

1. **INTRODUCTION**
  - A. *Purpose*
  - B. *Scope*
  - C. *Definitions, Acronyms, Abbreviations*
  - D. *Revision history*
  - E. *Reference Documents*
  - F. *Document Structure*
2. **ARCHITECTURAL DESIGN**
  - A. *Overview: High-level components and their interaction*
  - C. *Component view*
  - D. *Deployment view*
  - E. *Runtime view: You can use sequence diagrams to describe the way components interact to accomplish specific tasks typically related to your use cases*
  - F. *Component interfaces*
  - G. *Selected architectural styles and patterns: Please explain which styles/patterns you used, why, and how*
  - H. *Other design decisions*
3. **USER INTERFACE DESIGN:** Provide an overview on how the user interface(s) of your system will look like; if you have included this part in the RASD, you can simply refer to what you have already done, possibly, providing here some extensions if applicable.
4. **REQUIREMENTS TRACEABILITY:** Explain how the requirements you have defined in the RASD map to the design elements that you have defined in this document.
5. **IMPLEMENTATION, INTEGRATION AND TEST PLAN:** Identify here the order in which you plan to implement the subcomponents of your system and the order in which you plan to integrate such subcomponents and test the integration.
6. **EFFORT SPENT:** In this section you will include information about the number of hours each group member has worked for this document.
7. **REFERENCES**

<b>1. INTRODUCTION</b>	<b>3</b>
1.1. Purpose	3
1.2. Scope	3
1.3. Definitions, Acronyms, Abbreviations	3
1.4. Revision history	3
1.5. Reference Documents	3
1.6. Document Structure	3
<b>2. ARCHITECTURAL DESIGN</b>	<b>3</b>
2.1. Overview	3
2.2. Component view	3
2.3. Deployment view	3
2.4. Runtime view	3
2.5. Component interfaces	3
2.6. Selected architectural styles and patterns	3
2.7. Other design decisions	3
<b>3. USER INTERFACE DESIGN</b>	<b>3</b>
<b>4. REQUIREMENTS TRACEABILITY</b>	<b>4</b>
<b>5. IMPLEMENTATION, INTEGRATION AND TEST PLAN</b>	<b>4</b>
<b>6. EFFORT SPENT</b>	<b>4</b>
<b>7. REFERENCES</b>	<b>4</b>

<http://www.cloudcomputingpatterns.org/>

## Display and clustering

When a new violation is added it checks whether the same violation already exists, by checking on the same license plate, location, time. If it already exists it is added on the same report.

When the map is displayed, reports of the same violation type are clustered together.

## Suggestions

Suggestions interact with clustering, suggestion are hardcoded.

## Location

Each picture is linked with a location. The report however has only one location. The location in the report can be the average between the locations of the pictures.

## Database

Each user has the list of each report sent.

## Folders

Violation Reports

Clusters

Users

Municipalities

~~Groups~~

Accidents

Issued tickets

## Groups

Unifies the reports that represents the same violation. It is a list of indexes of the reports that represents the same violation. The tuple also has the state of the reports (since it is the same for all reports that represents the same violation)

*Groups merged with municipalities*

## Municipalities

### Structure

#### Municipalities

- Municipality 1
  - Approved
    - Group 1
      - Report1 ID
      - Report2 ID
    - Group 2
  - Rejected
    - Group 4
      - Report1 ID
      - Report2 ID
    - Group 73
  - Confirmed
    - ....
- Municip 2
- Municip 3

### Clusters

It is divided by violation type. Each violation type has the list of all clusters

#### Clusters

- Typeofviolation
  - Cluster1

- Location
- UserSuggestions (if implemented)
- Reports
  - Report1
  - Report2
- Cluster2

## Microservices (triggers)

- GroupingMS
    - Same violation
    - Verify if the violation has already been reported (location, time, licence plate)
  - ClusteringMS
    - Same location/type
  - ApprovingMS
    - Approves the report also using cloud vision
  - MunicipalityDataRetrieverMS
    - Accidents retriever
    - Tickets retriever
  - OnReportConfirmedMS / OnReportRejectedMS
    - Update the state of the reports from the municipality.
    - This is a trigger that when the state of a group of reports is updated, then we update all the corresponding reports in the DB.
- Statisticsc ms

## Reports

Using Google vision the pictures will be checked if they have at least one vehicle in one of their photos. The report has the state, because a user must be able to retrieve his reports and the state, without accessing "Clusters" which contains sensitive information.

