A.A. 2019/2020

POLITECNICO
MILANO 1863

Computer Science and Engineering

Software Engineering 2

DD

Design Document

version 1.1

Lorenzo Amaranto
Stefano De Cillis
Tayfun Gumus

1. Introduction

1.1 Purpose

The aim of this phase is to proffer technical details about the structure of the system to be developed, SafeStreets.
The design documentation provides written documentation of the design factors and the architectural choices that have to be made in the design in order to satisfy the business and technical requirements.

1.2 Scope

The idea of SafeStreets is to improve the condition of the general mobility in the city exploiting a virtuous relationship of collaboration between citizens reporting violations and public authorities. The S2B has been thought to guarantee the privacy of citizens who report violations and to help municipality to check the city through a distributed network of reports.

Registered citizens will have the possibility to report traffic infractions providing pictures and informations about the violation and its type. In addiction, an optional field will be the license plate input in order to help the system recognize it. The system will evaluate either to accept or to refuse reports, that must be compliant to minimum acceptance requirements. Then, Safestreets will validate reports. Citizen can also question the system in order to retrieve various kind of informations, like trends, based on the type of violation and other selectable parameters. Instead more informations can be mined by public authorities that will be also able to visualize the rankings about vehicles which committed more infractions and citizens who reported more violations.

SafeStreets shares a private interface to public authorities with which it can retrieve data and cross them with its own informations. The purpose it is to suggest reasonable solutions to each identified problem, such as adding barriers between streets and bike lanes or adding stakes on the sidewalk due to an aggressive parking, suggesting more checks in areas where there are double parking or car parked in reserved places for people with disabilities.

All the data collected by the system will be processed and made available for public authorities. In this way, the Local Police will use refined data in order to generate traffic tickets. Public entities can evaluate the influence of the system on the community with the statistics SafeStreets builds, retrieving indexes and trends, by using the feedbacks received by authorities.

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

- **User**: generic end user who interacts with the application.
- **Citizen**: users who reports the infraction.
- **Public Authority**: public institutions,such as Municipality, City Hall, Local Police.
- **Infraction**: a violation or infringement of a parking concerned law.
- **Possible Solution**: answer that the system gives back to improve the condition of an area classified as unsafe.
- **Unsafe Area**: cut of the city with at least 100 accidents and 250 validated report per year.
- **System, Platform**: SafeStreets.
- **Statistics**: elaborated report informations.
    1. For citizens, they include:
        A. Sort of the streets according to the number of violations associated (abs value).
        B. Sort of the streets according to the number of associated violations/per type of violation.
        C. Trend of the reported violations during the day.
    2. For Public Authorities, they include:
        A. All the Statistics available for citizens.
        B. Vehicles that have more than thirty validated reports.
        C. Ranking of citizens that reported violations.

- **TI-Statistics:** information extracted analyzing feedback.
- **Feedback**: daily record sent by the Local Police concerning traffic tickets issued.
- **Report**: a formal account of events given by a witness that any citizens can provide to the system with a list of pictures and a filled form with informations about the infraction.
- **Accepted Report**: state of a report that is compliant with the acceptance parameters.
- **Acceptance parameters**:
    1. Data and time must not be in the future.
    2. The licence plate must be inserted with the right number of characters and in the right position.
    3. There is no duplicate of the image in the database.
    4. In the database there is not a report inserted with FC, license plate, data and location all of them equal to the candidate report.

