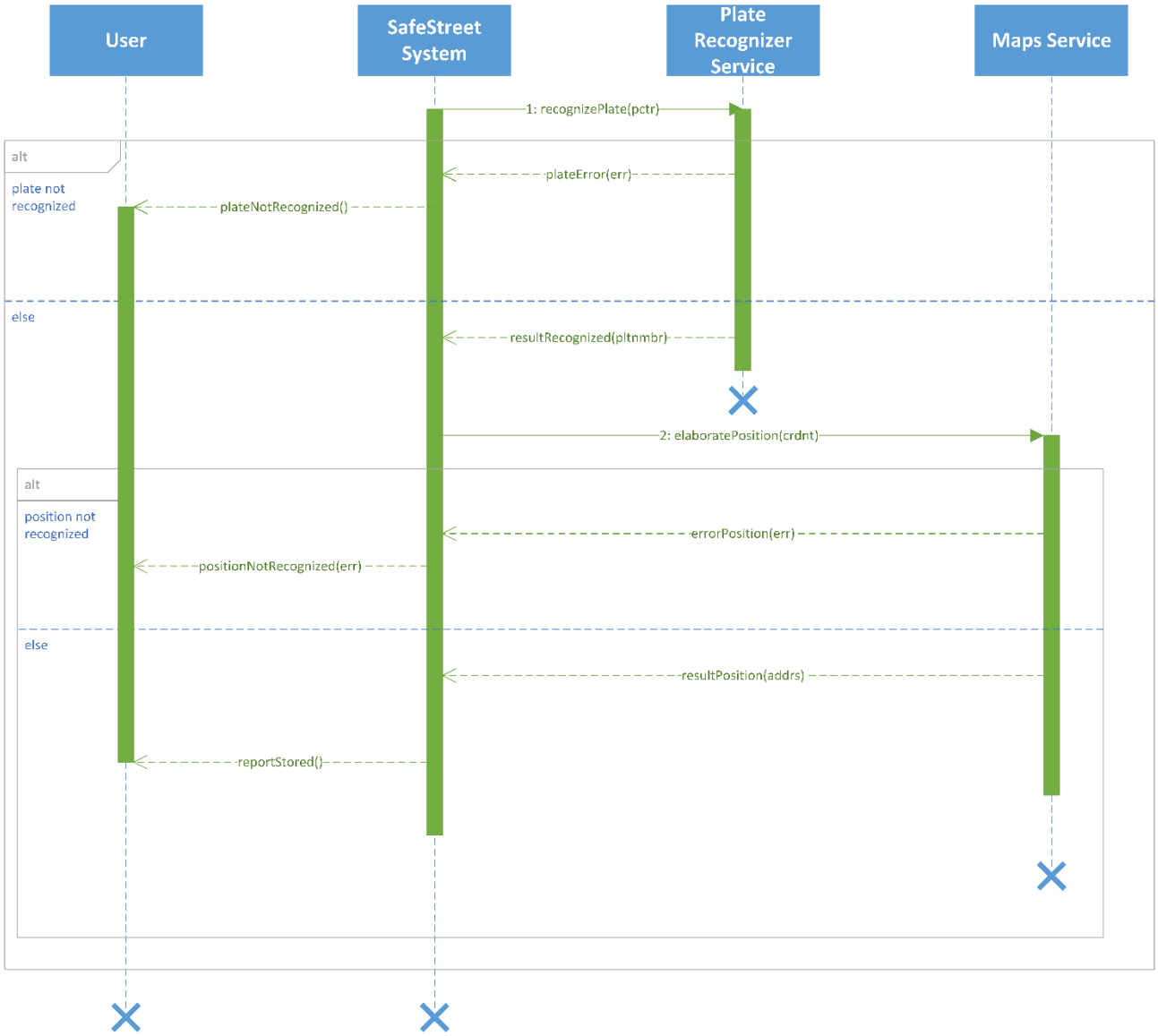

Figure 11 – Elaboration sequence diagram

This sequence diagram represents the sequence of operations that will be carried out by the System to elaborate the report sent by the User. This sequence of operations happens after the User has correctly submitted the report, like in the *Report* sequence diagram presented above. The SafeStreets' System uses the services exposed by the Plate Recognizer Service in order to retrieve the License Plate of the vehicle, and the Maps Service to get the address of the violations.

Request Data Analysis (U4)

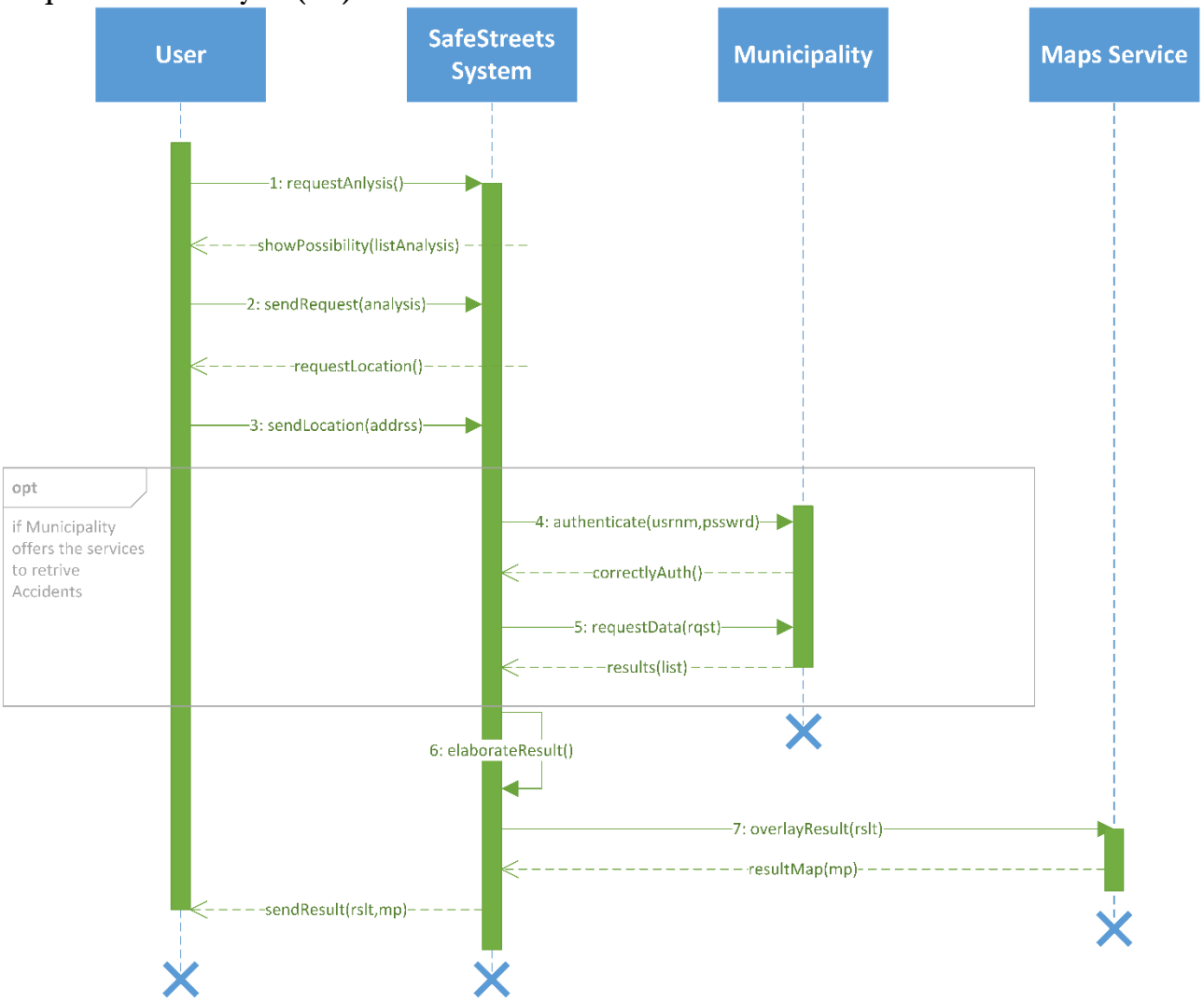

Figure 12 – Request data analysis sequence diagram

This sequence diagram represents the sequence operations done to perform a data analysis request made by the User. As before this operation can be carried out if the user has been correctly *logged* in the system. This sequence of operations performed by the User, can be carried out also from the Municipality.