## **Interface**

My reports

Settings

## **Web interface**

The interface takes each group that is approved and not yet reviewed.

When the employee clicks on a report, it is opened and the photos are downloaded.

# 1. Introduction

## 1.1. Purpose

TODO

## 1.2. Scope

As seen in the RASD, the idea behind this product is to give to the citizen the possibility to report traffic violations he sees during the day, to the authorities. Normally, the citizen would stop, call the non emergency police number, give information about current location, type of violation, brand of the car, license plate number, ecc., spending a lot of time on the phone just to report a violation. Our purpose is to make all of this quicker and easier. With our application, all that the user must do is snap a few pictures of the car in violation including the license plate and send the report to our system. In turn, the system will send the report to the municipality that operates in the corresponding city. We believe that reducing the effort will lead to an increase in the number of reports and ticket issued, reducing overall traffic violations.

Furthermore, the product will have an additional function: we want to give the citizens and municipalities the opportunity to mine the information regarding our data and that of the municipalities. Both the citizens and the municipalities will be able to see and filter violations occurred on the map of the city.

Finally the system will be able to give the municipality some suggestions on how to improve the condition of the roads. These suggestions will be based on the number of similar violations reported in the proximity of a specific area. For example, if in a certain area a lot of cars park on the street because there are not enough parking lots the system would suggest to increase their number or redesign them in a more space efficient way.

## 1.3. Definitions, Acronyms, Abbreviations

### 1.3.1. Definitions

- Entity: a “natural person”, or a “juridical person” in the case of the municipality.
- User: a citizen registered to the SafeStreets service.
- Municipality: a city or town that has corporate status and local government.
- Timestamp: a representation of date and time.
- Anonymized data: data that don't give any personal information, such as license plate, pictures and names.

- Valid violation report: a violation report composed by at least one picture, exactly one location, exactly one timestamp, exactly one type of violation and the license plate of the vehicle.
- Approved violation report: a valid violation report that may represent a correct violation report.
- Correct violation report: an approved violation report that the municipality evaluated as a traffic violation.
- Area: a district or a neighborhood of a country or city, especially one characterized by a particular feature or activity.
- GPS fix: a position derived from measuring in relation to GPS satellites.

### 1.3.2. Acronyms

- API: Application Programming Interface.
- GPS: Global Positioning System.
- UI: User Interface.
- S2B: Software To Be.
- GDPR: General Data Protection Regulation.
- OS: Operating System.
- RASD: Requirement Analysis and Specification Document
- HTTPS: Hypertext Transfer Protocol Secure
- HTML: Hypertext Markup Language

### 1.3.3. Abbreviations

- Gn: nth goal.
- Dn: nth domain assumption.
- Rn: nth requirement.

## 1.4. Revision history

- Version 1.0: Initial release.

## 1.5. Reference Documents

TODO

## 1.6. Document Structure

TODO

## 2. Architectural design

### 2.1. Overview

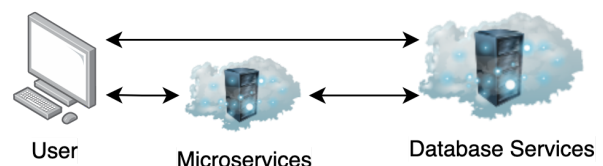
The paradigm that has been followed for designing the SafeStreets system is the Serverless paradigm<sup>[1]</sup>.

This modern approach has been chosen since the functionalities offered by SafeStreets are easy enough to be built on a 2-tier system, where the client interacts “directly” with the databases and the services. So the middle tier is removed and its logic is split on the remaining tiers. The system relies solely on a combination of third-party services, client-side logic and cloud-hosted procedure calls (microservices). This allows it to be extremely scalable and performant with little to no server-management.