- **Acceptance procedure**: steps the S2B will perform to establish if the report fits the acceptance parameters. It is a N.S.C. to discard any doubt about the usability of that specific report in the validation procedure.
- **Valid Report**: entity created based on accepted reports that have accomplished the validation process.
- **Validation Process**: two reports of the same violations are needed in order that violation valid. If the second report does not come within 30 days, the first report is not valid.
- **Adult**: an eighteen years old individual or older.
- **Limited/Italian local reality**: any municipality in Italy with at least 100.000 residents.
- **Updates =** Statistics available to the different kinds of users of SafeStreets. They comprehend also the processes of notification of the users.
- **Notification =** A newly identified unsafe area (for the municipality) and the outcome (accepted/refused) of an uploaded report (for a citizen).
- **Automated Undergoing Process =** Exchange of information related to the tickets issued by the Local Police, the number of incidents occurred in the territory of the municipality, and the creation/activation of a Citizen's account.
- **Database Connection** = It is a protocol that applications use in order to connect and send queries to the database. Some widely used database connection protocols are ODBC and JDBC.
- **Intranet** = An intranet is a computer network for sharing corporate information, collaboration tools, operational systems, and other computing services only within an organization, and to the exclusion of access by outsiders to the organization.
- **Database Management System** = An external component that allow other programs to query and insert data in the database.
- **Client-side** = Within computer networks, the term client side indicates the processing operations performed by a client in a client-server architecture. It therefore represents the front-end of a web application with multi-tier architecture.
- **Server-side** = In computer networks, the expression on the server side refers to operations performed by the server in a client-server environment, countering everything that is performed on the client (client side).
  It therefore represents the back-end of a web application with multi-tier architecture.
- **Data Warehouse** = A data warehouse is a centralized repository of information that can be analyzed to make more informed decisions. Data flows into the data warehouse from transactional systems, relational databases and other sources, normally at regular intervals.
- **E.T.L.** = It is the general procedure of copying data from one or more sources into a destination system which represents the data differently from the source(s) or in a different context than the source(s). Data extraction involves extracting data from homogeneous or heterogeneous sources; for this specific project from an E.R. schema. Data transformation processes data by data cleansing and transforming them into a

proper storage format/structure for the purposes of querying and analysis; finally, data loading describes the insertion of data into the final target database such as a data warehouse considering the instance of SafeStreets.

1.3.2 Acronyms

- **G.U.I.** = Graphical User Interface.
- **A.U.P.** = Automated Undergoing Process.
- **R.A.S.D.** = Requirement Analysis and Specification Document.
- **D.D.** = Design Document.
- **D.B.C.** = Database Connection.
- **D.B.M.S.** = Database Management System.
- **E.T.L.** = Extract Transform Load.
- **N.S.C.** = Necessary and Sufficient Condition.
- **S2B** = Software To Be.

1.3.3 Abbreviations

1.4 Document Structure

The D.D. is composed by seven sections, each one presented below:

- **Chapter 1**: This section gives the purpose of the document and the scope of the project in the first place. Secondly it provides lists of acronyms, definitions and abbreviations useful in better specifying words that might sound ambiguous if not clearly defined.
- **Chapter 2**: This section presents the different layers in which the system is divided in terms of components, considering their physical distributions and their interactions. Applied architectural and design patterns are mentioned.
- **Chapter 3**: This section redirects to the part of the R.A.S.D. where the UI are displayed.
- **Chapter 4**: This section connects the requirements defined in the R.A.S.D. to the elements defined in this document.
- **Chapter 5**: In this section a detailed explanation about the implementation, integration and testing phases,along with their scheduling, is carried out.
- **Chapter 6**: In this section are provided informations about the total amount of hours each member of the group spent working at the document.
- **Chapter 7**: In this section are listed the documents used as sources in drafting this document.

2. Architectural Design

2.1 Overview

In the diagram below an high level representation of the architecture of the system is shown. Notably, a three tier architecture is used to organize the different layers of software. In the Domain Logic Layer, it has been chosen to split upon two different groups of logical blocks two logically different functionalities the system perform in order to guarantee an high level of isolation and modularity. One group refers to the exchange of data, the other one incorporates the back-end logic. Nonetheless it has also to be considered the benefits that an architecture divided this way can provide, computationally speaking.

