



**POLITECNICO**  
MILANO 1863



**SafeStreets**

# **REQUIREMENTS ANALYSIS AND SPECIFICATION DOCUMENT**

Version 1 – 09/11/2019

---

*Authors:*

Giulio A. Abbo 10538950  
Gianmarco Accordi 10587213  
Massimiliano Bonetti 10560496

*Professor:*

Elisabetta Di Nitto  
*Academic year:*  
2019 – 2020

## 0 CONTENTS

---

0	Contents.....	2
1	Introduction .....	4
1.A	Purpose.....	4
1.A.1	Problem overview.....	4
1.A.2	Goals.....	4
1.B	Scope.....	4
1.C	Definitions, acronyms, abbreviations.....	6
1.C.1	Definitions .....	6
1.C.2	Acronyms.....	7
1.C.3	Abbreviations .....	7
1.D	Revision history.....	7
1.E	Reference Documents.....	7
1.F	Document structure.....	7
2	Overall Description.....	9
2.A	Product perspective .....	9
2.B	Product functions.....	11
2.C	User characteristics .....	12
2.C.1	The user.....	12
2.C.2	The municipality .....	13
2.D	Assumptions, dependencies and constraints.....	13
3	Specific requirements .....	14
3.A	External interface requirements.....	14
3.A.1	User interfaces.....	14
3.A.2	Hardware interfaces.....	16
3.A.3	Software interfaces.....	16
3.A.4	Communication interfaces.....	16
3.B	Functional requirements .....	16
3.B.1	Use Case Diagram .....	16
3.B.2	Scenarios .....	17
3.B.3	Use case analysis .....	18
3.B.4	Sequence Diagrams .....	21
3.B.5	Traceability Matrix.....	28
3.B.6	Mapping .....	28
3.C	Performance requirements.....	30
3.D	Design constraints.....	30

3.D.1	Standard compliance .....	30
3.D.2	Hardware limitations .....	30
3.D.3	Any other constraint.....	30
3.E	Software system attributes .....	30
3.E.1	Reliability.....	30
3.E.2	Availability .....	30
3.E.3	Security .....	31
3.E.4	Maintainability.....	31
3.E.5	Portability.....	31
4	Formal analysis using ALLOY .....	32
4.A	Generated worlds.....	32
4.B	Results.....	36
4.C	Alloy code .....	38
5	Effort spent .....	47
6	References.....	48

# 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*<sup>1</sup>.

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*<sup>2</sup>.

### 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

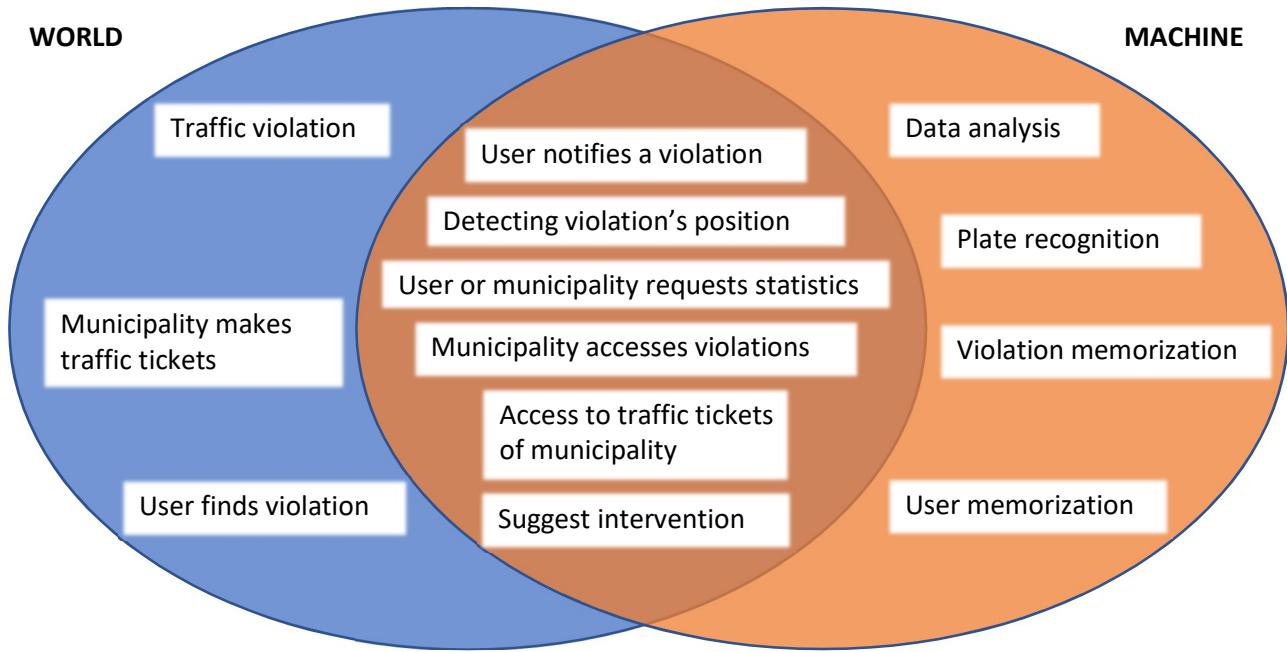
Here it is presented the initial distinction between events that belong to the world and those that belong to the machine (the system). The violations and the event *user finds a violation* are considered part of the world. Indeed, the machine is not aware of these events unless the user sends a report 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

---

<sup>1</sup> This is the *Advanced Function 1* of the *Project Assignment*.

<sup>2</sup> 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.



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 viability 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. For this reason, 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 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 from a user 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

## 1.D REVISION HISTORY

- Version 1.0 First version of the document.

## 1.E REFERENCE DOCUMENTS

- Specification document: “Mandatory Project Assignment AY 2019-2020”
- Standard ISO/IEC/IEEE 29148

## 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<sup>3</sup>, with some 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

---

<sup>3</sup> This distinction is the one proposed by M. Jackson in his paper “The World and the Machine”.

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.

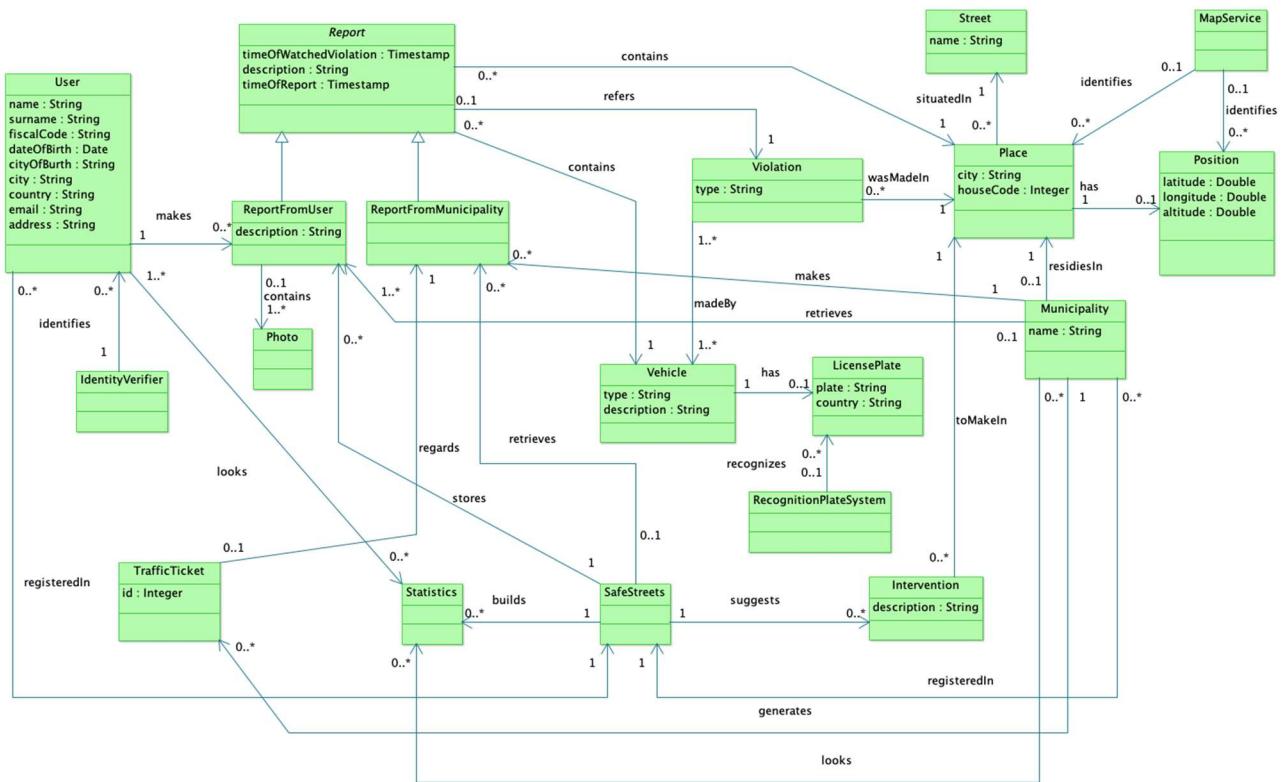
## 2 OVERALL DESCRIPTION

### 2.A PRODUCT PERSPECTIVE

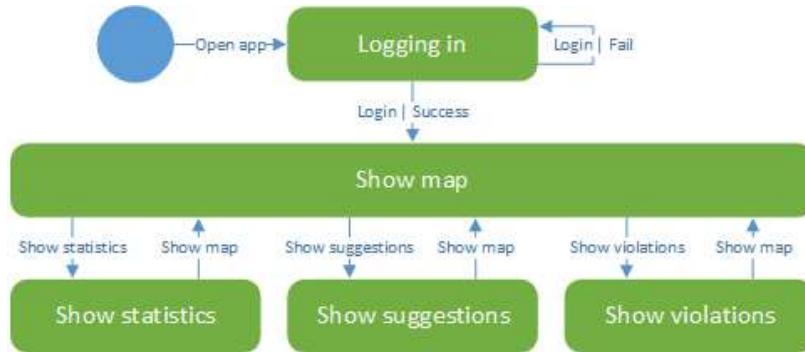
The system is divided into three products: a software for the user (mainly used on mobile devices), a software for the municipality and one for the system's backend.