Typical Server based architecture



Typical Serverless architecture



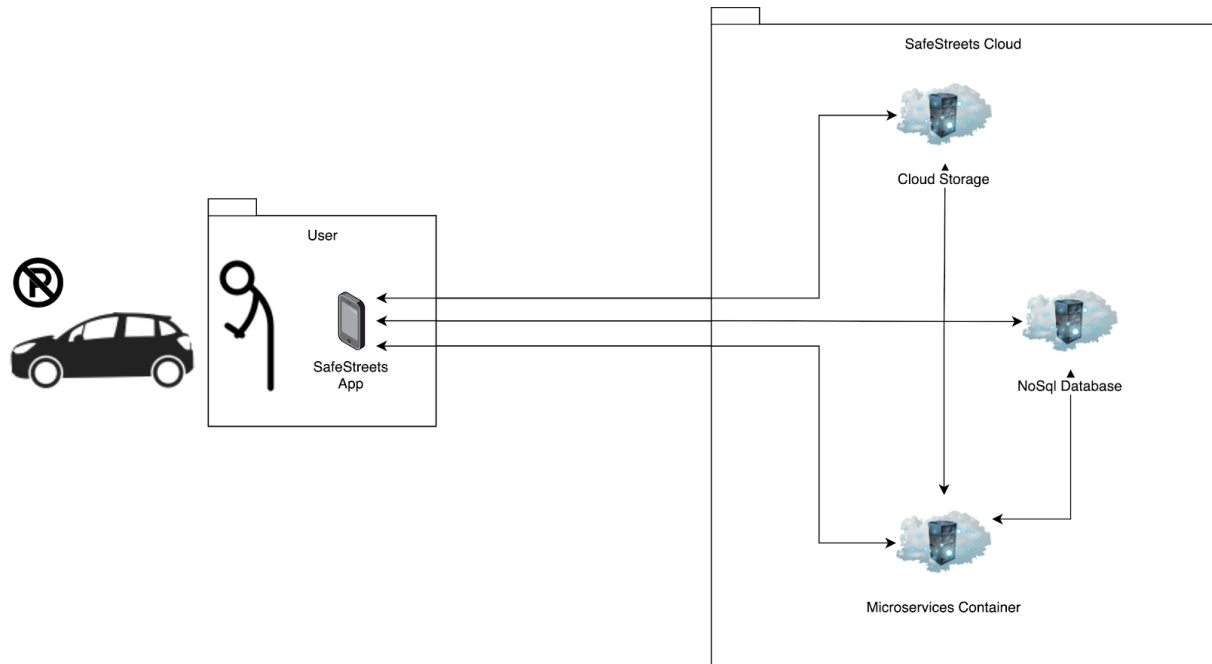
The architecture structure of SafeStreets can be seen from two different points of view: the point of view of the user and the point of view of the municipality.

In the next sections these point of views are represented using a high level view of the architecture using informal diagrams. It is important to note, however, that some simplifications have been made for allowing a better understanding of the architecture:

- Only nodes that are belonging to one of the three main actors in the system (user, municipality, SafeStreets) are displayed.
- The *Authentication Server* node is not represented since it will make the diagram a lot harder to read. This node provides functionalities for authenticating users and municipalities to the other nodes, so it's heavily used when performing login and sign up of users, login of municipalities, and to make sure that documents in the databases are managed according to the security rules.