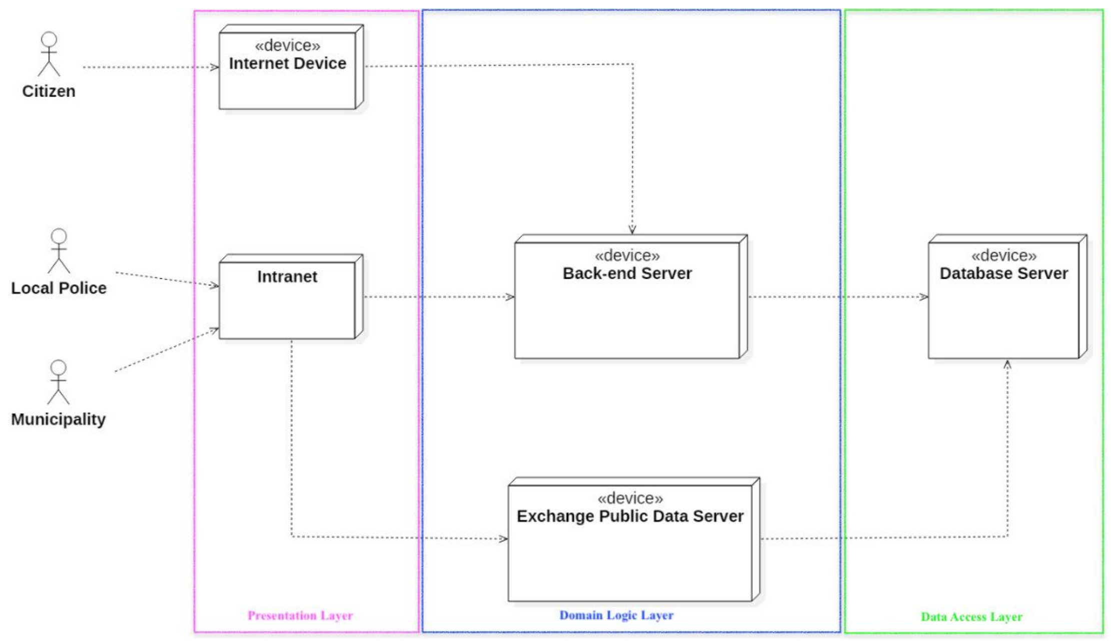

Fig. 1 High Level Architecture Diagram

- The **presentation** is the front-end **layer** in the 3-tier system and consists of all the user interfaces. This user interfaces are graphical one accessible through a web browser and which displays content and information useful to an end user to exploit the services provided by the system.

- The **domain logic layer** constitutes the back-end of the 3-tiered system. It is in charge of the functional business logic which drives the application's core capabilities. It has to be remarked that, since both end users accesses SafeStreets through a web browser, citizens and P.A. exploiting respectively whichever device capable to access Internet and the predisposed Intranet, there is no need of including part of the logic layer in the presentation, this way making the the client much more lighter.

- At the end, the **data access layer** manages and provides access to the data, with store and retrieve operations. Different databases are managed by the DBMS. Depending on the customer needs, the design, chosen in order to store the data, is composed by a relational database and a data warehouse, the latter further explained later in the DD.

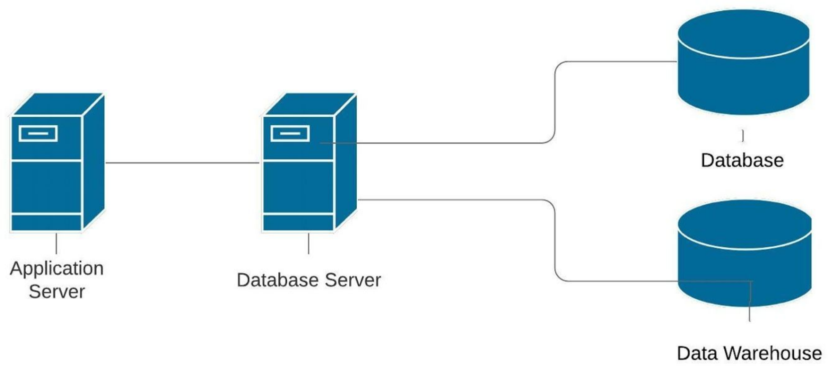

Fig.2 Database layer inspection diagram

2.2 Component views

It follows the diagram where it's shaped internal structure of SafeStreets, modeled considering all its designed components and interfaces, along with their relations. For this project, two main components are identified: SafeStreets Clients and Data Dispatching System.