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 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.

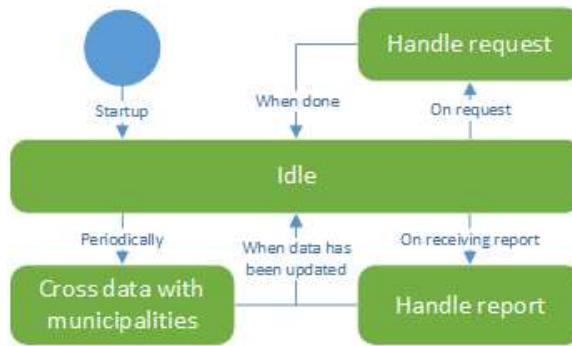
The next picture contains the class diagram representing the conceptual model of the application domain.



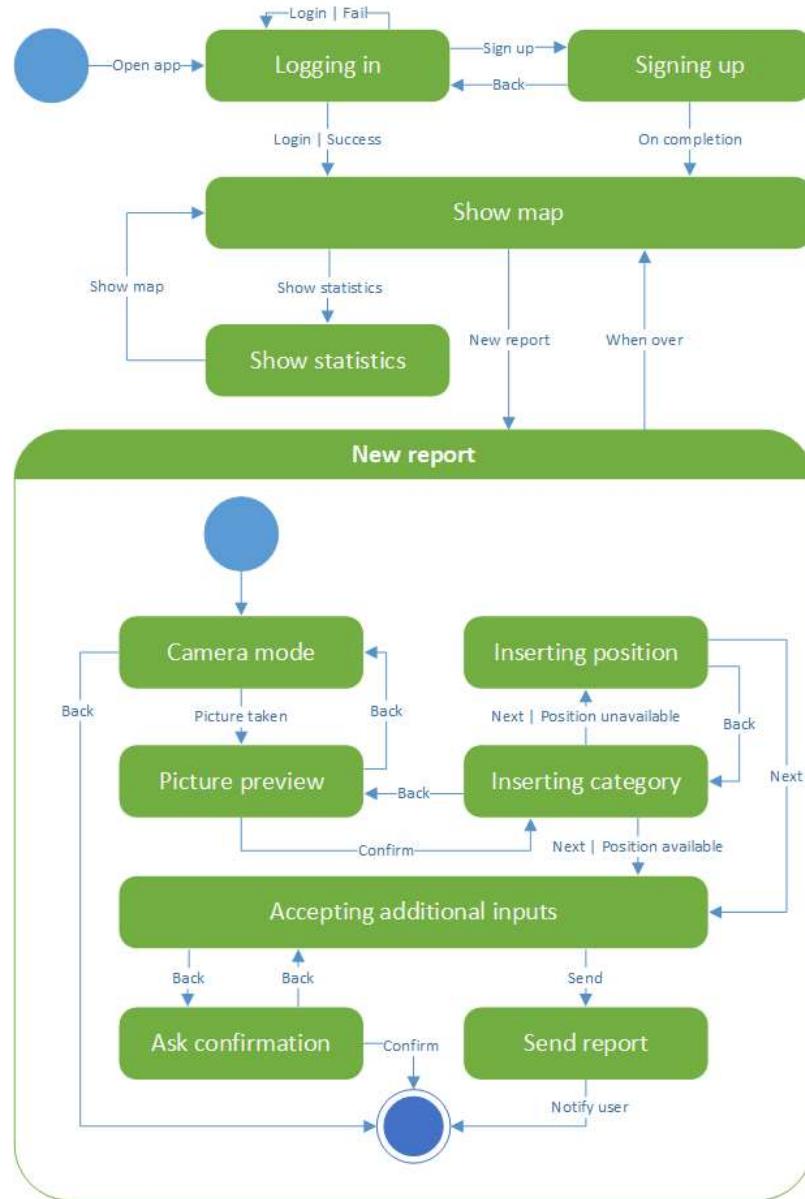
Here follows a state chart for the municipality app. After the login, the municipality is showed a map of its competence area with an overlay representing the streets with the most violations. From this state it can access other statistics, the suggestion provided by the system, and a list of the violations.



The next image represents the states of the back-end software. This software should be always running and in a multithreaded fashion; it awaits input from the municipality or the user, periodically it crosses data with the municipality.



The following picture 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.



## 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 except for 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 to take a picture or select one from the device.
- 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 Document 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. The Accidents are:

- 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 viability 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:

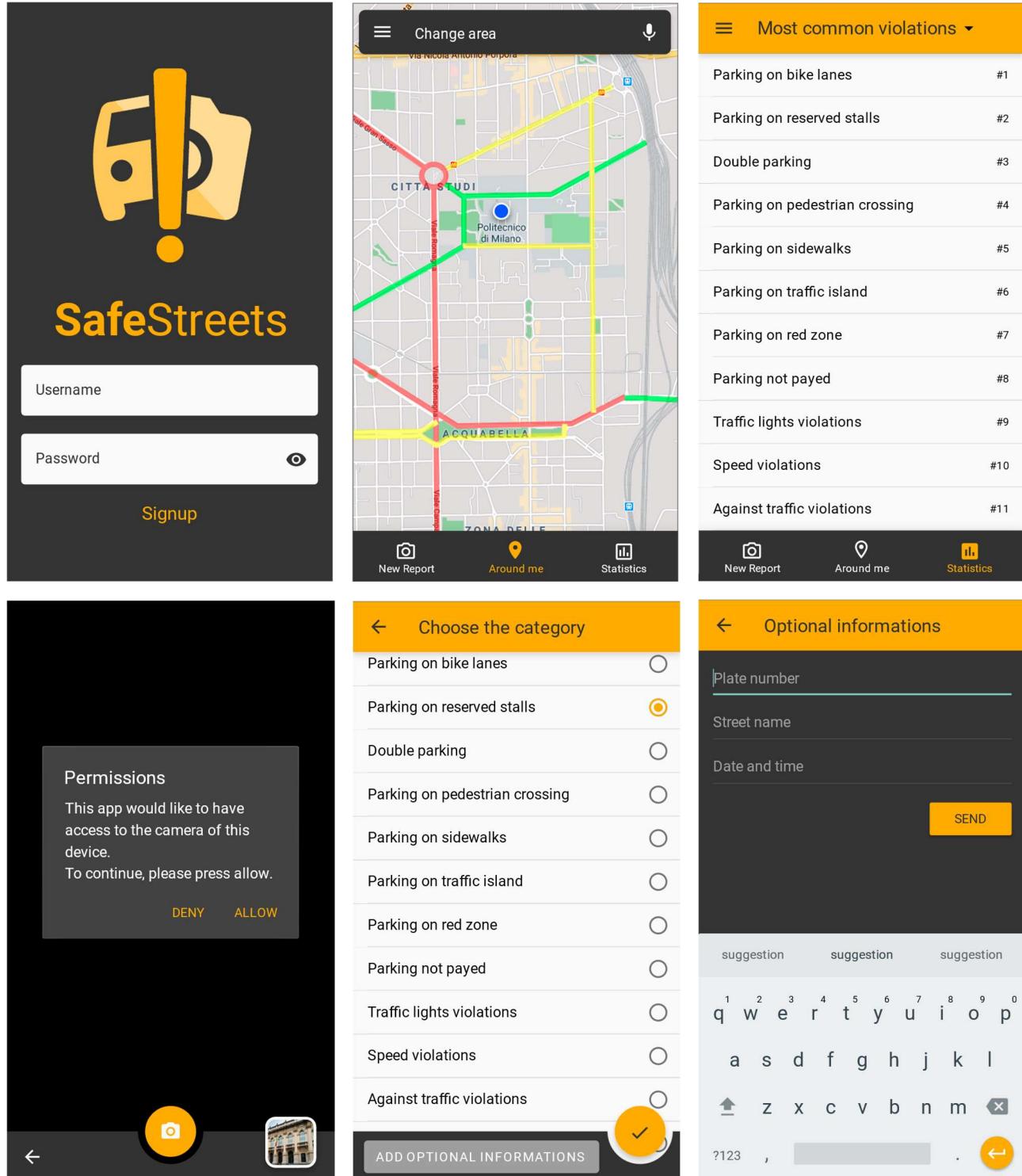
- D1: If the license plate recognizer recognises the license plate, then the result is correct.
- D2: If the map service recognises the street name from the coordinates, then the result is correct.
- D3: The identity card is correctly verified.
- D4: The Municipality possesses only real violations.
- D5: The data retrieved by the smartphone's GPS are correct.
- D6: The time reported by the user's smartphone is correct.