Municipality Registration (U6)

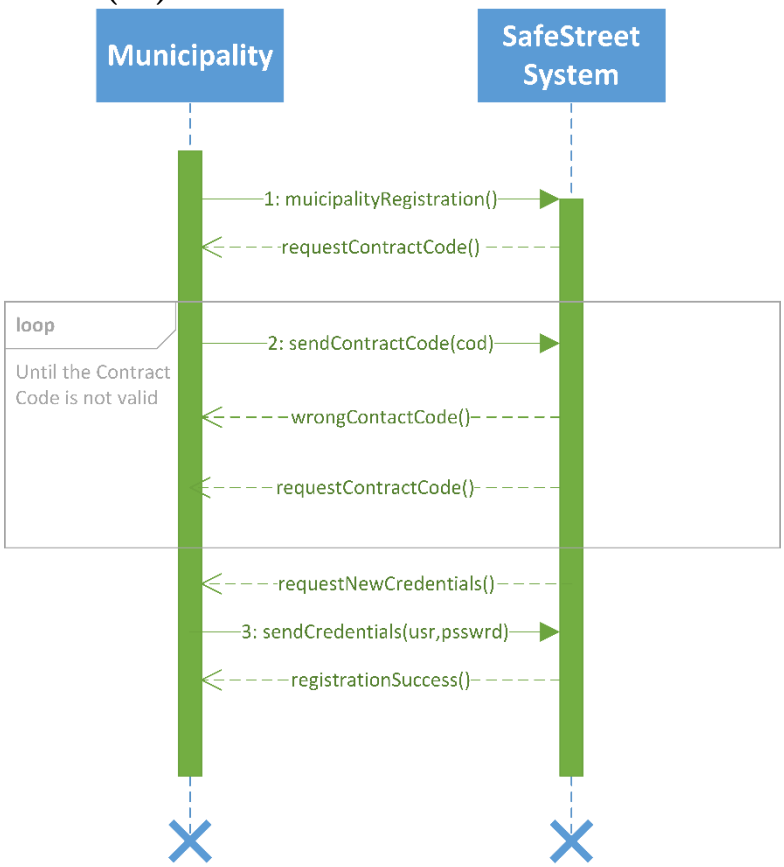

Figure 13 – Municipality registration sequence diagram

This sequence diagram represents the sequence of operations for the Municipality registration. After this the Municipality can correctly use the SafeStreets' services. The Municipality must insert the contract code that has been previously obtained from SafeStreets, after the contract was signed.

User Registration (U1)

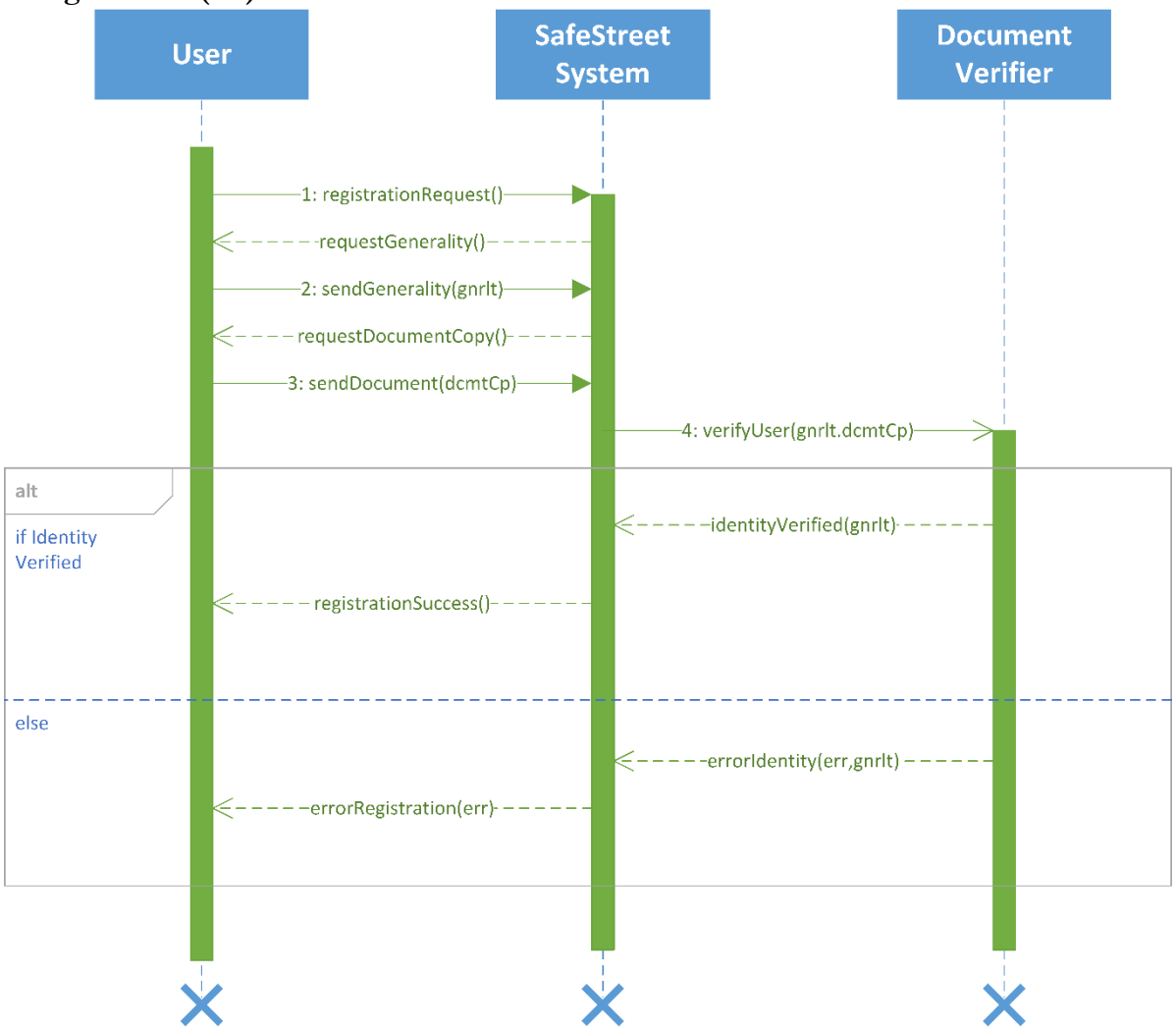

Figure 14 – User registration sequence diagram

In this sequence diagram are represented the operations done to perform the registration of a User. After this operation the User can *login* the SafeStreets' System and can use all the services exposed by the System.

Municipality requests violations (U5)

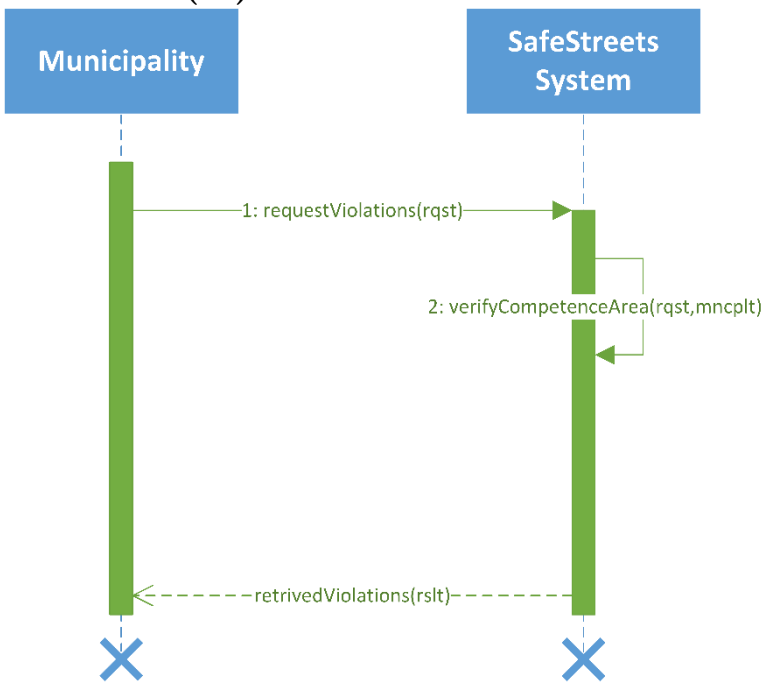

Figure 15 – Municipality requests violations

This sequence diagram happens after the Municipality has been correctly *logged* in the system as shown in the previous sequence diagram. The municipality can access all the violations of his competence area, so the SafeStreets' System needs to find out what request can be displayed to the Municipality, by looking at the location of the Municipality and of the violations.