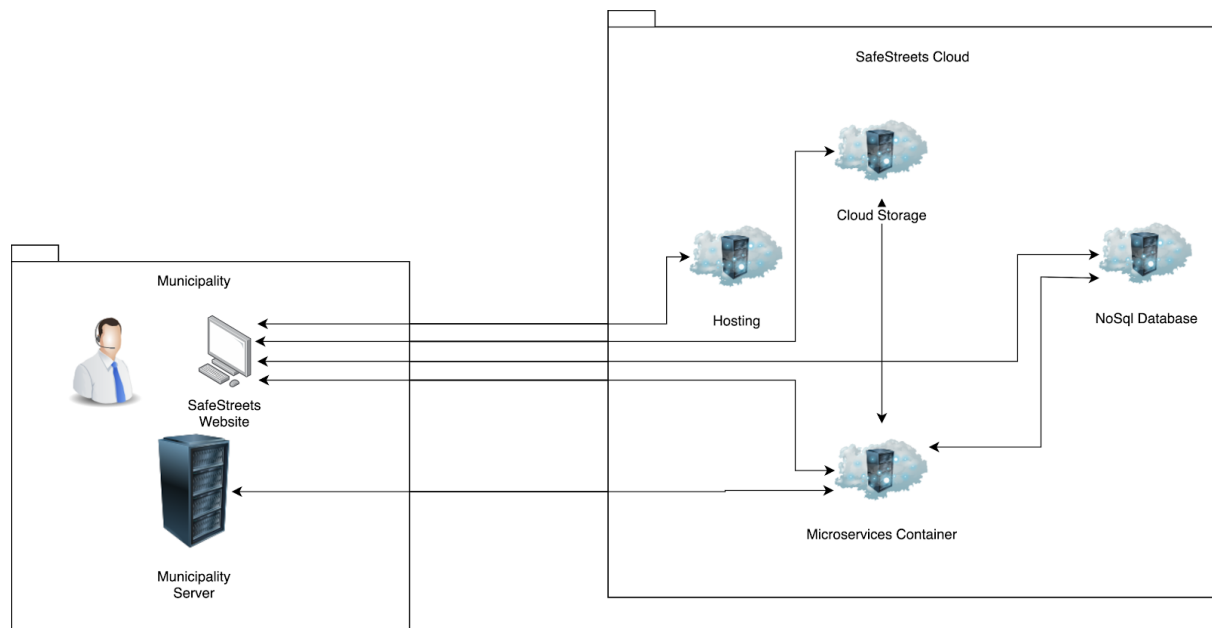
### 2.1.1. User's point of view



The user interacts with 3 nodes primarily:

- **Cloud Storage:** using this node the application of the user can easily upload large files (images) on the cloud.
- **NoSql Database:** by interacting with this node the user's application can read and write data on the database. Note that the access to the database is limited to what the security rules allow (limiting the access of data that the user can't have).
- **Microservices Container:** this node contain stateless functions that are run in their own isolated secure execution context, are scaled automatically, and have a lifecycle independent from other functions. They can modify the database and the storage while also interacting with the user.

### 2.1.2. Municipality's point of view

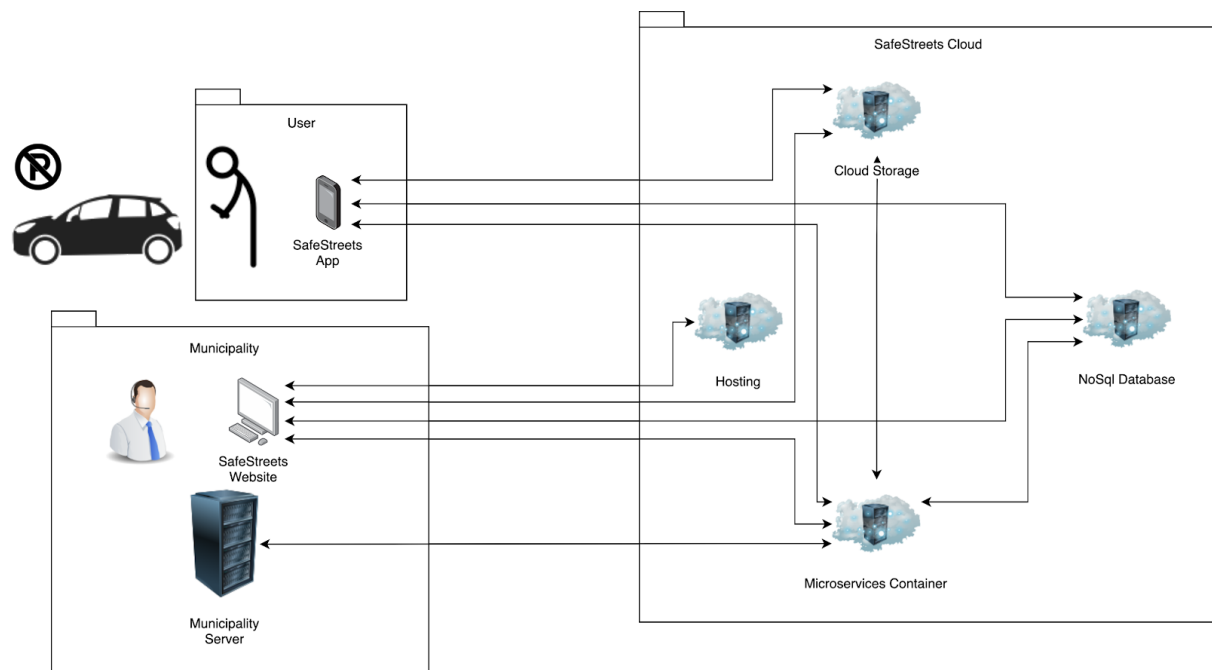


The *Cloud Storage*, the *NoSql Database* and the *Microservice Container* from the user's point of view are still present here, but now their function depends on the municipality behaviour.