## 3 SPECIFIC REQUIREMENTS

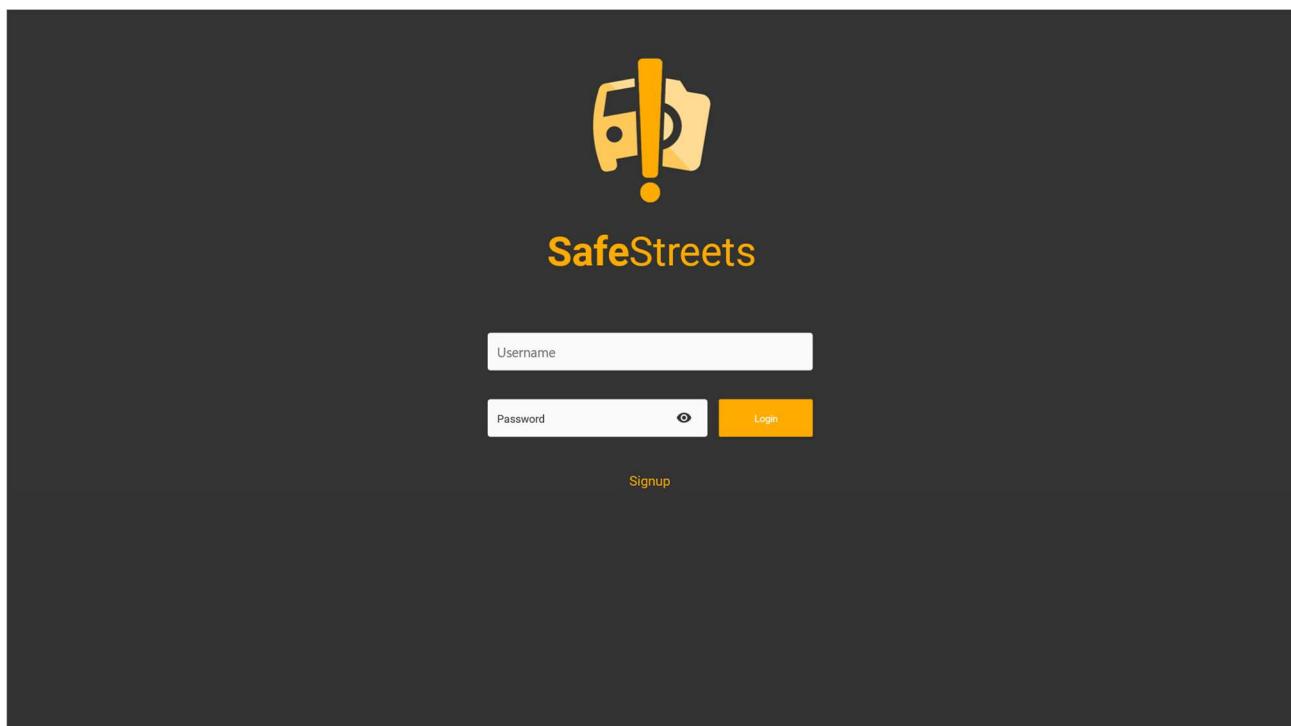
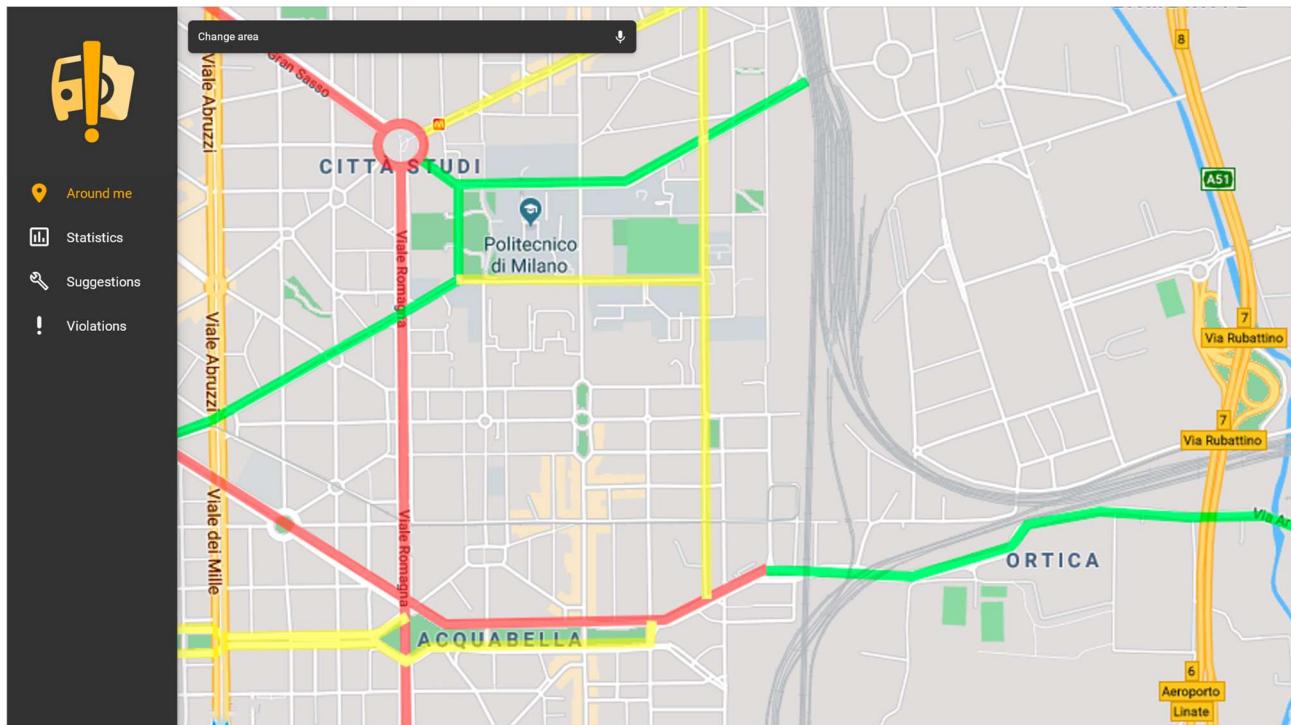
### 3.A EXTERNAL INTERFACE REQUIREMENTS

#### 3.A.1 User interfaces

The software will provide a different UI for the user and the municipality, providing different functionalities. Here follow some prototype mock-ups of what the user UI might look like: from the top it is included a login screen, show map, show statistics and three screens of the new report process.



When the user opens the app the login menu is displayed, here the user can complete the login operation or the registration, and when the user is correctly logged in then the system display the homepage, where the user can see the area around him, its position if the GPS is available, and also some statistics about the most important road around him. The footer menu allows the User to send a new report or to ask for some statistics. In the statistics menu the user can choose the area of interest and from a menu he can choose which type of statistic he requires. While in the new report menu the user will compile the report of the violation.



### 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.

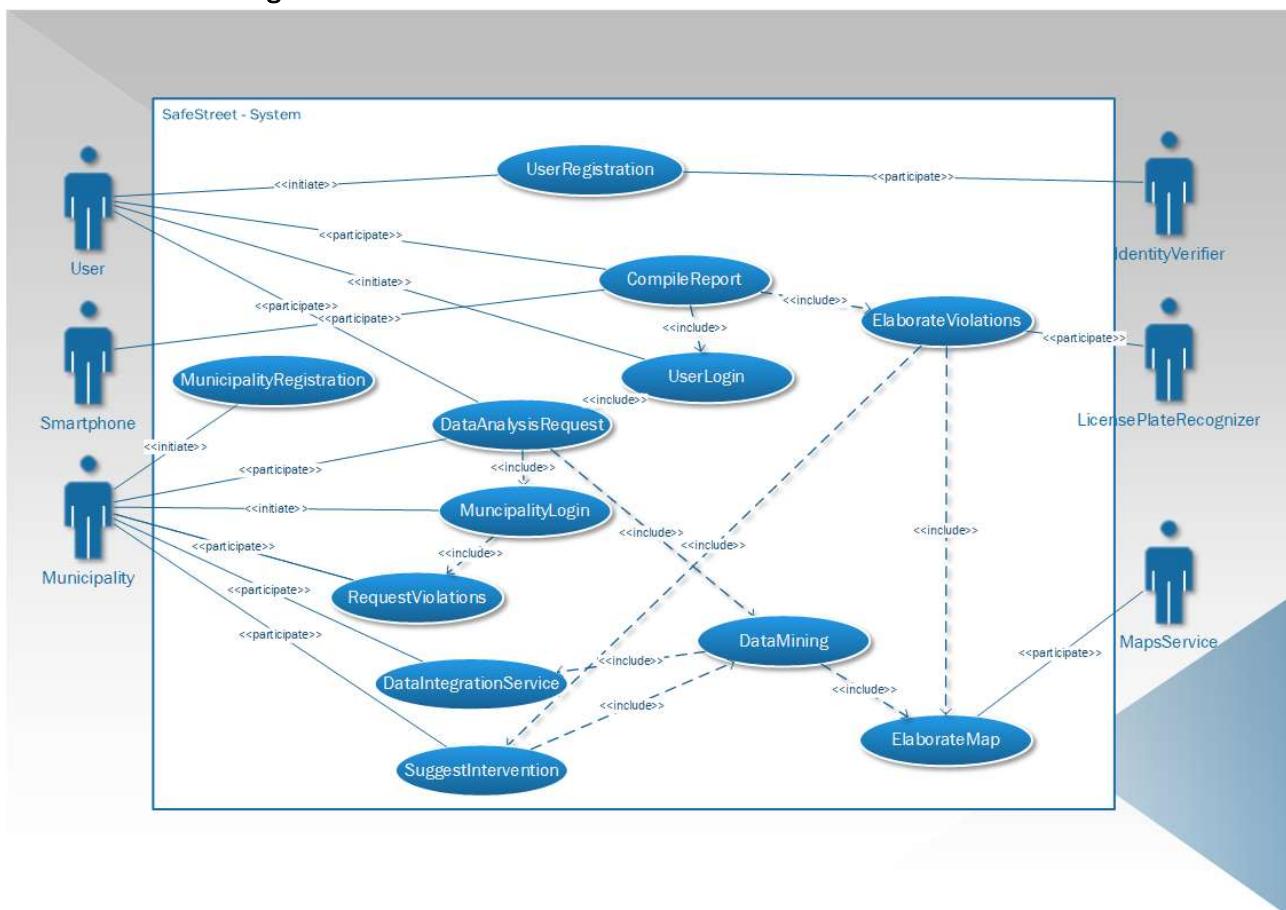
The system will also need interfaces to handle the communication with the municipalities for the data integration, suggestions, and access reports services.

### 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



In all the following Use Cases it's implicit that in the Exception can also happen 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.

The smartphone is considered as an actor because it allows to retrieve GPS coordinates and to take a picture for the camera. The smartphone is also used to retrieve the timestamp of the notification, if the user doesn't provide the timestamp inside the required field during the compilation of the report.

### 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 the SafeStreets' App on his smartphone and he registers himself by providing the copy of his Identity Card and his generality. After the System has authenticated the User and it has verified Bob's identity, Bob starts to fill the report of the violations (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 his effectiveness in discovering new violation. The Monza's Municipality also discover that the number of people that are using the SafeStreets' App in the province is growing very fast. So, Monza's Municipality decides to use the services of SafeStreets, so he contacts the SafeStreets' Organization to elaborate a contract that gives to the Municipality a Contract Code. That Code must be prompted during the registration of the Municipality at the registration portal. The Monza's Municipality sends the code to the System that verifies it and then ask to the Municipality the username and the password that will be used for the future login operations. The Municipality specifies his preference and it can access the violations reported to the SafeStreets' System that are under its authority area.