3.B.5 Traceability Matrix

The following traceability matrix keeps track of the relationship between the use cases and the requirements.

REQUIREMENTS	USE CASE
R1	*U4: Data Analysis Request* *U5: Access Reports Service* *U7: Elaborate Report*
R2	*U4: Data Analysis Request*
R3	*U4: Data Analysis Request* *U5: Access Reports Service*
R4	*U5: Access Reports Service*
R5	*U4: Data Analysis Request* *U5: Access Reports Service* *U7: Elaborate Report*
R6	*U3: Compile Report*
R7	*U4: Data Analysis Request*
R8	*U3: Compile Report*
R9	*U3: Compile Report* *U7: Elaborate Report*
R10	*U4: Data Analysis Request*
R11	*U4: Data Analysis Request*
R12	*U8: Suggestion Service*
R13	*U7: Elaborate Report*
R14	*U1: User Registration* *U2: Login*
R15	*U6: Municipality Registration* *U2: Login*
R16	*U3: Compile Report*
R17	*U1: User Registration*
R18	*U7: Elaborate Report*
R19	*U4: Data Analysis Request* *U5: Access Reports Service* *U7: Elaborate Report* *U8: Suggestion Service*

3.B.6 Mapping

In this section for each goal are listed the related requirements and domain assumptions.

- **G1: The System accepts valid reports by the users about the parking violations.**
    - R1: The reports about the violations are correctly stored.
    - R5: The system must avoid the manipulation of the violations.
    - R6: The system must be able to retrieve the position from the user or from the GPS
    - R8: The system must allow the User to take a picture or select one from the device.
    - R9: The system accepts reports from the User.
    - R13: The System accepts only reports with a valid plate number and position.
    - R14: The System must allow the user to perform the registration and the login.
    - R16: The System must ask the User the non-mandatory attributes of the report.
    - R17: The system must communicate with the Identity Verifier.

  - R18: The system must communicate with the Plate Recognizer Service.
  - R19: The System must communicate with the Maps Service.
  - D1: If the License Plate Recognizer recognise the license plate, then the result is correct.
  - D2: If the map service recognises the street name from the coordinates, then the result is correct.
  - D3: The Identity card is correctly verified.
  - D5: The Data retrieved by the Smartphone's GPS is correct.
  - D6: The time reported by the User's smartphone is correct.
  - D7: The user does not send false reports.
- **G2: The System suggests possible interventions to the Municipality.**
  - R1: The reports about the violations are correctly stored.
  - R5: The system must avoid the manipulation of the violations.
  - R11: The municipality can view all the statistics calculated by the system.
  - R12: The System must suggest interventions to the Municipality.
  - R15: The System must allow the municipality to perform the registration and the login.
  - R19: The System must communicate with the Maps Service.
  - D7: The user does not send false reports.
- **G3: The System allows the Municipality to retrieve submitted parking violations of its competence area.**
  - R1: The reports about the violations are correctly stored.
  - R3: The Municipality can access only the data of the violations of its competence area.
  - R5: The system must avoid the manipulation of the violations.
  - R7: Only the Municipality can access the submitted parking violation of its competence area
  - R15: The System must allow the municipality to perform the registration and the login.
  - R19: The System must communicate with the Maps Service.
  - D7: The user does not send false reports.
- **G4: The System gives some statistics to the User about the violations.**
  - R1: The reports about the violations are correctly stored.
  - R2: The user can view the statistics calculated by the System with some exceptions.
  - R5: The system must avoid the manipulation of the violations.
  - R14: The System must allow the user to perform the registration and the login.
  - R10: The System must calculate some statistics.
  - R19: The System must communicate with the Maps Service.
  - D7: The user does not send false reports.
- **G5: The System can give all the statistics to the Municipality about the violations.**
  - R1: The reports about the violations are correctly stored.
  - R5: The system must avoid the manipulation of the violations.
  - R10: The System must calculate some statistics.
  - R11: The municipality can view all the statistics calculated by the system.
  - R15: The System must allow the municipality to perform the registration and the login.
  - R19: The System must communicate with the Maps Service.
  - D7: The user does not send false reports.
- **G6: The System can retrieve the violations verified by the Municipality.**

- R4: Violations registered by the Municipality can be retrieved by the system.
- R5: The system must avoid the manipulation of the violations.
- R15: The System must allow the municipality to perform the registration and the login.
- D4: The Municipality possesses only real violations.

3.C PERFORMANCE REQUIREMENTS

The software should be used without waiting times; an exception can be made when the application is uploading the report or when is downloading the statistics from the server, because these functions rely on the speed of the internet connection.

3.D DESIGN CONSTRAINTS

3.D.1 Standard compliance

In accordance with GDPR, all personal and sensitive data will be secured both in the transmission and in the storing; users will be asked consent to acquire and use their data.

The system will adhere to the *material* usability standards provided by Google and will follow the *material* guidelines for the other aspects of the system front end.

3.D.2 Hardware limitations

The system for the user will not need a powerful device to run, but will rely on a stable enough Internet connection, a device camera with a resolution sufficient to allow the plate recognition by the Plate Recognizer, and GPS functionalities if possible.

The system for the Municipality will need only an Internet connection stable enough.

The system for the backend will need to cope with the volume of requests.

3.D.3 Any other constraint

The system will ensure different visibility levels for the user and the municipality about the violations and the statistics, according to what already stated in this document.

3.E SOFTWARE SYSTEM ATTRIBUTES

3.E.1 Reliability

The system will be designed to be run continuously in absence of external faults. The data provided by the system will always be reliable; to achieve this the data will be handled securely, and the system will rely on the underlying layers for the other implementation aspects.

3.E.2 Availability

The system does not need extremely high availability, since its services are not critical. The system will strive for 99% availability. In the future a higher level could be achieved by duplicating the database over multiple locations, or by duplicating the back-end logic, for example using executor pools and multiple physical servers.

3.E.3 Security

Since the data from the reports can be used by the municipality to generate traffic tickets, it is important that the chain of custody of the information is never broken. This means that the confidentiality and integrity of the data must be ensured both in its transmission and in its storage.

The system will ensure security by encryption, authentication and different levels of authorizations.
All the communications will be encrypted, preventing interception and modification of the data.
Anonymous access is forbidden: the authentication will prevent fabrication of data and will allow to assign different authorizations to the two actors.
The system will rely on the underlying abstraction levels for these functionalities and for the protection of the data stored.

3.E.4 Maintainability

The system will be designed to be highly maintainable through extensible design solutions – especially for what concerns the definition of new statistics, suggestions and types of violations – and readable and testable code in the implementation phase.

3.E.5 Portability

The system will be built entirely on an underlying abstraction layer (a virtual machine, like Java, or an engine). This, provided the minimum hardware requirements, should ensure a sufficient level of portability for the system.

4 FORMAL ANALYSIS USING ALLOY

In this chapter, the most important concepts represented in the class diagram of the section 2.A have been modelled through Alloy. The objective of this chapter is to show the consistency of the model and to prove the correctness of some goals.

The entities have been represented through signatures. Some requirements have been represented with facts.

The goals 1, 2, 4 and 5 have been modelled with one predicate for each one. There is also one predicate for these goals all together.