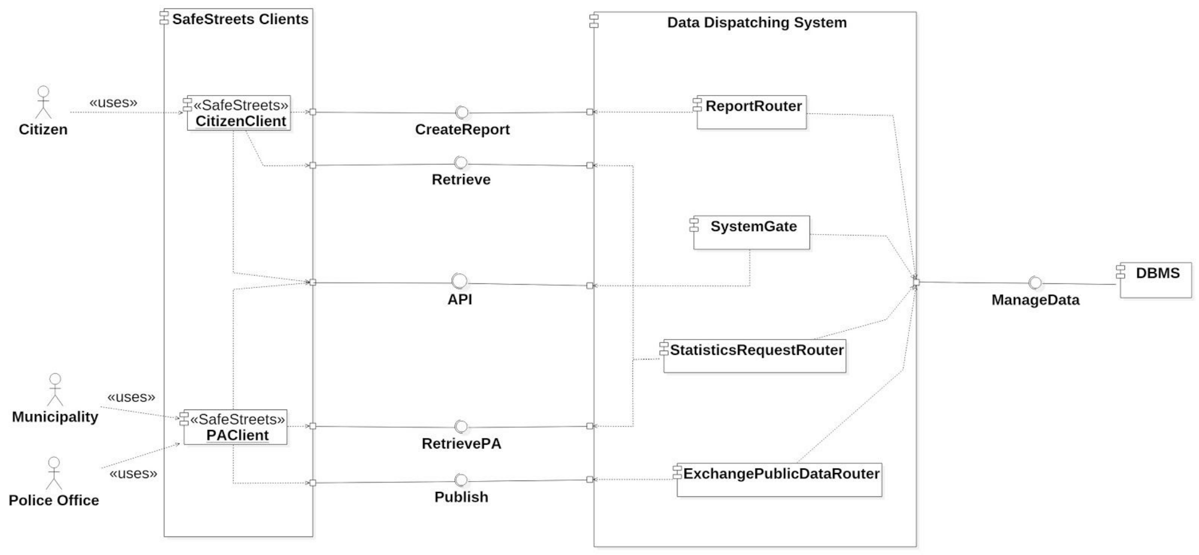

Fig.3 Component Diagram

The **Data Dispatching System** behaves like a middle layer between SafeStreets Clients and the DBMS component. As middle layer in a centralized system, it has the responsibility to handle all the data between client-side and backend-side.

The **Data Dispatching System** is divided in **ReportRouter**, **StatisticsRequestRouter**, **ExchangePublicDataRouter** and **SystemGate**.
The first one is used to address all the reports coming from citizens to the DBMS, further providing to evaluate them with particular procedures, acceptance and validation, already detailed in the R.A.S.D. and in the rest of the DD.
ExchangePublicDataRouter aims to forward all the data coming from the PAs to the right handler based on the information they are pushing in the **Data Dispatching System**. In accordance with the assumptions in the R.A.S.D. , both Municipality and Police Office can publish on our system.
The **StatisticsRequestRouter** takes in account all the requests from each client and routes them back responses with the requested statistic. Different interfaces are used based on the client and the level of visibility of the data treated.
It also takes in account all the operations needed for the computation of all the Statistics that will be made available from the System to all its different users.
SystemGate is in charge of all the operations concerning the access of all users to SafeStreets. Its purpose is to validate each session in which client and server sides are communicating in order to make ensure that resources are available based on the roles of the client and information leakage is avoided.

SafeStreets Clients component integrates two types of client, one for each role. Each type has its own purpose thereby different interfaces are used. Both Municipality and Police Office are seen as the same entity therefore they use the same client. So, by design, each PA is treated in the same way.

2.2.1 ReportRouter Component view

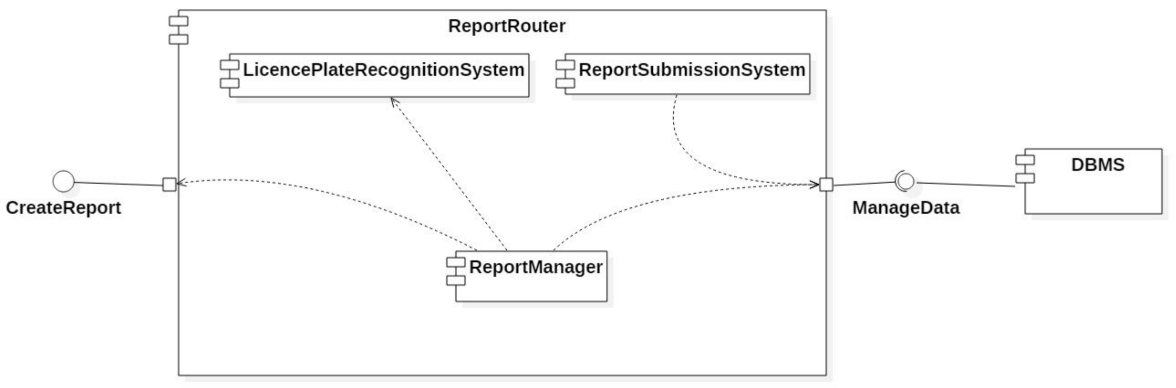

Fig.4 ReportRouter Component Diagram

As the **ReportRouter** component includes several important functionalities of the whole Safestreets system, it has been chosen to illustrate more in detail the components it is constituted by, along with the interfaces involved in the process.
Its components are:

- **ReportManager**: it handles the extraction of all the data Citizens put as input in an object Report, considering also the eventual manually inserted license plate number. It also handles all the operations concerning the application of the acceptance procedure, with the consequent notification of the results to the associated Citizen, and the validation rules, crucial to put in place the "chain of custody of information" required. For the extraction of the the license plate from the picture itself it will use the logic implemented in the LicensePlateRecognitionSystem, described below.