#### **UserDataAdapter**

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

#### **MunicipalityTicket**

The Monza's Municipality, after the registration, wants to access the violations, about his authority area, that has been reported to the SafeStreets System. In order to do so, the Municipality signs in the System and after he requests for a pull of the requested violations, the System then will check the request of the Municipality and he will send only the violations that are under the authority of the Municipality. The Municipality then retrieves the requested violations and he can check them and eventually it will emit a ticket. 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 offers this service.

#### **RetrieveMunicipalityViolations**

The Monza's Municipality offers also the possibility of to retrieve the information about the violations that occurs on its territory and that has been registered by them. The SafeStreets uses the credentials

provided by the Municipality to authenticate for the service. Then SafeStreets 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 infraction that need 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 position, the System store 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 Users sends the request of Registration
2. The Users starts to fill the registration forms
  - 2.1. A picture of the User is registered
  - 2.2. The User inserts his generality
  - 2.3. The User provides also his email, username and password he will use
  - 2.4. The User inserts the copy of his/her 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 generality
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 Verifies 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 generality, 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 generality and tries to sign in
2. The System receives this request and search the requested account
3. The System verify 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 has correctly sign in in his account

**Exceptions:** If the User provide an email that doesn't appears inside the User List, if the provided password is wrong, also if the User is already logged in, in all of this cases the User will receives an error messages and he has to retry the login with a different email or password.

#### **U3: Report Compilation**

**Actors:** User, Smartphone

**Entry conditions:** The User wants to notify a Violation

**Flow of events:**

1. The User sign in the System
2. The User starts to fill a form for the Violations notification
3. The System asks the User for a picture of the violation with the license plate of the vehicle
  - 3.1. The User can now select to select a picture from the memory of his smartphone, or the User can take directly a photo from his camera
4. The User can decide to select a picture from the gallery or make a shoot from 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 ask the User to select the violation type
  - 5.4. The User inserts the violation type
  - 5.5. Then the System ask the user to fill some options that are not mandatory
  - 5.6. The User responds with the non-mandatory options
6. The System then 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 in the system, so he will receives an error message and he has to retry the login.

#### **U4: Analysis Request**

**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 sign in
2. The User or the Municipality ask 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 than retrieves the information from his knowledge and asks also the Municipality for their knowledge about the violations, if the Municipality of interests offers the service of retrieve the violations
5. The System from the retrieved data mines the information
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 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 his knowledge of the violations or the knowledge of the Municipality, the system cannot interpreted the data that it has retrieved (for example Maps Service cannot understand the position), the User or the Municipality has the wrong right access for the requested data analysis, in all of this case the operation is aborted by the System and the User or the Municipality need to redo the request.

#### **U5: Violations Request**

**Actors:** Municipality, Maps Service.

**Entry conditions:** The Municipality wants to retrieve the violations notified to the System.

**Flow of events:**

1. The Municipality sign 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 elaborate the request and retrieve all the latest violations concerning the Municipality, and he verifies the right access to the violations
4. The System sends the latest violations to the Municipality
5. The Municipality access the update sent by the System

**Exit conditions:** The Municipality get 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 request a registration operation
2. The System asks the Municipality to insert his 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 operation
5. The Municipality sends the required credentials
6. The System memorize the credentials choose by the Municipality
7. The System ask the Municipality if they offer a service for retrieve the violations store by them
  - 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:** The Municipality provides a wrong Contract Code, the Municipality provides wrong credentials to access their service (if the provides the service of access their stored violations), in all of this case the Municipality receives an error messages, and the operation needs to be redo.

**Before the registration:** The Municipality contacts an employee of SafeStreets in order to get a contract code for the registration, that is provided when the two parts sign a contract. This operation is not considered managed by the system, so it is not included in the use case and sequence diagram.

## U7: Report Elaboration

**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 exact coordinate of the violation
5. The Maps Service sends back the coordinate to the System
6. The System then stores the violations

**Exit conditions:** The violation has been correctly stored.

**Exceptions:** The License Plate Recognizer or the Maps Service are unavailable, this means that the System will try to contact them in future to validate the Report.

## U8: Municipality Suggestions

**Actors:** Municipality

**Entry conditions:** The System notify the Municipality about possible interventions.

**Flow of events:**

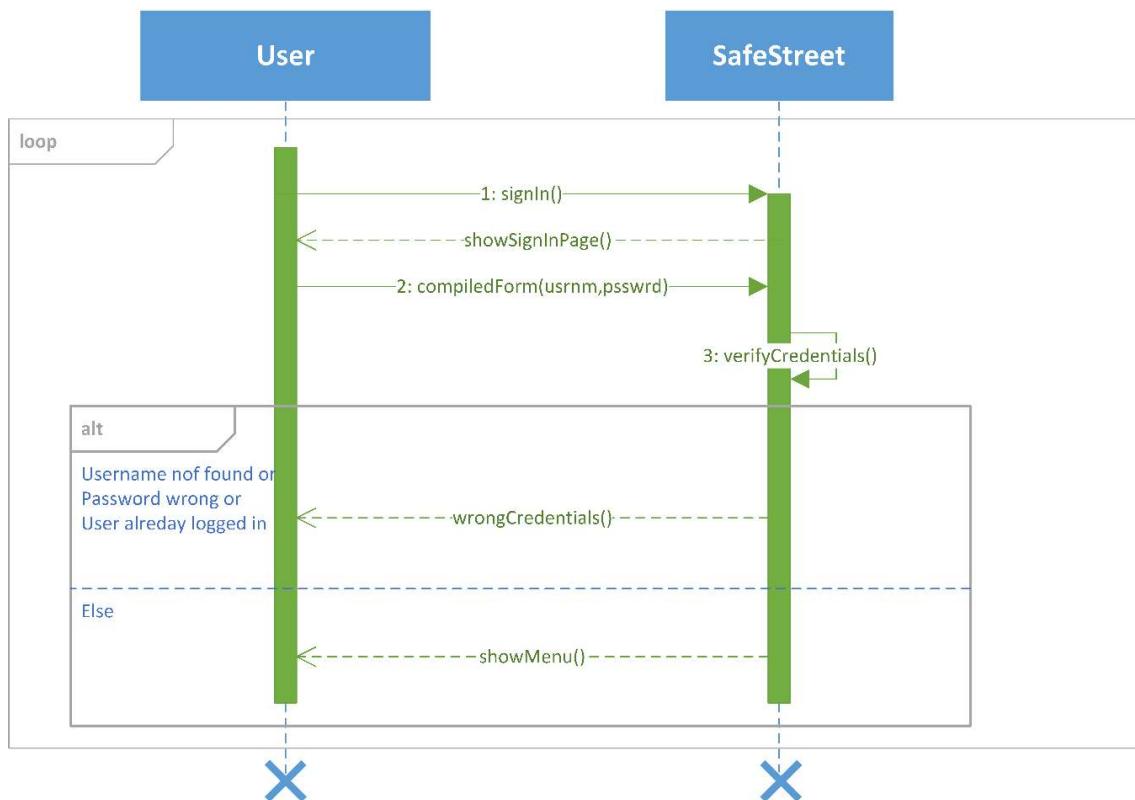
1. The Municipality logs into the system
2. The Municipality wants to get some suggestions
3. The system then calculates if some suggestions are available
  - 3.1. The system verifies the competence area of the Municipality
  - 3.2. The system then finds the more common violations in a certain area
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