The goals 3 and 6 have been modelled with one assertion for each one.

4.A GENERATED WORLDS

G1: The System accepts valid reports by the users about the parking violations.

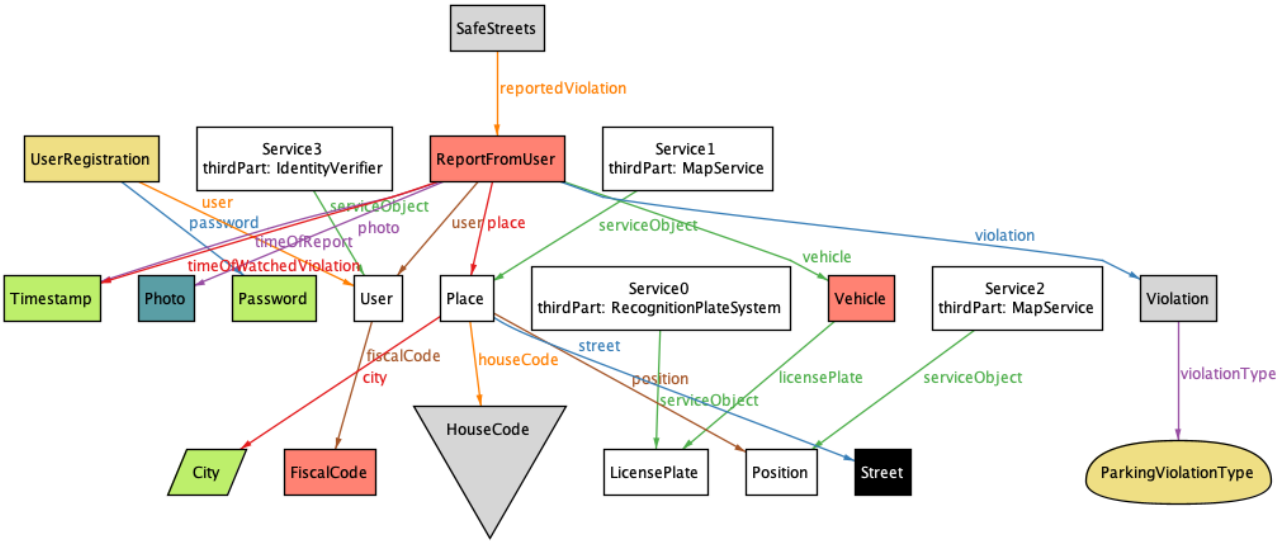

Figure 16

This world shows that there is at least one report (ReportFromUser) made by one user registered in SafeStreets.
In particular the report contains various information: the timestamp of the report, the timestamp of when the violation was seen, some photos, the user that has done the report, the place of the violation and the vehicle that has done the violation.
The user is identified by the Identity Verifier with the Service3. The place and the position are identified by the map service and then the license plate is identified by the recognition plate system. Finally, the place contains the specific position, the city, the street and the house code.

G2: The System suggests possible interventions to the Municipality.

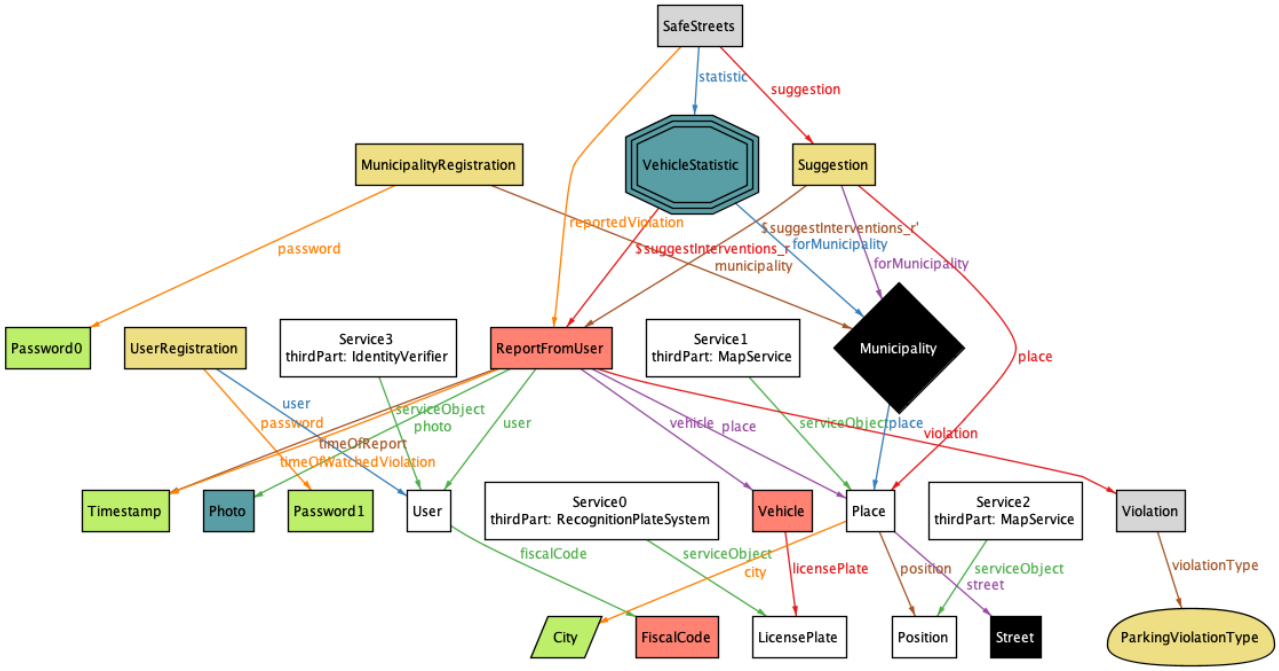

Figure 17

Here it is possible to see that SafeStreets has generated one suggestion for a Municipality in a specific place. The suggestion was made based on the reports stored in SafeStreets.

G4: The System gives some statistics to the User about the violations.

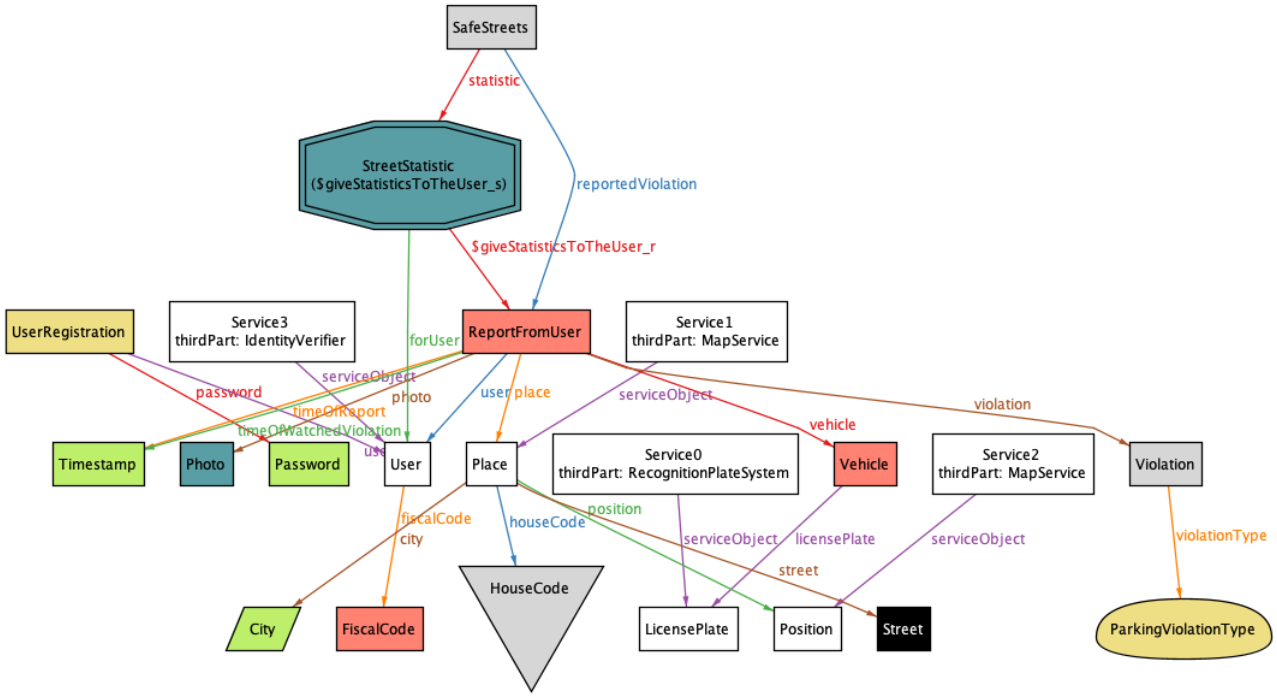

Figure 18

This world shows one statistic about the streets made by SafeStreets for a user.