However one new node is added in the *SafeStreets Cloud*:

- *Hosting Provider*: this node is a simple server that listens for connections on port 443 (HTTPS) and serves static HTML pages to the municipality.

### 2.1.3. Overall high level view of the architecture



2.2. Component view

2.3. Deployment view

2.4. Runtime view

2.5. Component interfaces

2.6. Selected architectural styles and patterns

2.7. Other design decisions

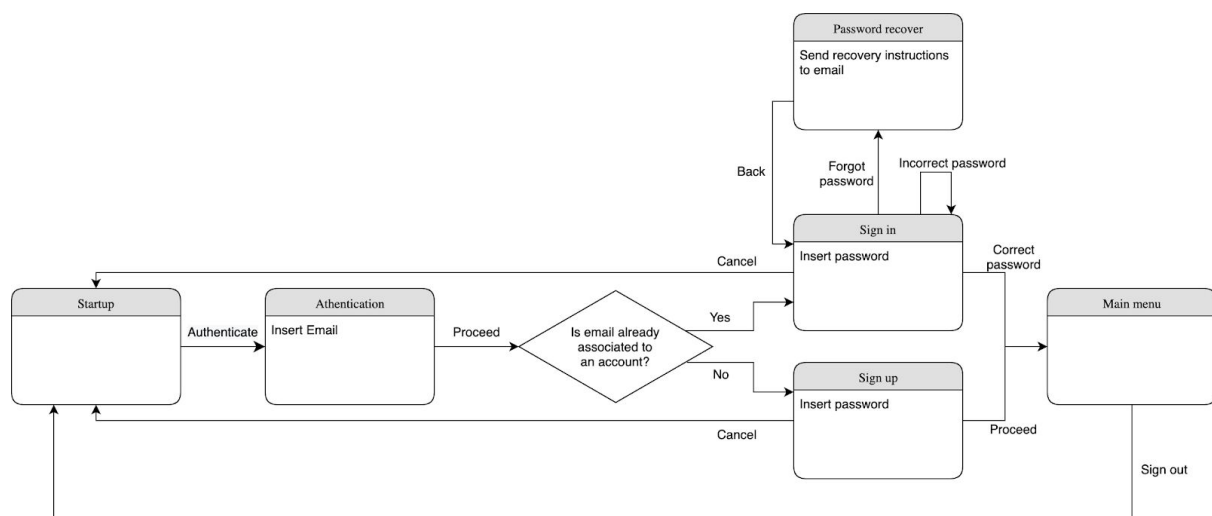
## 3. User interface design

### 3.1. SafeStreets App

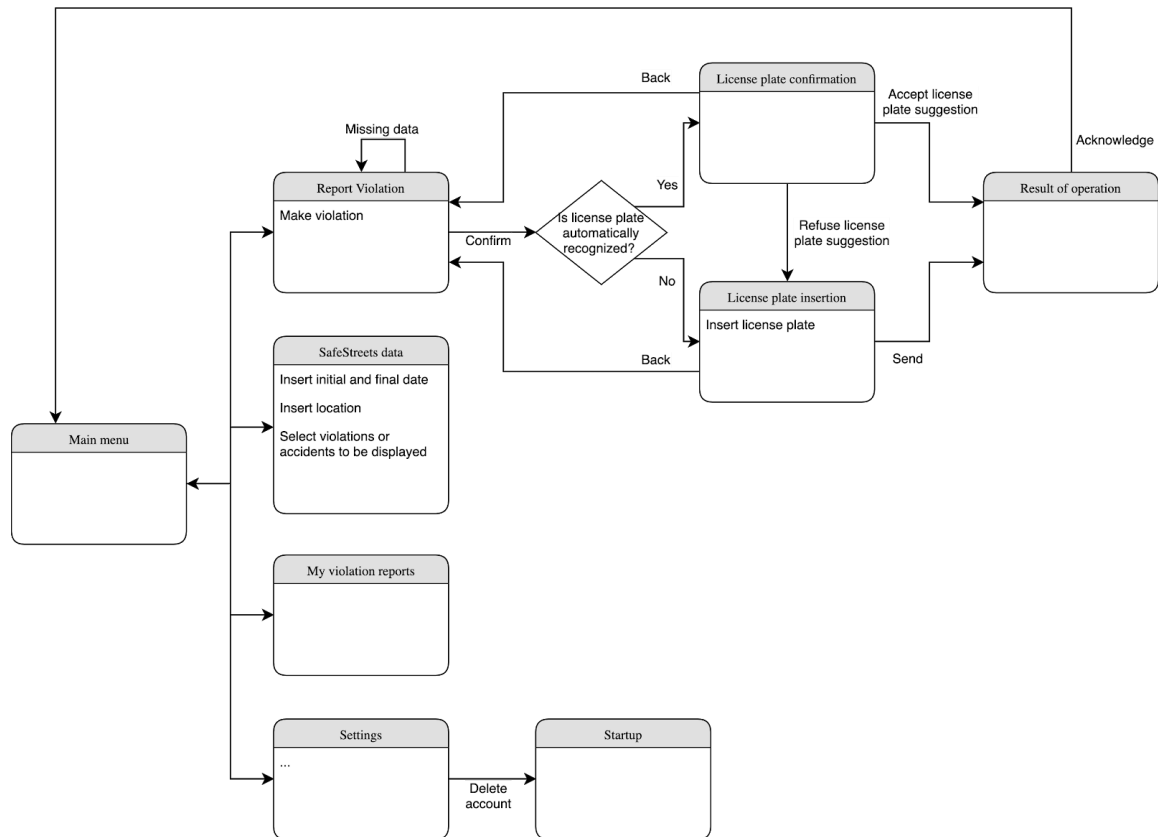
In this section it is shown the user interface design applied to the SafeStreets application.

First are presented the UX diagrams where it is shown the flow followed by the users to utilize the app. There are two UX diagrams to better represent different parts of the application: one for the authentication process of the user and the other for the main features of the app that are available only after the authentication process. Then mockups are presented. The mockups presented here were already developed for the RASD document, but are copied for readability purposes and to associate them with their corresponding page in the UX diagram.