**Exit conditions:** The Municipality doesn't want other suggestions, or there aren't other suggestions.

**Exceptions:** The Municipality can not sign in.

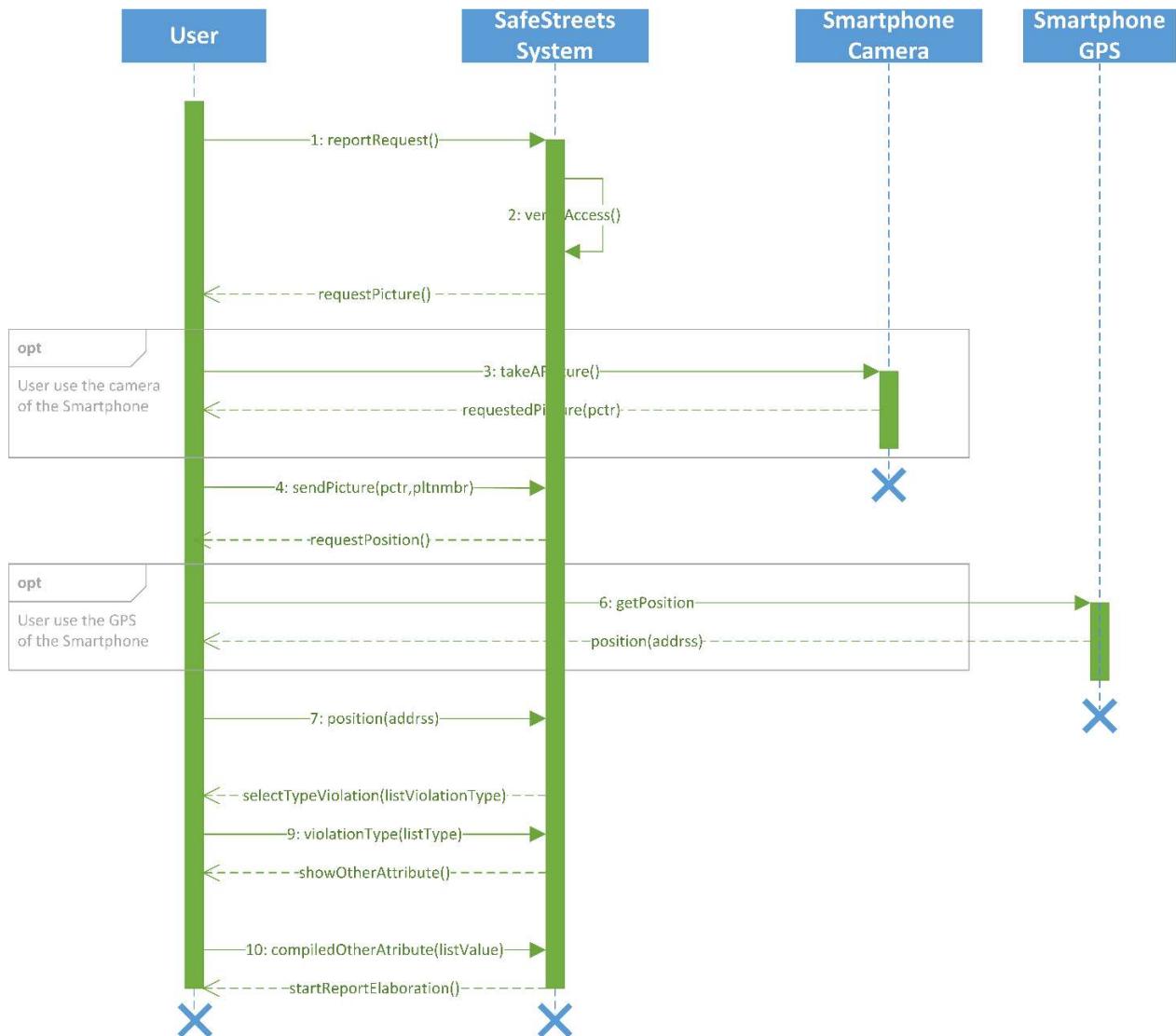
### 3.B.4 Sequence Diagrams

#### Login



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

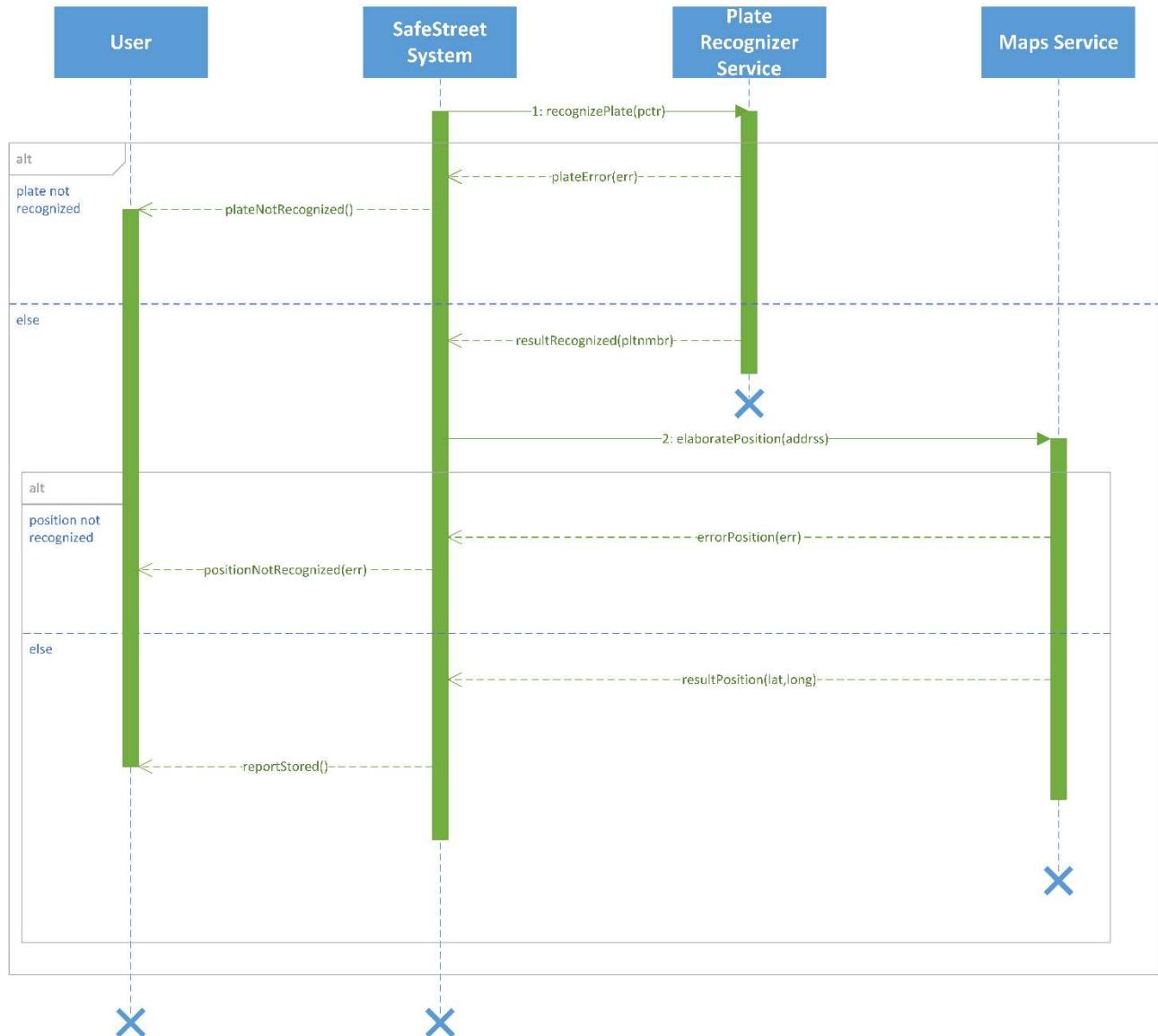
## Report



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 it 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.

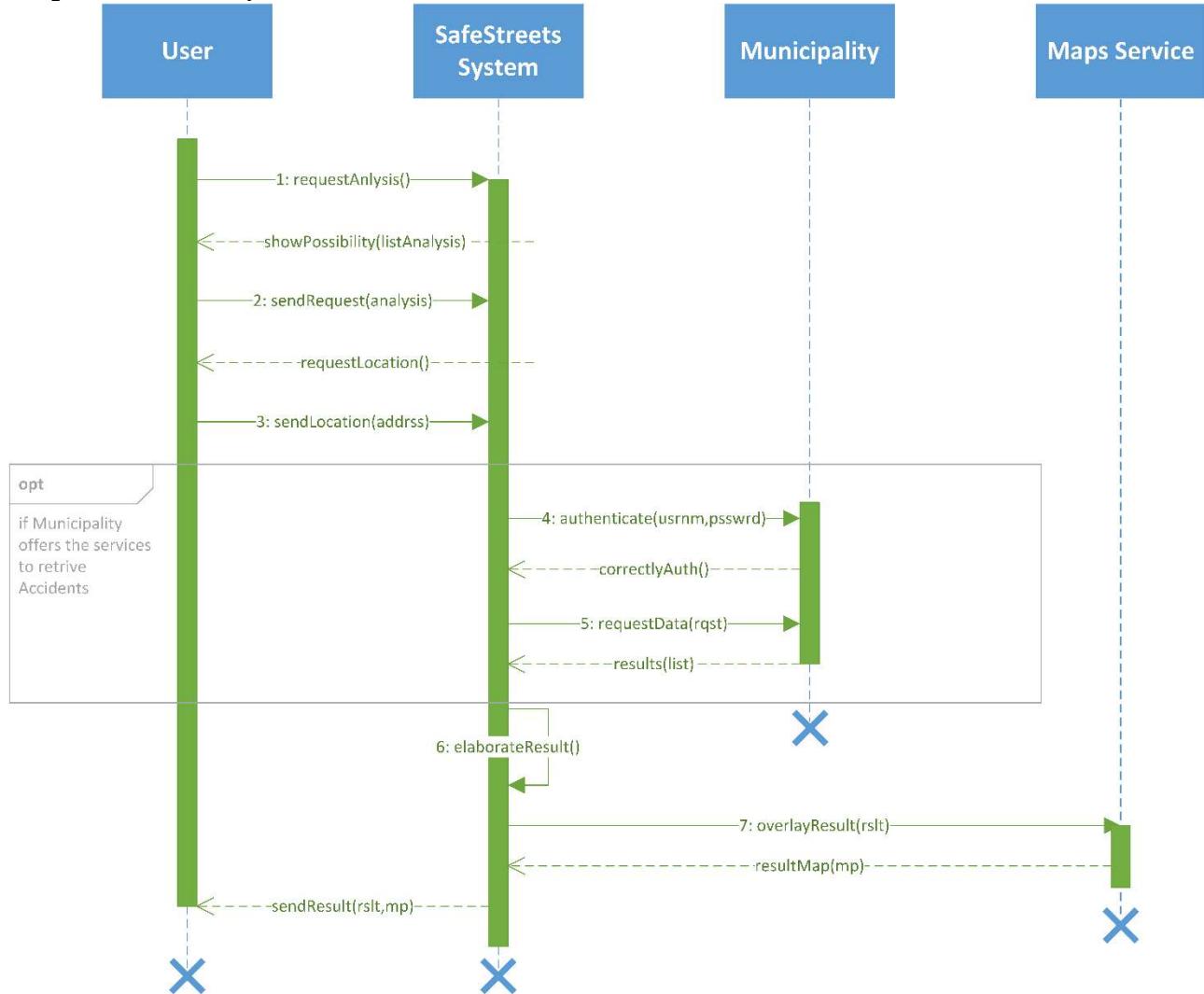
Also the last respond message indicates that the SafeStreets' System will take care of the elaboration of the report that is presented in the sequence diagram *Elaboration* below.

## Elaboration



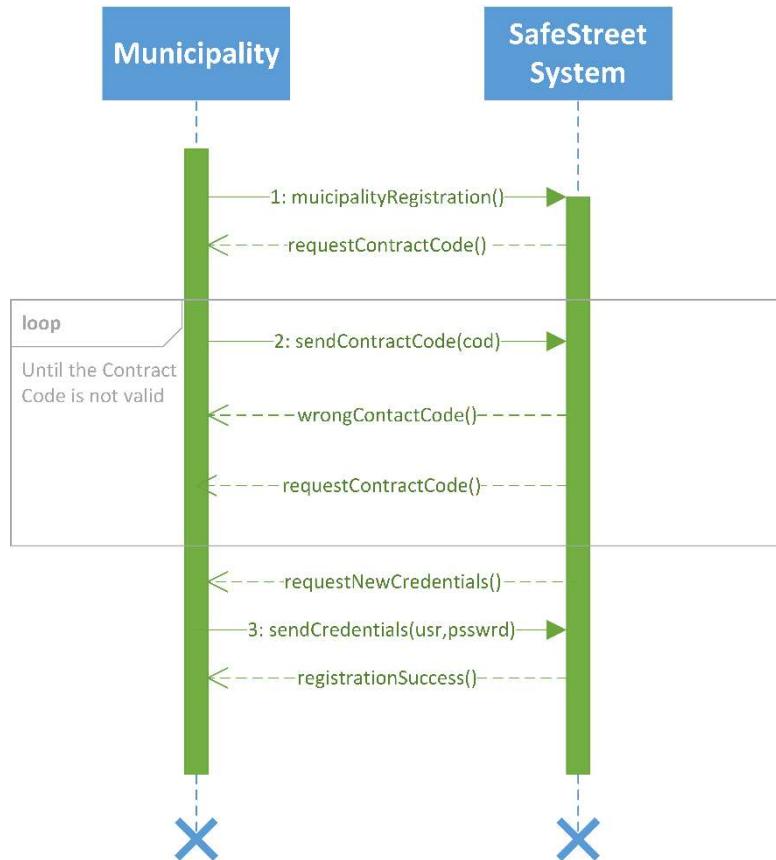
This Sequence Diagram represents the sequence of operations that will be carried out by the SafeStreets' System to elaborate the report delivered by the user. This sequence of operations happens after the User has correctly submitted the report, like in the *Report* sequence diagram presented above. The SafeStreets' System use 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 precise position of the violations.