G5: The System can give all the statistics to the Municipality about the violations.

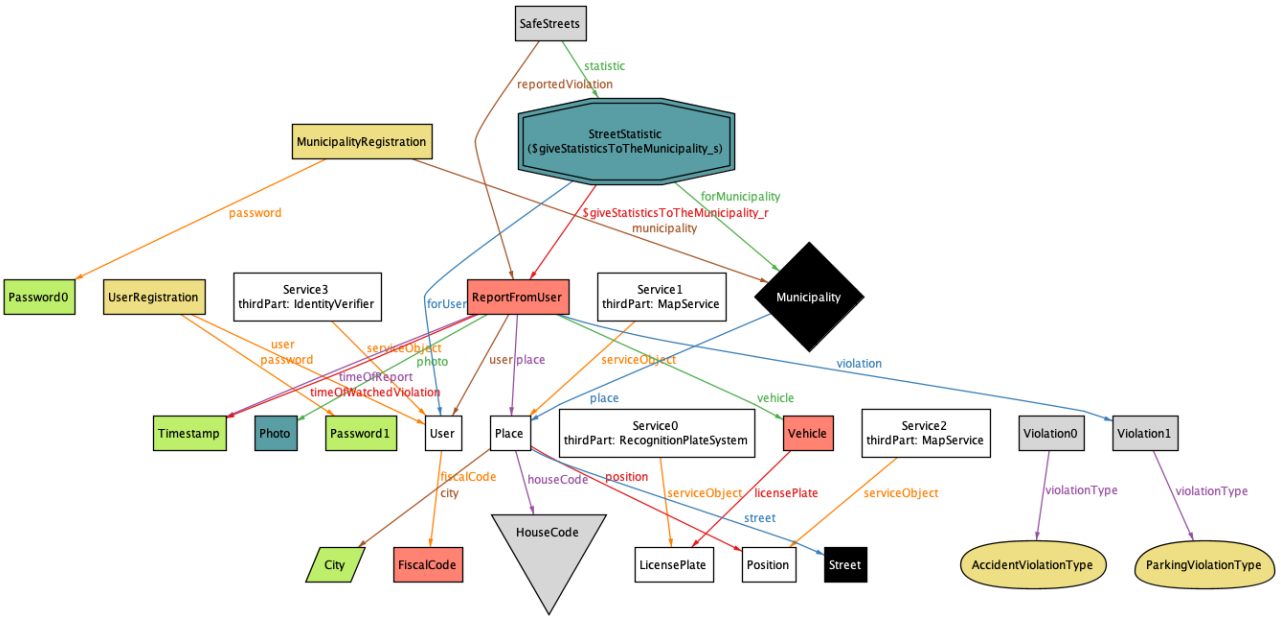

Figure 19

This world shows one statistic about the streets made by SafeStreets for a Municipality.

G1 and G2 and G4 and G5:

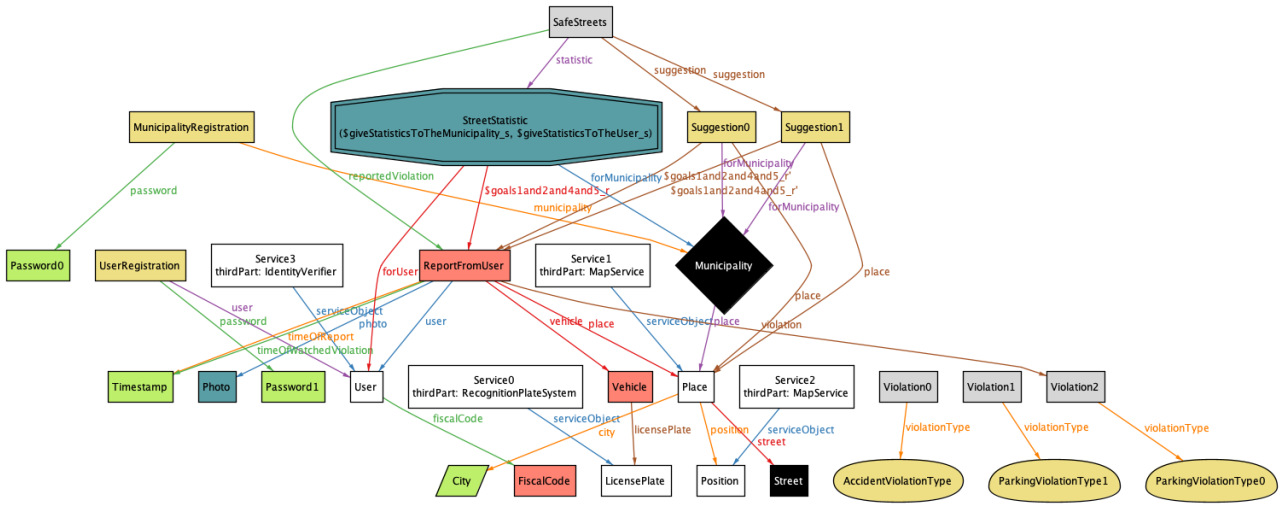

Figure 20

This world satisfies the goals 1, 2, 4 and 5.
There is one report made by a user, one statistic for one user, one statistic for one Municipality and one suggestion for one Municipality.

World with no Municipalities:

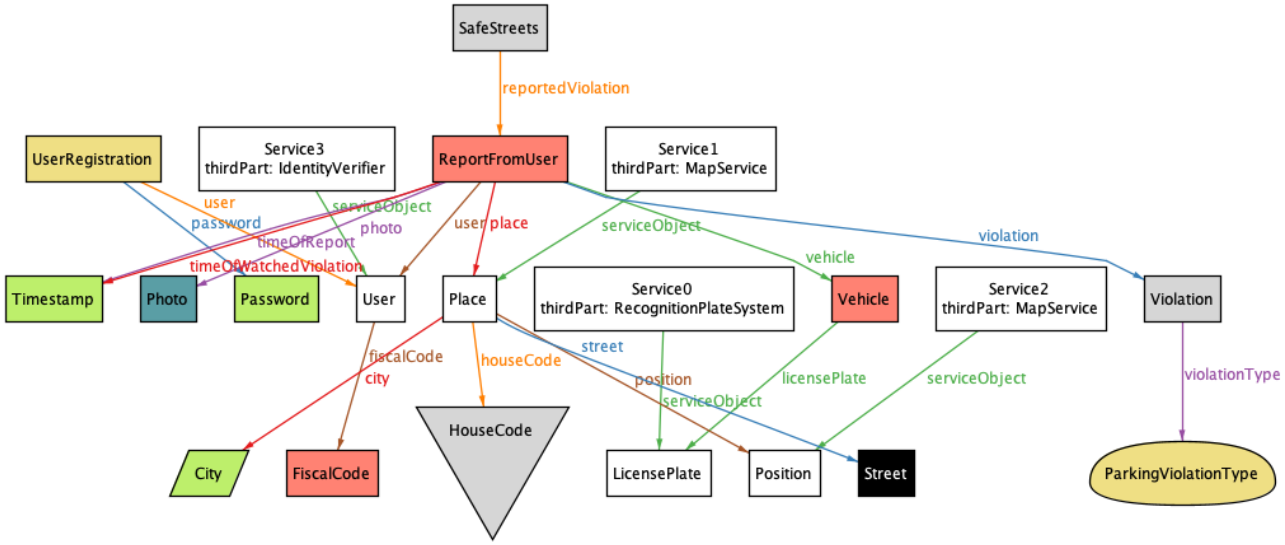

Figure 21

This world shows the consistency of the model even when there are no Municipalities registered in the SafeStreets' system.