- **LicensePlateRecognitionSystem**: The algorithm that, from the picture provided as input extract the license plate number of the vehicle committing the supposed violation is here implemented.

- **ReportSubmissionSystem**: it handles all the actions needed in order to properly store the reports into the Database.

2.2.2 Data access layer - Data structure description

Fig.5-6 shows how data are stored into the database and what are the relationships between them. Yet, in order to guarantee an high level of comprehensibility of the figure, some hypotheses need to be clearly stated:

- The S2B will consider not only Italian license plate, but works with foreign license plates too. This is because, as it has been said in the R.A.S.D. SafeStreets is thought to work with *Limited/Italian local reality*. Still, vehicles can come from abroad country too. Fifteen characters is set to be the maximum length of the license plate for security reasons.
- The *violationType* is an input inserted by the Citizens, selected among the options provided by SafeStreets.
- For the *violationStreet* field we assume that the streets are the ones of the city where the system is implemented.
- The *licensePlate* field is assumed to be set by SafeStreets after the elaboration process that extracts from the inputs the license plate.
- The validation process is carried out by SafeStreets.
- Statistics are not assumed as fixed data structures, hence they're not inserted in the database.

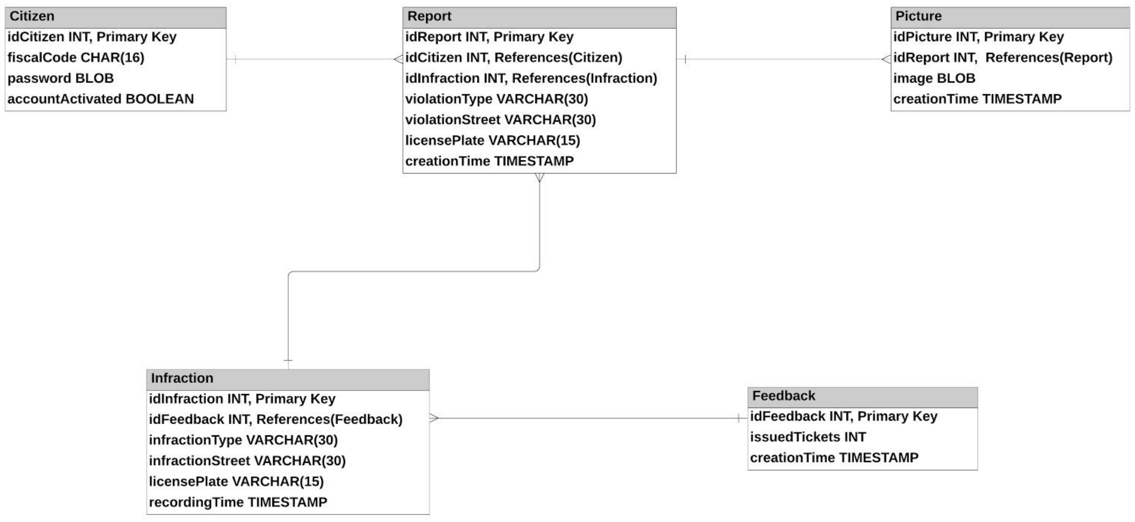
Citizen
idCitizen INT, Primary Key
fiscalCode CHAR(16)
password BLOB
accountActivated BOOLEAN
Report
idReport INT, Primary Key
idCitizen INT, References(Citizen)
idInfraction INT, References(Infraction)
violationType VARCHAR(30)
violationStreet VARCHAR(30)
licensePlate VARCHAR(15)
creationTime TIMESTAMP
Picture
idPicture INT, Primary Key
idReport INT, References(Report)
image BLOB
creationTime TIMESTAMP
Infraction
idInfraction INT, Primary Key
idFeedback INT, References(Feedback)
infractionType VARCHAR(30)
infractionStreet VARCHAR(30)
licensePlate VARCHAR(15)
recordingTime TIMESTAMP
Feedback
idFeedback INT, Primary Key
issuedTickets INT
creationTime TIMESTAMP