### Request Data Analysis



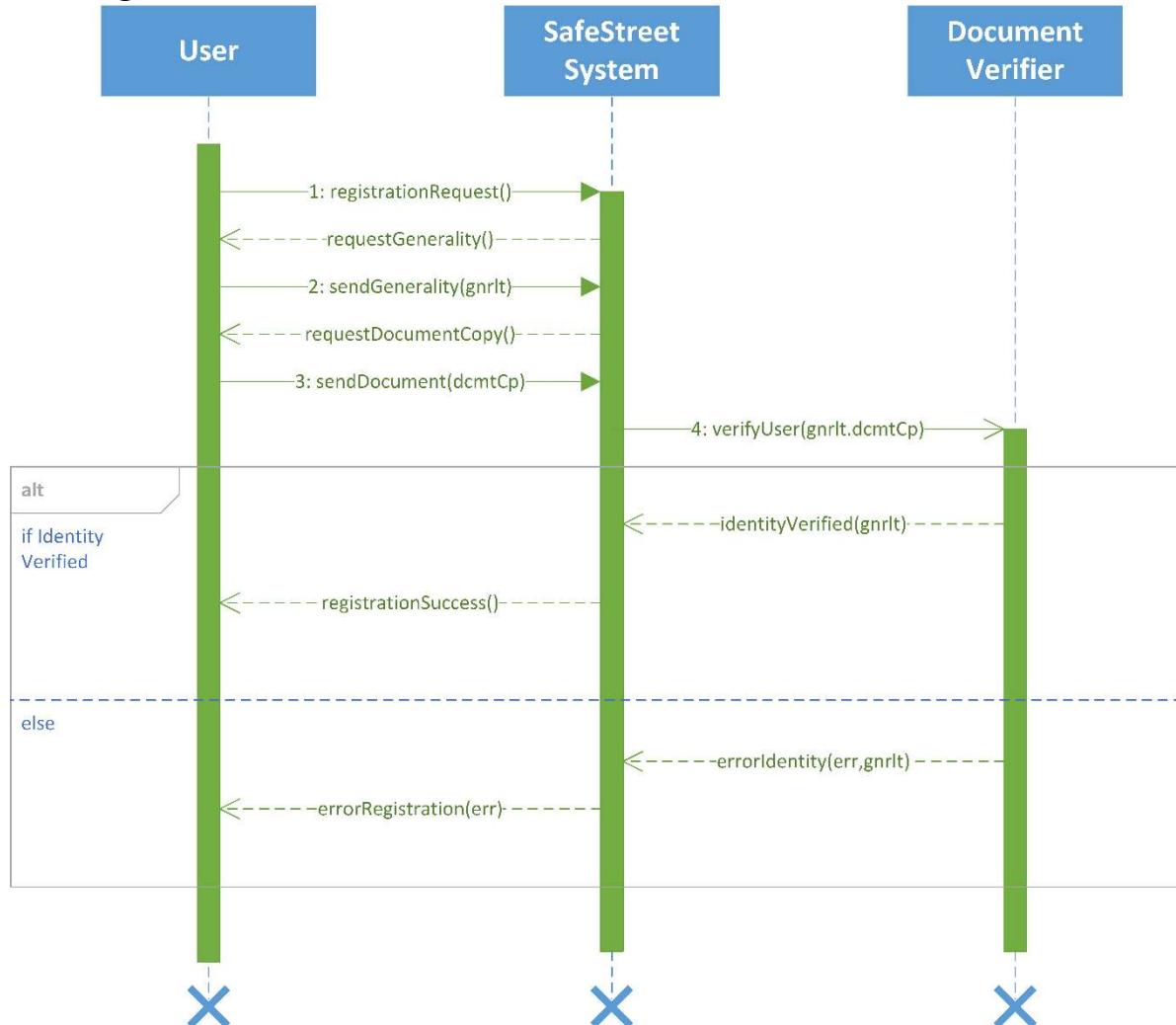
This sequence diagram represents the sequence operations done to perform a data analysis request made by the User. As before this operation can be carried out if the user has been correctly *logged* in the system. This sequence of operations performed by the User, can be carried out also from the Municipality.

## Municipality Registration



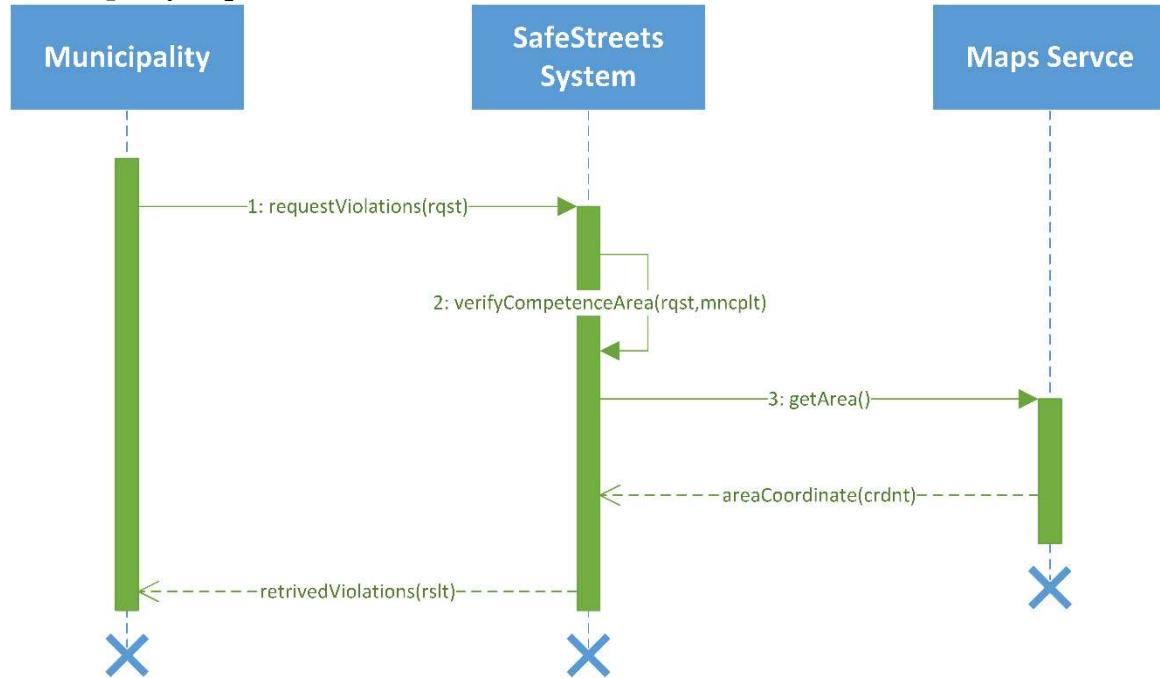
This sequence diagram represents the sequence of operations done to serve the Municipality Registration. After this the Municipality can correctly use the SafeStreets' services. The Municipality needs to insert the contract code that has been previously obtained by SafeStreets, after a contract has been signed.

## User Registration



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 the user can use all the services exposed by the SafeStreets' System.

### 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 need to find out what request can be displayed to the Municipality, using the services exposed by the Maps Service, in order to find out what violations belong inside the Municipality's competence area.

### 3.B.5 Traceability Matrix

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

REQUIREMENTS	USE CASE
R1	U4: Analysis Request U5: Violations Request U7: Report Elaboration
R2	U4: Analysis Request
R3	U4: Analysis Request U5: Violations Request
R4	U5: Violations Request
R5	U4: Analysis Request U5: Violations Request U7: Report Elaboration
R6	U3: Report Compilation
R7	U4: Analysis Request
R8	U3: Report Compilation
R9	U3: Report Compilation U7: Report Elaboration
R10	U4: Analysis Request
R11	U4: Analysis Request
R12	U8: Municipality Suggestions
R13	U7: Report Elaboration
R14	U1: User Registration U2: Login
R15	U6: Municipality Registration U2: Login
R16	U3: Report Compilation
R17	U1: User Registration
R18	U7: Report Elaboration
R19	U4: Analysis Request U5: Violations Request U7: Report Elaboration U8: Municipality Suggestions

### 3.B.6 Mapping

In this section we indicate for each goal the requirements and the domain assumption to ensure the correlated goal:

- **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 to take a picture or select one from the device.
  - R9: The system accepts valid 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 Document 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 are correct.
  - D6: The time reported by the User's smartphone is correct.
- **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.
- **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 violations.
  - R15: The System must allow the municipality to perform the registration and the login.
  - R19: The System must communicate with the Maps Service.
- **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 except for the vehicles that have committed the highest number of violations.
  - 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.
- **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.
- **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.

The messages will be sent using the RDF standard.

Reverse Geocoding output will be handled as follows: complete street name, house number, postal code, city, province, country.

#### 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, possibly GPS functionalities.

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 be reliable at all times; 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 splitting 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, provided the minimum hardware requirements, will be built entirely on an underlying abstraction layer (a virtual machine, like Java, or an engine). This should ensure a sufficient level of portability for the system, which will be for example accessible via app or web.