World with one Municipality:

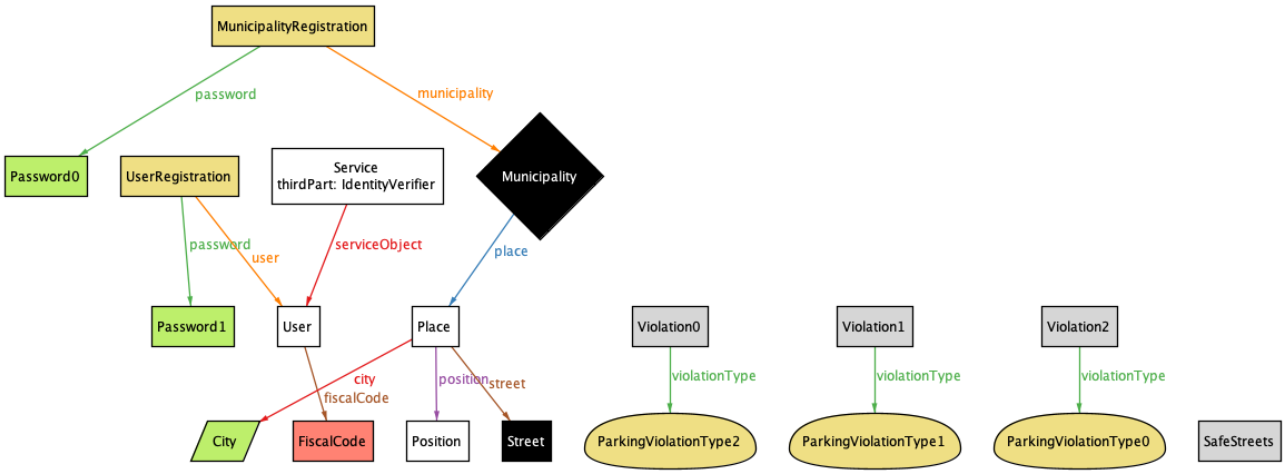

Figure 22

In this world there is one Municipality registered in the system. In this case there aren't report from the user yet.

World with more Municipalities:

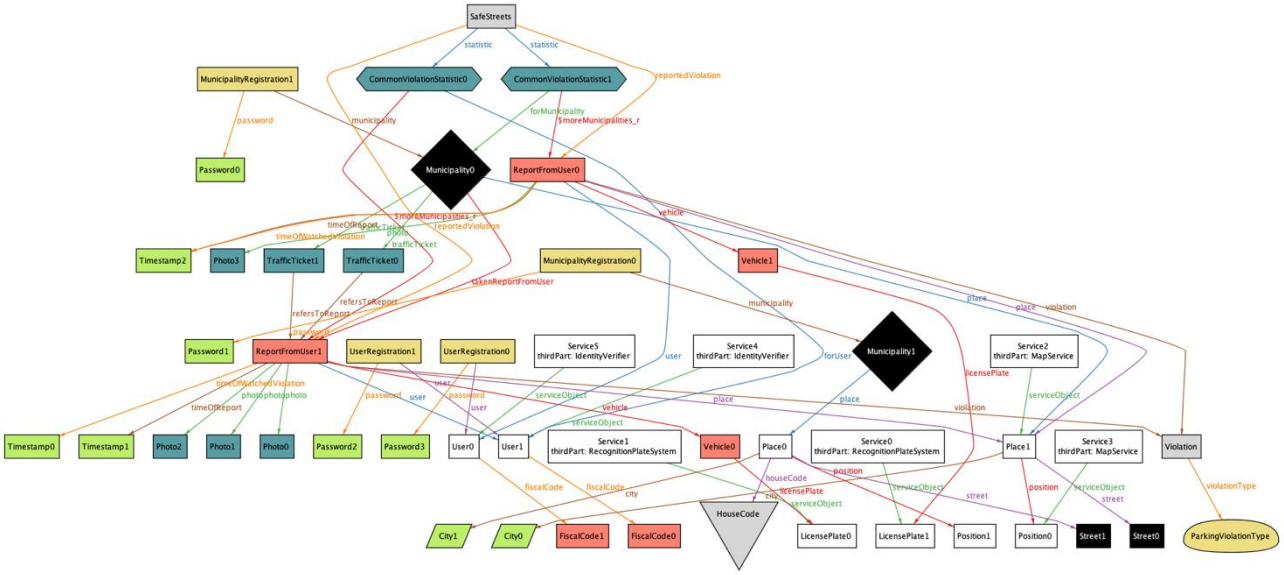

Figure 23

This world is the most complete because it contains two registered Municipalities and two reports made by two users.

4.B RESULTS

The next image shows the results of the predicates and of the assertions.

All the predicates are consistent, and Alloy has not found any counterexample on the assertions.

```
Executing "Run acceptCompleteReports for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  8677 vars. 584 primary vars. 17508 clauses. 18ms.
  Instance          found. Predicate is consistent. 20ms.

Executing "Run suggestInterventions for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  8677 vars. 584 primary vars. 17508 clauses. 17ms.
  Instance          found. Predicate is consistent. 24ms.

Executing "Check theMunicipalityCanRetrieveSubmittedViolations"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4942 vars. 360 primary vars. 9405 clauses. 13ms.
  No counterexample found. Assertion may be valid. 0ms.

Executing "Run giveStatisticsToTheUser for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  8729 vars. 588 primary vars. 17595 clauses. 20ms.
  Instance          found. Predicate is consistent. 24ms.

Executing "Run giveStatisticsToTheMunicipality for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  8729 vars. 588 primary vars. 17595 clauses. 18ms.
  Instance          found. Predicate is consistent. 22ms.

Executing "Check safeStreetsCanRetrieveViolationsFromTheMunicipality"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4954 vars. 361 primary vars. 9330 clauses. 14ms.
  No counterexample found. Assertion may be valid. 4ms.

Executing "Run goals1and2and4and5 for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  8809 vars. 592 primary vars. 17762 clauses. 24ms.
  Instance          found. Predicate is consistent. 26ms.

Executing "Run noMunicipality for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  8687 vars. 584 primary vars. 17550 clauses. 23ms.
  Instance          found. Predicate is consistent. 23ms.

Executing "Run oneMunicipality"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4850 vars. 354 primary vars. 9154 clauses. 12ms.
  Instance          found. Predicate is consistent. 9ms.

Executing "Run moreMunicipalities for 12"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  97721 vars. 4440 primary vars. 255926 clauses. 239ms.
  Instance          found. Predicate is consistent. 1393ms.

10 commands were executed. The results are:
  #1: Instance found.                  acceptCompleteReports is consistent.
  #2: Instance found.                  suggestInterventions is consistent.
  #3: No counterexample found. theMunicipalityCanRetrieveSubmittedViolations may be valid.
  #4: Instance found.                  giveStatisticsToTheUser is consistent.
  #5: Instance found.                  giveStatisticsToTheMunicipality is consistent.
  #6: No counterexample found. safeStreetsCanRetrieveViolationsFromTheMunicipality may be valid.
  #7: Instance found.                  goals1and2and4and5 is consistent.
  #8: Instance found.                  noMunicipality is consistent.
  #9: Instance found.                  oneMunicipality is consistent.
  #10: Instance found.                  moreMunicipalities is consistent.
```