#### 3.1.1. User authentication - UX diagram



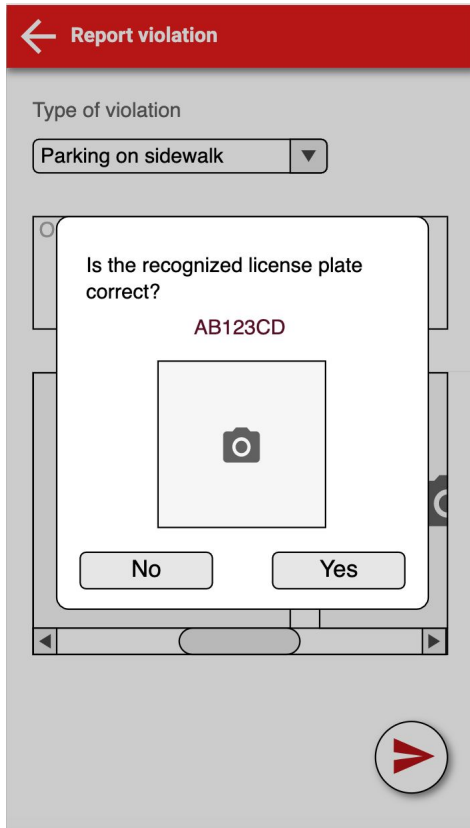
### 3.1.2. App features - UX diagram



### 3.1.3. Mockup of the “Report Violation” page

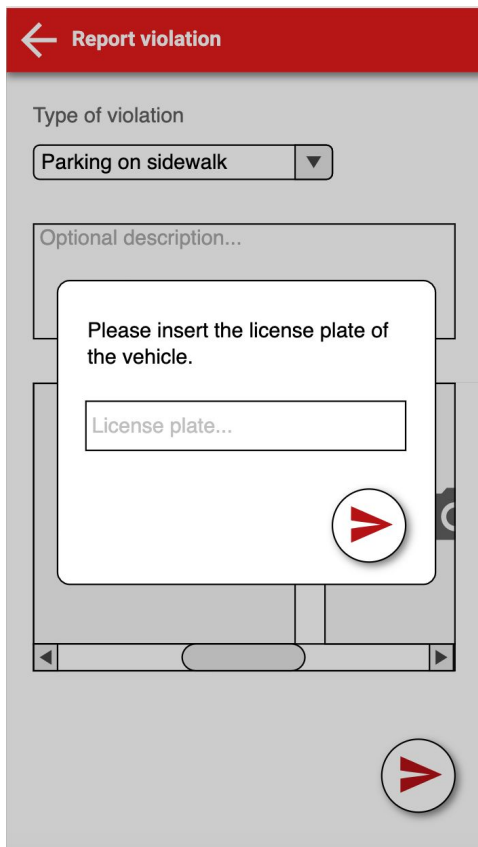
The mockup shows the 'Report violation' screen. At the top is a red header with a back arrow and the text 'Report violation'. Below the header, there is a 'Type of violation' section with a dropdown menu currently showing 'Parking on sidewalk'. Underneath is a text input field labeled 'Optional description...'. The bottom half of the screen is occupied by a large image area with a camera icon in the center, indicating where the user can take a photo. At the very bottom, there is a red circular button with a white right-pointing arrow, serving as a 'Next' or 'Submit' button.

### 3.1.4. Mockup of the “License plate confirmation” page




The mockup shows a mobile application interface for reporting a violation. At the top is a red header bar with a white back arrow and the text "Report violation". Below this is a section titled "Type of violation" with a dropdown menu currently showing "Parking on sidewalk". A modal dialog box is centered on the screen, asking "Is the recognized license plate correct?". Inside the dialog, the license plate "AB123CD" is displayed above a camera icon. At the bottom of the dialog are two buttons: "No" and "Yes". The background of the app is light gray, and there is a large red arrow button in the bottom right corner.

### 3.1.5. Mockup of the “License plate insertion” page




The mockup shows a mobile application interface for reporting a violation. At the top is a red header bar with a white back arrow and the text "Report violation". Below this is a section titled "Type of violation" with a dropdown menu currently showing "Parking on sidewalk". Below the dropdown is a text input field labeled "Optional description...". A modal dialog box is centered on the screen, asking "Please insert the license plate of the vehicle.". Inside the dialog is a text input field with the placeholder text "License plate...". At the bottom right of the dialog is a red arrow button. The background of the app is light gray, and there is a large red arrow button in the bottom right corner.


### 3.1.6. Mockup of the “SafeStreets data” page

 **Violations on the territory**

Start date

23/10/2019 

End date

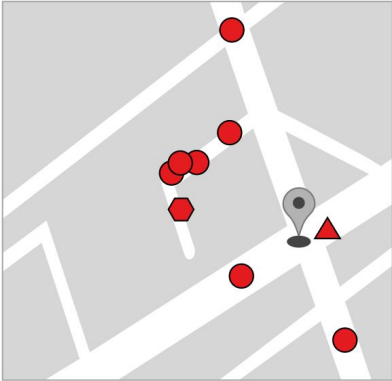
10/11/2019 

Display...

☒ Violations 

Select types

☒ Accidents



### 3.2. Municipality web interface

TODO

## 4. Requirements traceability

### 3.2.4.1. Satisfying goal 1

[G1] A citizen can report a violation.

- [D1] A citizen who wishes to report a violation has a mobile phone with the SafeStreets app installed.
- [R1] A citizen not yet registered must be able to sign up and become a user.
- [R2] The application must allow users to authenticate through log in.
- [R3] The application must allow users to report a violation.

### 3.2.4.2. Satisfying goal 2

[G2] A violation report received by the system must have enough information to be valid, i.e. has at least one picture of the violation, exactly one GPS position, exactly one timestamp, exactly one type of violation and the license plate of the vehicle.