## 4 FORMAL ANALYSIS USING ALLOY

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

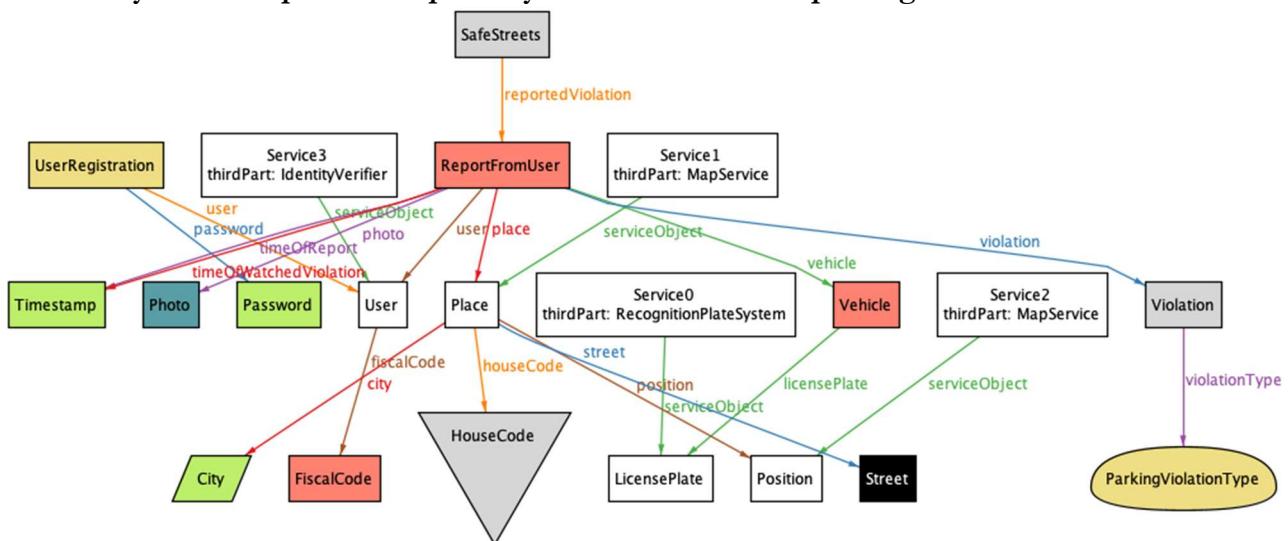
The entities have been represented through signatures. Some requirements have been represented with facts.

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.**

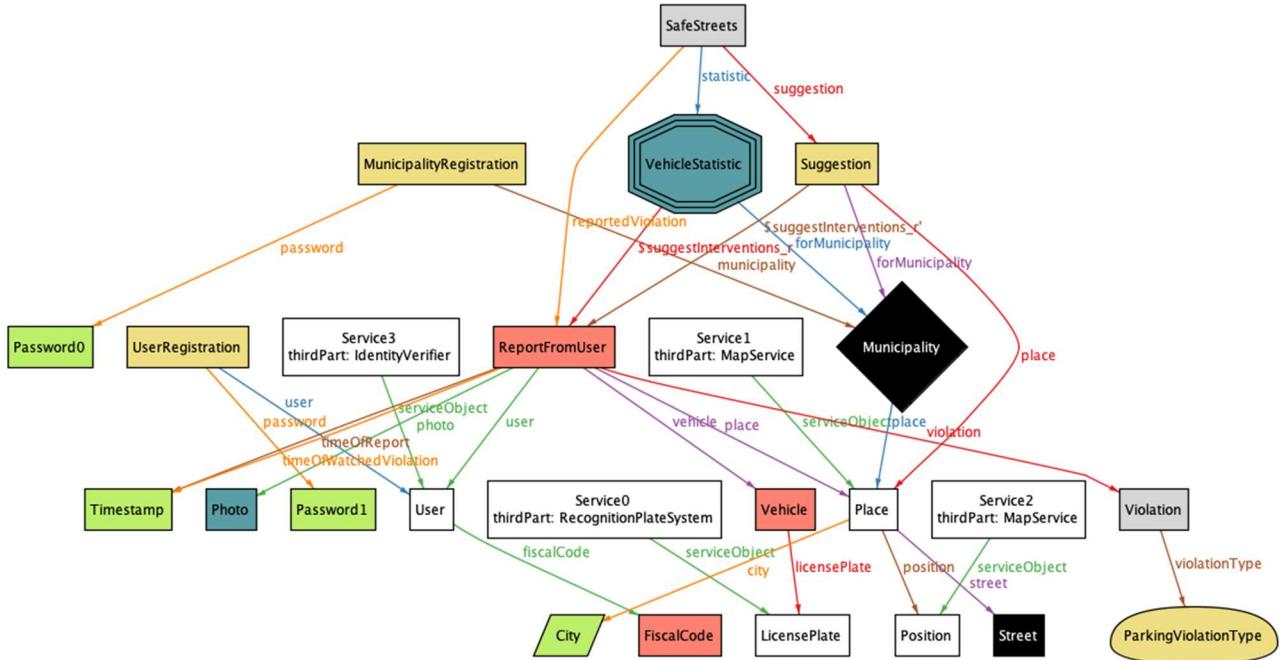


This world shows that there is at least one report (ReportFromUser) made by one user registered in SafeStreets.

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.

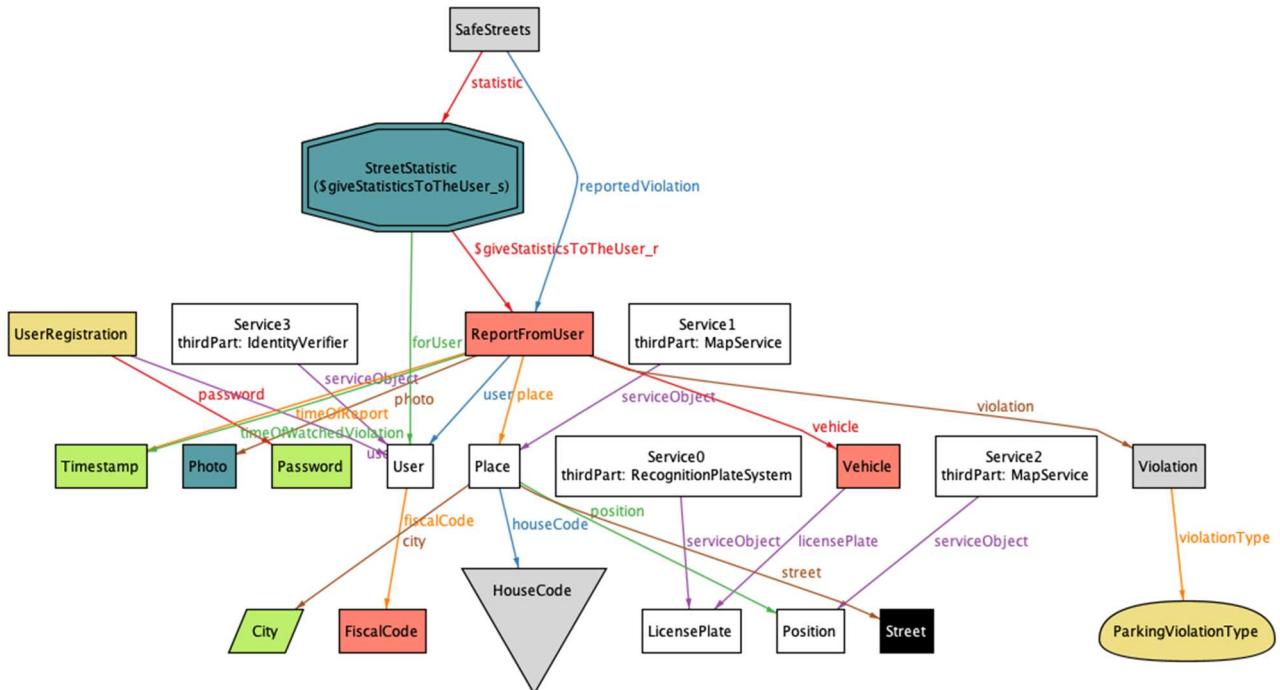
The user is identified by the Identity Verifier with the Service3. The place and the position are identified by the map service and then the license plate is identified by the recognition plate system. Finally, the place contains the specific position, the city, the street and the house code.

## G2: The System suggests possible interventions to the Municipality.



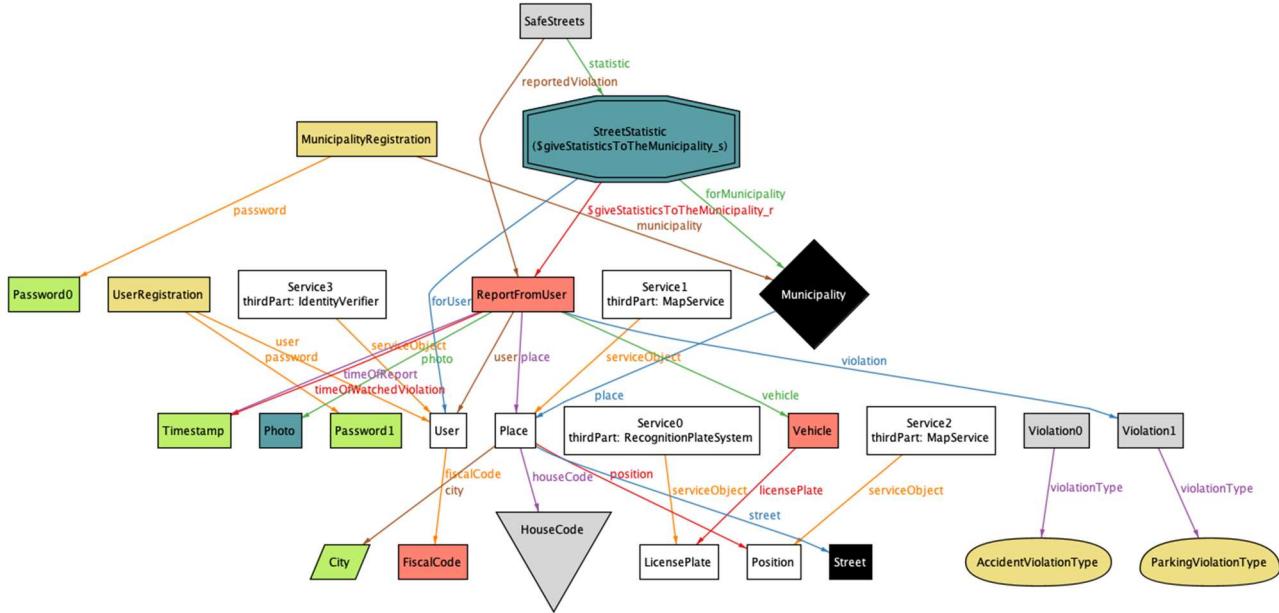
Here it is possible to see that SafeStreets has generated one suggestion for a Municipality in a specific place. The suggestion was made based on the reports stored in SafeStreets.