4.C ALLOY CODE

Here is the code of the Alloy model.

```
//Italian fiscal code
sig FiscalCode {}

//Object of a service offered by a third part
abstract sig ServiceObject{}

//User of the SafeStreets' System
//He/she must be verified by a third part
sig User extends ServiceObject {
   fiscalCode: one FiscalCode
}

//Password of the registration in SafeStreets' System
sig Password {}

//Photo of a violation
sig Photo {}
//Time and date
sig Timestamp {}

//Specific position in the world, identified by some coordinates
sig Position extends ServiceObject {}
//House code
sig HouseCode{}
//Street
sig Street{}
//City
sig City{}
//Specific  place in the world, identified by a city, an house code and a street or by a position
sig Place extends ServiceObject {
   city: one City,
   houseCode: one HouseCode,
   street: one Street,
   position: one Position
}

//Type of violation
abstract sig ViolationType {}
//Type of violation referring to a parking violation
sig ParkingViolationType extends ViolationType {}
//Type of violation referring to an accident
sig AccidentViolationType extends ViolationType {}

//License plate of a vehicle
sig LicensePlate extends ServiceObject {}
//Vehicle
sig Vehicle {
   licensePlate: one LicensePlate
}
```

```
//Report referring to a violation
abstract sig Report {
    //Timestamp of when the violation was seen, if it is empty it means that it is equal to timeOfReport
    timeOfWatchedViolation: lone Timestamp,
    //Timestamp of when the report was made
    timeOfReport: one Timestamp,
    //Vehicle that has done the violation
    vehicle: one Vehicle,
    //Place where the violation was seen
    place: one Place,
    violation: one Violation
}

//Report made by an user
sig ReportFromUser extends Report {
    user: one User,
    //Photos made by the user
    photo: some Photo
}

//report made by a Municipality
sig ReportFromMunicipality extends Report {

}

//Real violation that happened in the world
sig Violation {
    violationType: one ViolationType
}

//Municipality
sig Municipality {
    //Place where the Municipality resides
    place: one Place,
    //TrafficTickets made by the Municipality
    trafficTicket: set TrafficTicket,
    //Reports taken by the Municipality from Safestreets
    takenReportFromUser: set ReportFromUser
}

//Registration to the SafeStreets' System
abstract sig Registration {
    password: one Password
}

//Registration of an user
sig UserRegistration extends Registration {
    user: one User
}

//Registration of a Municipality
sig MunicipalityRegistration extends Registration {
```

```
    municipality: one Municipality
}

//SafeStreets' System
one sig SafeStreets {
    suggestion: set Suggestion,
    statistic: set Statistic,
    reportedViolation: set ReportFromUser,
    //Reports taken from the municipality
    takenReportFromMunicipality: set ReportFromMunicipality
}

//Third part that makes a service for SafeStreets
abstract sig ThirdPart{}
//Verifier of the identity of an user
one sig IdentityVerifier extends ThirdPart {}
//Recognizer of the traffic plate of a car
one sig RecognitionPlateSystem extends ThirdPart {}
//Third part that identifies the position and/or the street of a violation
one sig MapService extends ThirdPart {}

//Service made by a third part on an object
sig Service {
    serviceObject: one ServiceObject,
    thirdPart: one ThirdPart
}
//Traffic ticket
sig TrafficTicket {
    refersToReport: one Report
}
//Statistic
abstract sig Statistic {
    forUser: lone User, //means that the statistic is for a user
    forMunicipality: lone Municipality //means that the statistic is for a Municipality
} {
    //a statistic is always done for a user or for a Municipality
    #forUser=1 or #forMunicipality=1
}
//Statistic about the streets
sig StreetStatistic extends Statistic{}
//Statistic about the effectiveness of the service of SafeStreets
sig EffectivenessOfServiceStatistic extends Statistic{}
//Statistic about the vehicles
sig VehicleStatistic extends Statistic{}
//Statistic about the common violations
sig CommonViolationStatistic extends Statistic{}

//Suggestion for the Municipality
sig Suggestion {
    //Place where the suggestion refers
    place: one Place,
    forMunicipality: one Municipality
```

```
}


--Facts:--
//Each user has an unique fiscal code
fact eachUserHasAnUniqueFiscalCode {
   all disj u1, u2 : User | u1.fiscalCode != u2.fiscalCode
}

//Each fiscal code is associated to an user
fact eachFiscalCodeToAnUser {
   all f: FiscalCode | one u: User | u.fiscalCode=f
}

//Each Municipality resides on a different city
fact eachMunicipalityIsInADifferentCity {
   all disj m1, m2: Municipality | m1.place.city != m2.place.city
}

//Each place has a different position
fact eachPlaceHasDifferentPosition {
   all disj p1, p2: Place | p1.position!=p2.position and (p1.city!=p2.city or p1.houseCode!=p2.houseCode or
      p1.street!=p2.street)
}

//Each vehicle has a unique license plate
fact licencePlateIsUnique {
   all disj v1, v2: Vehicle | v1.licensePlate!=v2.licensePlate
}

//Each license plate is associated to a vehicle
fact eachLicencePlateToAVehicle {
   all l: LicensePlate | one v: Vehicle | v.licensePlate=l
}

//Each traffic ticket was made by one and only one Municipality
fact trafficTicketMadeByOneMunicipality {
   all disj m1, m2: Municipality | m1.trafficTicket & m2.trafficTicket = none

   all t: TrafficTicket | one m: Municipality | t in m.trafficTicket
}

//Each photo, timestamp and vehicle are associated to a report
fact eachPhotoTimeVehicleToAReport {
   all p: Photo | (one r: ReportFromUser | p in r.photo)
   all t: Timestamp | (one r: Report |  r.timeOfWatchedViolation=t or r.timeOfReport=t)
   all ve: Vehicle | (one r: Report |  r.vehicle=ve)
}

//Each violation type is associated to a violation
fact eachViolationTypeToAViolation {
   all vT: ViolationType | one v: Violation | v.violationType=vT
}
```

```
//Each city, house code, street and position are associated to a place
fact eachCityHouseCodeStreetCityPositionToAPlace {
   all c: City | one p : Place | p.city=c
   all hC: HouseCode | one p : Place | p.houseCode=hC
   all s: Street | one p : Place | p.street=s
   all pos: Position | one p : Place | p.position=pos
}

//Each place, position and license plate in a report from an user were identified by a third part
fact eachPlaceAndPositionAndTrafficPlateInAReportWereIdentified {
   all p: Place | all vR: ReportFromUser | p=vR.place implies
      one s: Service | one mS: MapService | s.thirdPart=mS and s.serviceObject=p

   all p: Position | all vR: ReportFromUser | p=vR.place.position implies
      one s: Service | one mS: MapService | s.thirdPart=mS and s.serviceObject=p

   all tP: LicensePlate | all vR: ReportFromUser | tP=vR.vehicle.licensePlate implies
      one s: Service | one rPS: RecognitionPlateSystem | s.thirdPart=rPS and s.serviceObject=tP
}

//Each place and position can be verified by only the MapService
fact eachPlaceAndPositionIdentifiedByOnlyMapService {
   all p: Place | all t: ThirdPart | not( some s: Service | s.thirdPart=t and s.serviceObject=p and t not in MapService)

   all p: Position | all t: ThirdPart | not( some s: Service | s.thirdPart=t and s.serviceObject=p and t not in MapService)
}

//Each license plate can be verified by only the MapService
fact eachLicensePlateIdentifiedByOnlyRecognitionPlateSystem {
   all lp: LicensePlate | all t: ThirdPart | not( some s: Service | s.thirdPart=t and s.serviceObject=lp and t not in Recognitio
            nPlateSystem)
}

//Each Password is associated to a registration
fact eachPasswordToARegistration {
   all p: Password | one r: Registration | r.password=p
}

//Each statistic and suggestion were made by SafeStreets
fact eachStatisticAndSuggestionToSafeStreets {
   all stat: Statistic | one safeS: SafeStreets | stat in safeS.statistic
   all sugg: Suggestion | one safeS: SafeStreets | sugg in safeS.suggestion
}

//Each traffic ticket refers to a violation made in the city of the Municipality that has done the traffic ticket
fact trafficTicketsOnlyInTheCityOfTheMunicipality {
   all m: Municipality | all tt: TrafficTicket | tt in m.trafficTicket implies tt.refersToReport.place.city=m.place.city
}

//Each suggestion is for a Municipality that resides on the same city of the place of the suggestion
fact suggestionForMunicipalityWithSameCity {
```

```
    all s: Suggestion | s.place.city=s.forMunicipality.place.city
}

//All reports from an user refers to a parking violation
fact reportFromUserRefersToAParkingViolation {
    all r: ReportFromUser | one pv: ParkingViolationType | r.violation.violationType=pv
}

//The statistics about the vehicles are not for the users
fact vehicleStatisticNotForUsers {
    all vS: VehicleStatistic | vS.forUser=none
}

//Each traffic ticket refers to a report present in the Municipality that has done the traffic ticket
fact trafficTicketsReferToReportInTheMunicipality {
    all tt: TrafficTicket | tt.refersToReport in ReportFromMunicipality or (one m: Municipality | tt in m.trafficTicket
        and tt.refersToReport in m.takenReportFromUser)
}

//Each place is associated to a Municipality or a report
fact eachPlaceInMunicipalityOrReport {
    all p: Place | (one m: Municipality | m.place=p) or (one r: Report | r.place=p)
}

//For a statistic there must be at least one report from an user
//and for a suggestiont there must be at least one report from an user in the same place of the suggestion
fact forEachStatisticAndSuggestionThereMustBeAtLeastOneReportFromUser {
    all s: Statistic | some r: ReportFromUser | one safeS: SafeStreets | r in safeS.reportedViolation and s in
        safeS.statistic
    all s: Suggestion | some r: ReportFromUser | one safeS: SafeStreets | r in safeS.reportedViolation and
        s.place in r.place
}


//R7
//The Municipality can take the reports from SafeStreets only if its competence area (its city)
fact theMunicipalityCanTakeReportedViolationOnlyOfItsCompetenceArea {
    all m: Municipality | all r: ReportFromUser | r in m.takenReportFromUser implies r.place.city=m.place.city
}


//R9
//Accept valid reports
//In SafeStreets there are all the reports from the users
fact acceptValidReports {
    all r: ReportFromUser | one safeS: SafeStreets | r in safeS.reportedViolation
}


//R14
//Each user was identified by an Identity Verifier
fact eachUserWasIdentified {
    all u: User | one s: Service | one idV: IdentityVerifier | s.serviceObject=u and
```

```
        s.thirdPart=idV
}


//R14
//Every user is registered
fact everyUserIsRegistered {
    all u: User | one r: UserRegistration | r.user=u
}


//R15
//Every Municipality is registered
fact everyMunicipalityIsRegistered {
    all m: Municipality | one r: MunicipalityRegistration | r.municipality=m
}


--end of facts--



--Goals:--

//G1
//The System accepts valid reports by the users about the parking violations
//There is at least one report from an user
pred acceptCompleteReports {
    #ReportFromUser>0
}

run acceptCompleteReports for 4

//G2
//The System suggests possible interventions to the Municipality
//There is at least one suggestion from SafeStreets
pred suggestInterventions {
    #Suggestion>0
}

run suggestInterventions for 4

//G3
//The System allows the Municipality to retrieve submitted parking violations of its competence area
//After taken the violation, the Municipality has all and only the reported violations of its city
assert theMunicipalityCanRetrieveSubmittedViolations {
    all m: Municipality | theMunicipalityTakesTheReportedViolations[m, Violation] implies
        (all r: ReportFromUser | r in m.takenReportFromUser implies
                r.place.city=m.place.city else r.place.city!=m.place.city)
}

check theMunicipalityCanRetrieveSubmittedViolations
```

```
//G4
//The System gives some statistics to the User about the violations
//There is at least statistic from SafeStreets for a user
pred giveStatisticsToTheUser {
   #Statistic>0
   some s: Statistic | s.forUser!=none
}

run giveStatisticsToTheUser for 4

//G5
//The System can give all the statistics to the Municipality about the violations
//There is at least statistic from SafeStreets for a Municipality
pred giveStatisticsToTheMunicipality {
   #Statistic>0
   some s: Statistic | s.forMunicipality!=none
}

run giveStatisticsToTheMunicipality for 4

//G6
//The System can retrieve the violations verified by the Municipality
assert safeStreetsCanRetrieveViolationsFromTheMunicipality {
   all safeS: SafeStreets | all m: Municipality | safeStreetsRetrievesViolationsFromTheMunicipality[m, safeS] implies
      (all tt: TrafficTicket | tt in m.trafficTicket implies (tt.refersToReport in safeS.takenReportFromMunicipality or
         tt.refersToReport in safeS.reportedViolation))
}

check safeStreetsCanRetrieveViolationsFromTheMunicipality

//G1 and G2 and G4 and G5
//G1
//The System accepts valid reports by the users about the parking violations
//There is at least one report from an user
//G2
//The System suggests possible interventions to the Municipality
//There is at least one suggestion from SafeStreets
//G4
//The System gives some statistics to the User about the violations
//There is at least statistic from SafeStreets for an user
//G5
//The System can give all the statistics to the Municipality about the violations
//There is at least statistic from SafeStreets for a Municipality
pred goals1and2and4and5 {
   acceptCompleteReports
   suggestInterventions
   giveStatisticsToTheUser
   giveStatisticsToTheMunicipality
}

run goals1and2and4and5 for 4
```

```
--end of goals--


--Operations:--
//The Municipality takes the reports from the users
pred theMunicipalityTakesTheReportedViolations[m: Municipality, vs: set Violation] {
   m.takenReportFromUser=m.takenReportFromUser+vs
}

//SafeStreets retrives the reports from a Municipality
pred safeStreetsRetrievesViolationsFromTheMunicipality[m: Municipality, safeS: SafeStreets] {
       safeS.takenReportFromMunicipality=safeS.takenReportFromMunicipality+m.trafficTicket.refersToReport
}

--end of the operations--


//Worlds:
//Without Municipalities
pred noMunicipality {
   #Municipality=0
   #ReportFromUser=1
}

run noMunicipality for 4

//With only one Municipality
pred oneMunicipality {
   #Municipality=1
}

run oneMunicipality

//With two Municipalities
pred moreMunicipalities {
   #Municipality=2
   #User=2
   #ReportFromUser=2
   #TrafficTicket=2
}

run moreMunicipalities for 12
```