- [R4] The application must allow reporting of violations only from devices equipped with a GPS receiver which are in the conditions to obtain a GPS fix.
- [R5] The application must allow reporting of violations only from devices equipped with a camera.
- [R6] The application must allow reporting of violations only from devices with an active internet connection.
- [R7] A user has the possibility to specify the type of the reported violation choosing from a list.
- [R8] The application creates a violation report with at least one picture, exactly one timestamp, exactly one location, exactly one type of violation and the license plate of the vehicle.

### 3.2.4.3. Satisfying goal 3

[G3] Users and municipality can retrieve information about violations, accidents and issued tickets in a certain area, with different levels of visibility.

- [D2] Municipality offers a service to retrieve information about accidents.
- [D3] Municipality offers a service to retrieve information about tickets.
- [R9] The system saves all the information regarding the violations reported by users.
- [R10] The system must be able to retrieve data regarding issued tickets and accidents from the municipality.
- [R11] The system must offer an interface to retrieve information about violations, accidents and issued tickets.
- [R12] The system must show to users only anonymized data.
- [R13] The system must analyze valid violations report and approve which of them may represent a correct violation.



#### *3.2.4.4. Satisfying goal 4*

[G4] Municipality will be able to retrieve suggestions for possible interventions in order to increase safety.

- [D2] Municipality offers a service to retrieve information about accidents.
- [D3] Municipality offers a service to retrieve information about tickets.
- [R9] The system saves all the information regarding the violations reported by users.
- [R10] The system must be able to retrieve data regarding issued tickets and accidents from the municipality.
- [R14] The system must be able to elaborate data about violations, accidents, issued tickets and generate useful suggestions about possible interventions.
- [R15] The system must offer an interface to the municipality for retrieving useful suggestions about possible interventions.

#### *3.2.4.5. Satisfying goal 5*

[G5] Municipality receives enough information about the violation in order to issue a ticket.

- [D4] When a device is able to obtain a GPS fix, the location provided has an accuracy of at least 20 meters.
- [D5] The municipality checks if approved violation reports can actually represent a traffic violation (for example they could check if the license plate actually corresponds to the car model in the picture).
- [R8] The application creates a violation report with at least one picture, exactly one timestamp, exactly one location, exactly one type of violation and the license plate of the vehicle.
- [R13] The system must analyze valid violations report and approve which of them may represent a correct violation.
- [R16] The system must offer a service to the municipality for retrieving correct violations report.

#### *3.2.4.6. Satisfying goal 6*

[G6] The integrity of the violation report is guaranteed.

- [D6] Data transferred through connections that use modern encryption protocols can not be manipulated.
- [R17] The application will allow using pictures in a violation report only if the picture was taken by the application itself, preventing it to be manipulated on the device.
- [R18] All connections used by the system use modern encryption protocols.
- [R19] Data saved in the server can not be manipulated.

## 5. Implementation, integration and testing plan

The following is a table of the main features of our system, complete with their importance and their difficulty of implementation. (why is it used)

Feature	Relevance	Difficulty of implementation
Report violations	High	Medium
Visualize data about violation on the map	High	High
(User) See own violation	Medium	Low
(Municipality) Retrieve and review violations	High	High

### Feature 1: Report violations

This feature is indeed the foundation on which the app is built. Only when a large pool of data is collected, the users will be able to use the application to retrieve and visualize data.

The difficulty of implementing this is Medium because there is the necessity (AI to recognize photos)

## 6. Effort spent

General, for all (needs to be copied in tables)

22/11 - 3 hr 0 min - discussion about architecture

24/11 - 1 hr 15 min - architecture diagram

25/11 - 1 hr 0 min - Component view

28/11 - 1 hr 0 min - Component view

Dennis Motta		
Date	Task(s)	Time
23/11	User interface design	2 hr 0 min
24/11	User interface design	1 hr 0 min
26/11	Architectural design overview	2 hr 0 min
27/11	Fixes after DD review; Architectural design overview	1 hr 45 min
<b>Total:</b>		

## 7. References

1. "Serverless paradigm". [https://en.wikipedia.org/wiki/Serverless\\_computing](https://en.wikipedia.org/wiki/Serverless_computing)
- 2.