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



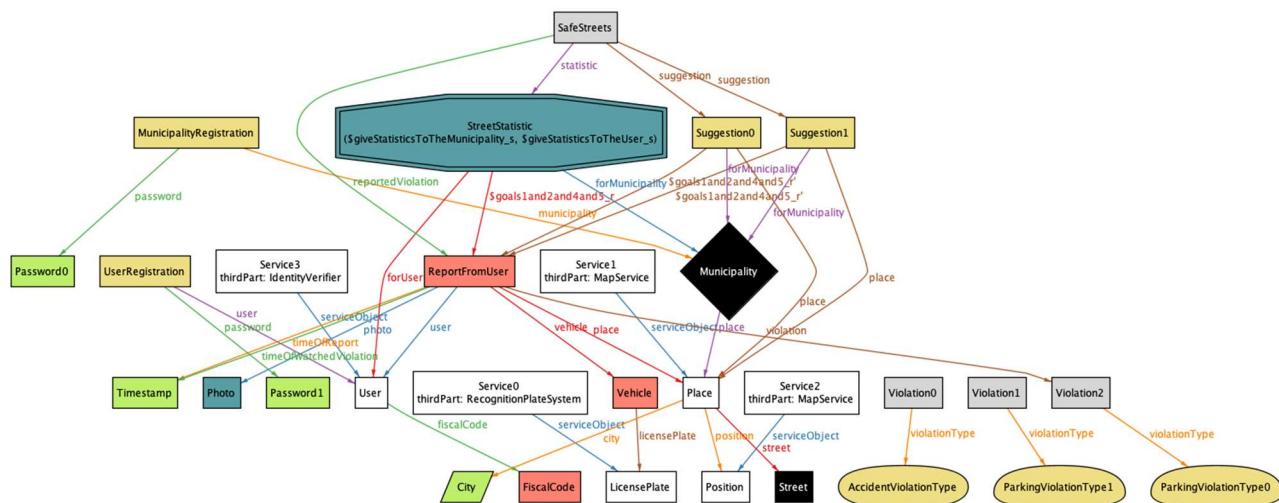
This world shows one statistic about the streets made by SafeStreets for a user.

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



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

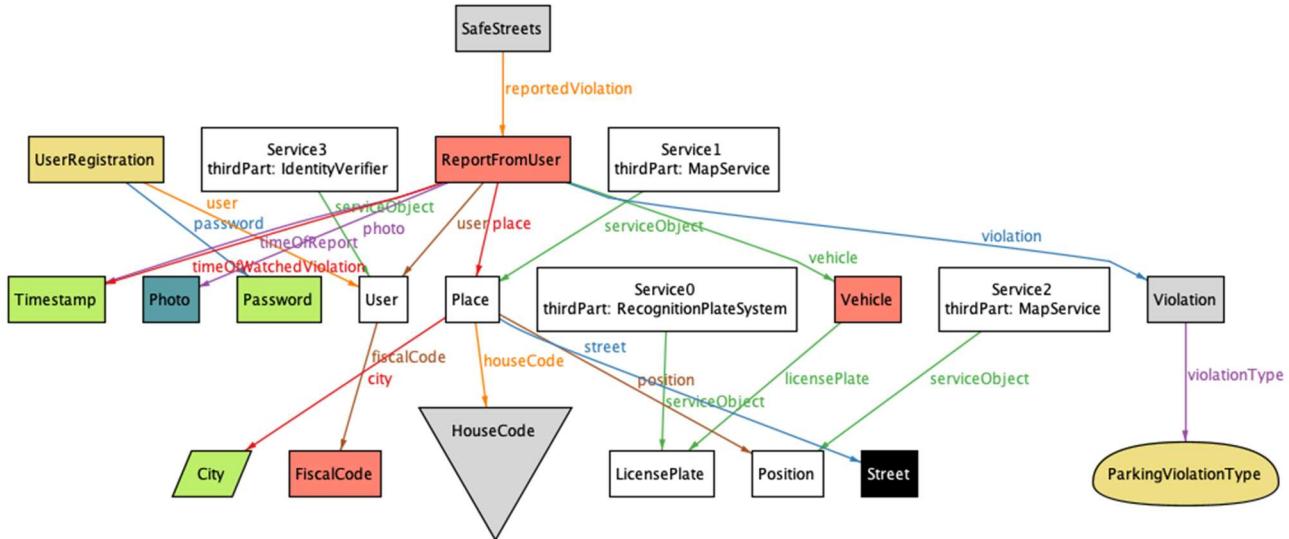
### G1 and G2 and G4 and G5:



This world satisfies the goals 1, 2, 4 and 5.

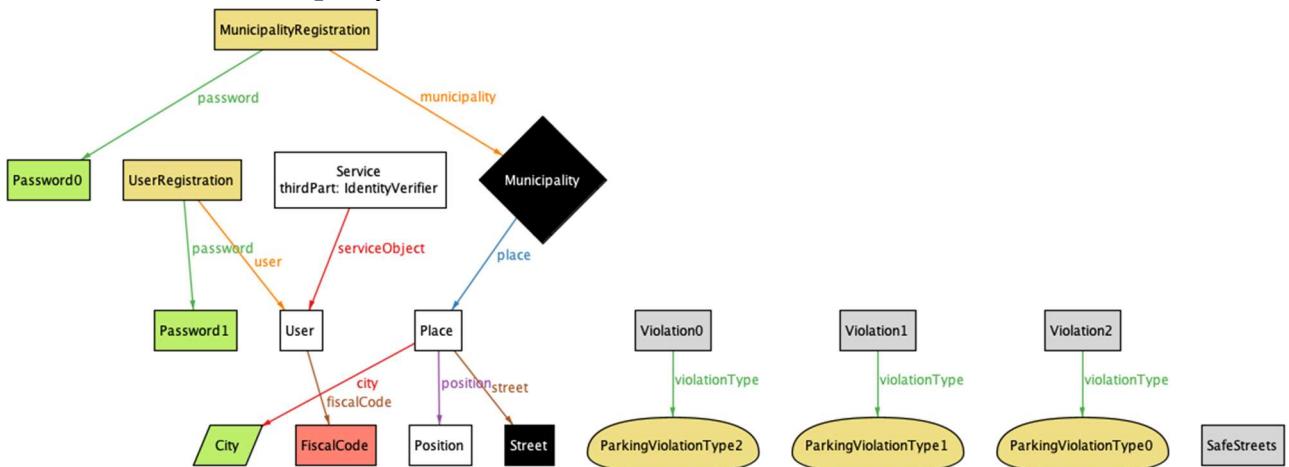
There is one report made by a user, one statistic for one user, one statistic for one Municipality and one suggestion for one Municipality.

### World with no Municipalities:



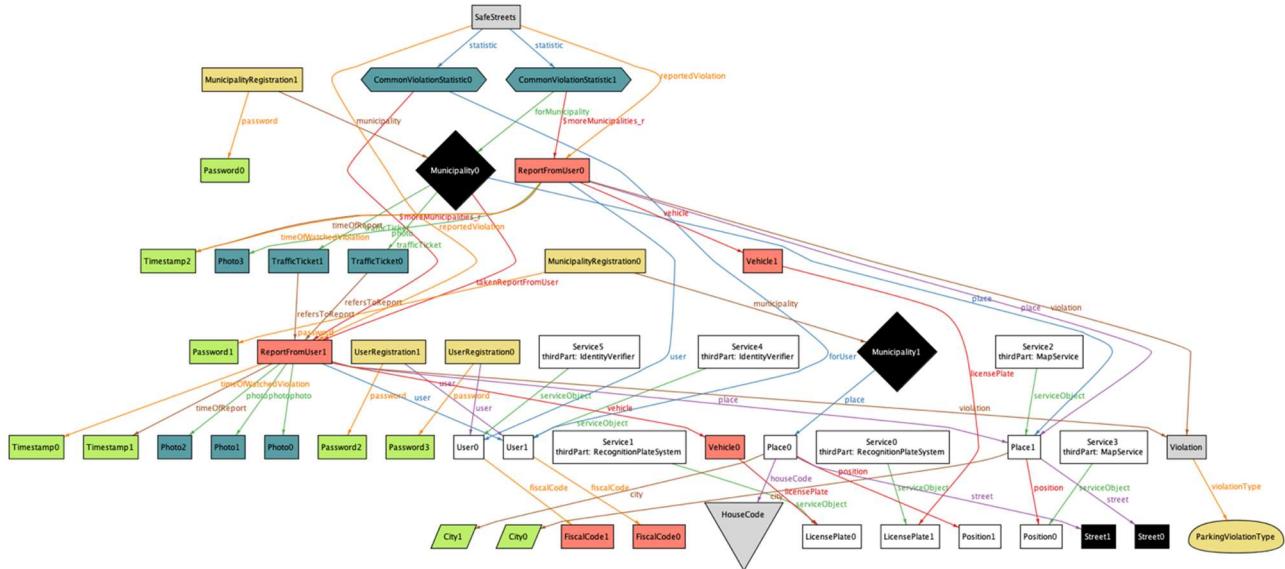
This world shows the consistency of the model even when there are no Municipalities registered in the SafeStreets' system.

### World with one Municipality:



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:



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 an user
    forMunicipality: lone Municipality //means that the statistic is for an Municipality
} {
    //a statistic is always done for an 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 licencePlatesUnique {
    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 RecognitionPlateSystem)
}

//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 suggestion 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 retrieves 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	4
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:
  - “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”