Fig.5 E.R. Diagram

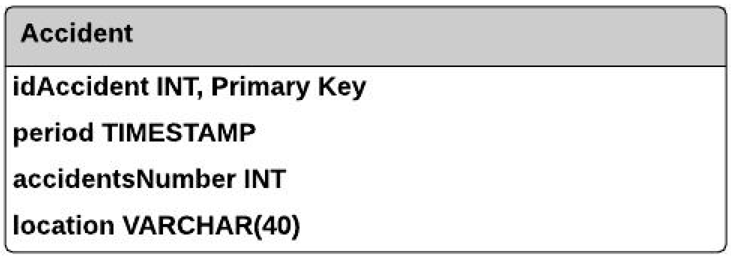

Fig.6 E.R. Diagram

The *Accident* Entity is used to model data uploaded by the municipality concerning the number of accidents occurred in its territory. Crossing these informations with the one SafeStreets has about the violations, unsafe areas are identified. The crossing is feasible exploiting the fields *location*, *period* of the *Accident* entity, and the fields *recordingTime* and *infractionStreet* of entity *Violation.* The field *infractionType* will be used to suggest possible solutions for the specific area.

2.3 Deployment view

In order to illustrate the system in terms of its nodes, and how each component previously identified is located upon those nodes, a deployment diagram of the system is reported below. All the relevant node of the system will be located Server-side. This is useful to highlight once again the thin client condition by design of the client.
Server-side section of the system comprehend Back-end Server, Exchange Public Data Server and Database Server.

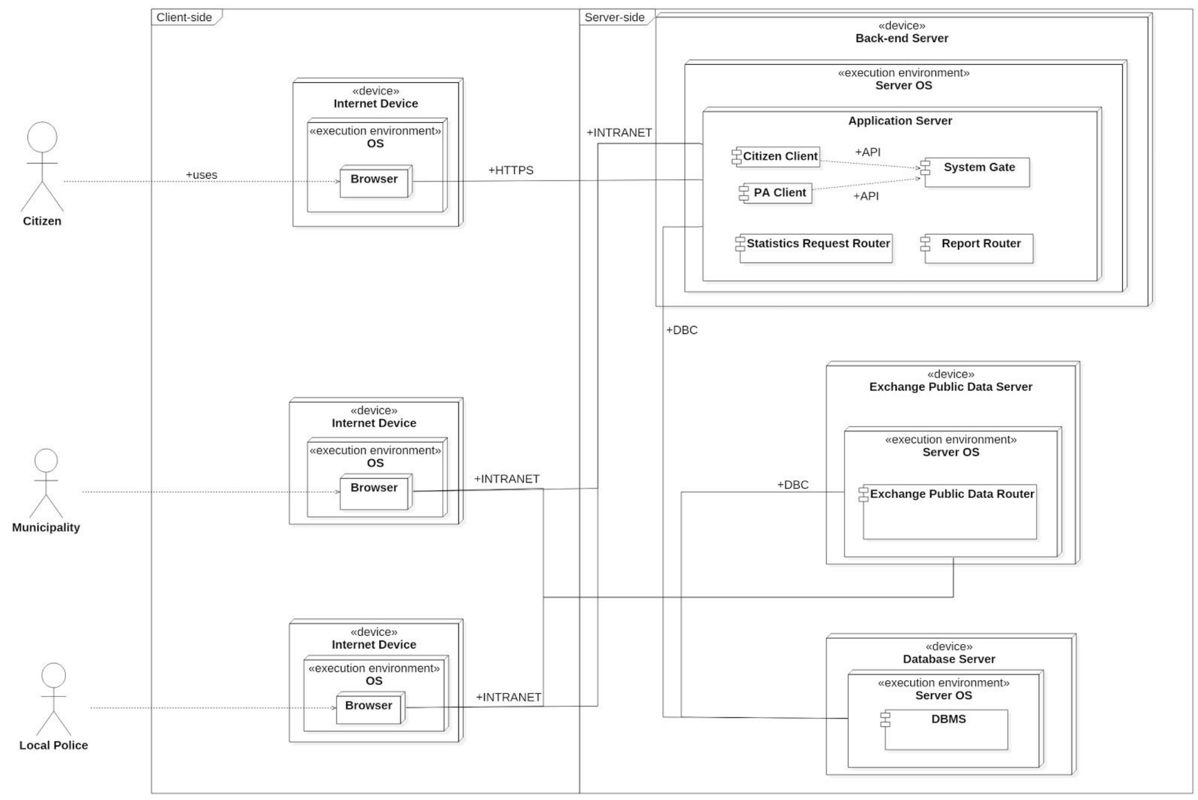

Fig.7 Deployment Diagram

2.4 Runtime view

For the next diagrams, the user will be assumed to be logged.