5 EFFORT SPENT

- Abbo Giulio Antonio

DESCRIPTION OF THE TASK	HOURS
Purpose, scope, definitions	5,5
Product perspective	9
Product functions	5
Domain assumptions	3,5
External interface requirements	9
Functional requirements	7
Non-functional Requirements	2
Formal analysis using Alloy	2

- Accordi Gianmarco

DESCRIPTION OF THE TASK	HOURS
Purpose, scope, definitions	3,5
Product perspective	2
Product functions	3,5
Domain assumptions	2,5
External interface requirements	2
Functional requirements	10,5
Non-functional Requirements	1,5
Formal analysis using Alloy	5

- Bonetti Massimiliano

DESCRIPTION OF THE TASK	HOURS
Purpose, scope, definitions	2,5
Product perspective	4
Product functions	4
Domain assumptions	4
External interface requirements	1
Functional requirements	7,5
Non-functional Requirements	1
Formal analysis using Alloy	18

6 REFERENCES

- Slides of the Software Engineering 2 course by prof. Di Nitto at Politecnico di Milano:
  - "2. Alloy"
  - "3.a Requirements Engineering"
  - "3.c World and Machine"
  - "4.a Elicitation of Requirements"
  - "4.d UML for Requirements Engineering"
  - "5.b Structure of RASD"
- NUSEIBEH, Bashar; EASTERBROOK, Steve. Requirements engineering: a roadmap. In: *Proceedings of the Conference on the Future of Software Engineering*. ACM, 2000. p. 35-46.
- ZAVE, Pamela; JACKSON, Michael. Four dark corners of requirements engineering. *ACM transactions on Software Engineering and Methodology (TOSEM)*, 1997, 6.1: 1-30.
- JACKSON, Michael. The world and the machine. In: *1995 17th International Conference on Software Engineering*. IEEE, 1995. p. 283-283.
- GOOGLE. Material Design Guidelines. [Accessed 8 November 2019]. Available from: https://material.io/