2.4.1 Observer Sequence Diagrams

In order to show the way the Observer pattern is exploited, three sequence diagrams are here reported. Fig.8 and Fig.9 concern the two processes of notification the system has to carry out, while Fig.10 describe the process of constantly up to date statistics provision the system has to accomplish in all its main functionality.

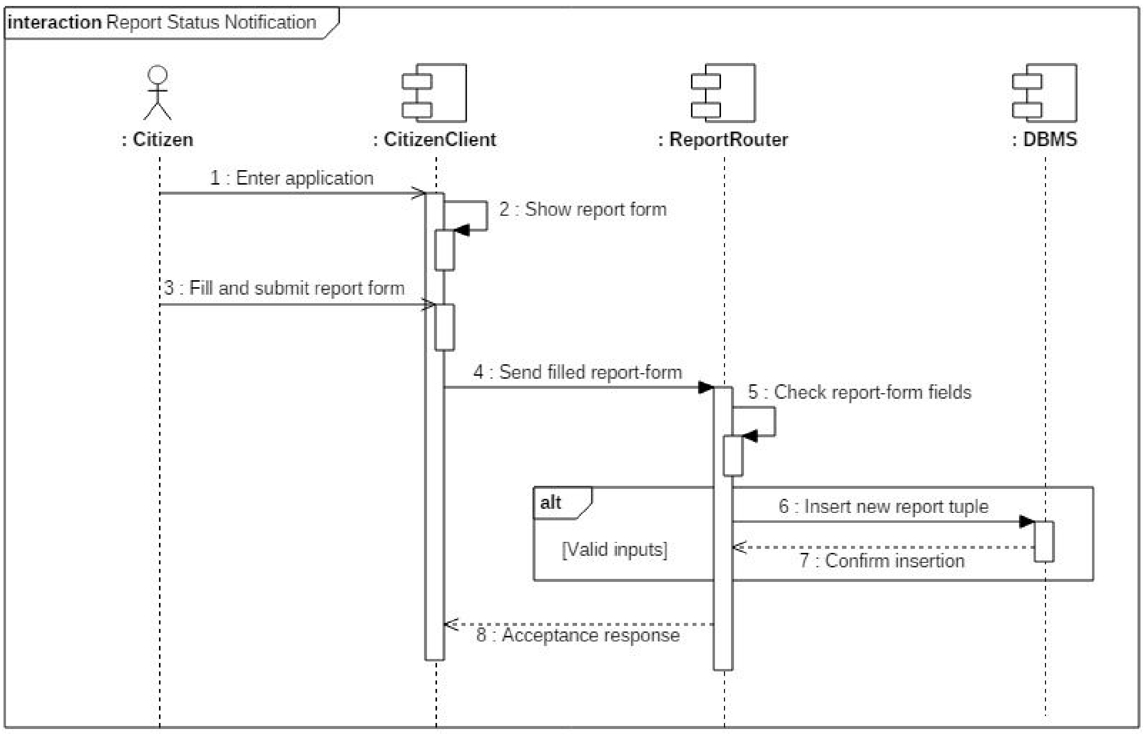
interaction Report Status Notification
: Citizen
: CitizenClient
: ReportRouter
: DBMS
1 : Enter application
2 : Show report form
3 : Fill and submit report form
4 : Send filled report-form
5 : Check report-form fields
alt
[Valid inputs]
6 : Insert new report tuple
7 : Confirm insertion
8 : Acceptance response

Fig.8 Report Status Notification Diagram

The sequence diagram represented in Fig.8 concerns the upload and the acceptance procedure of a report performed by a citizen. In this instance, the Observer is the citizen, while the Subject is the state or the report, that, after the checks carried out by SafeStreets will be either accepted or refused.

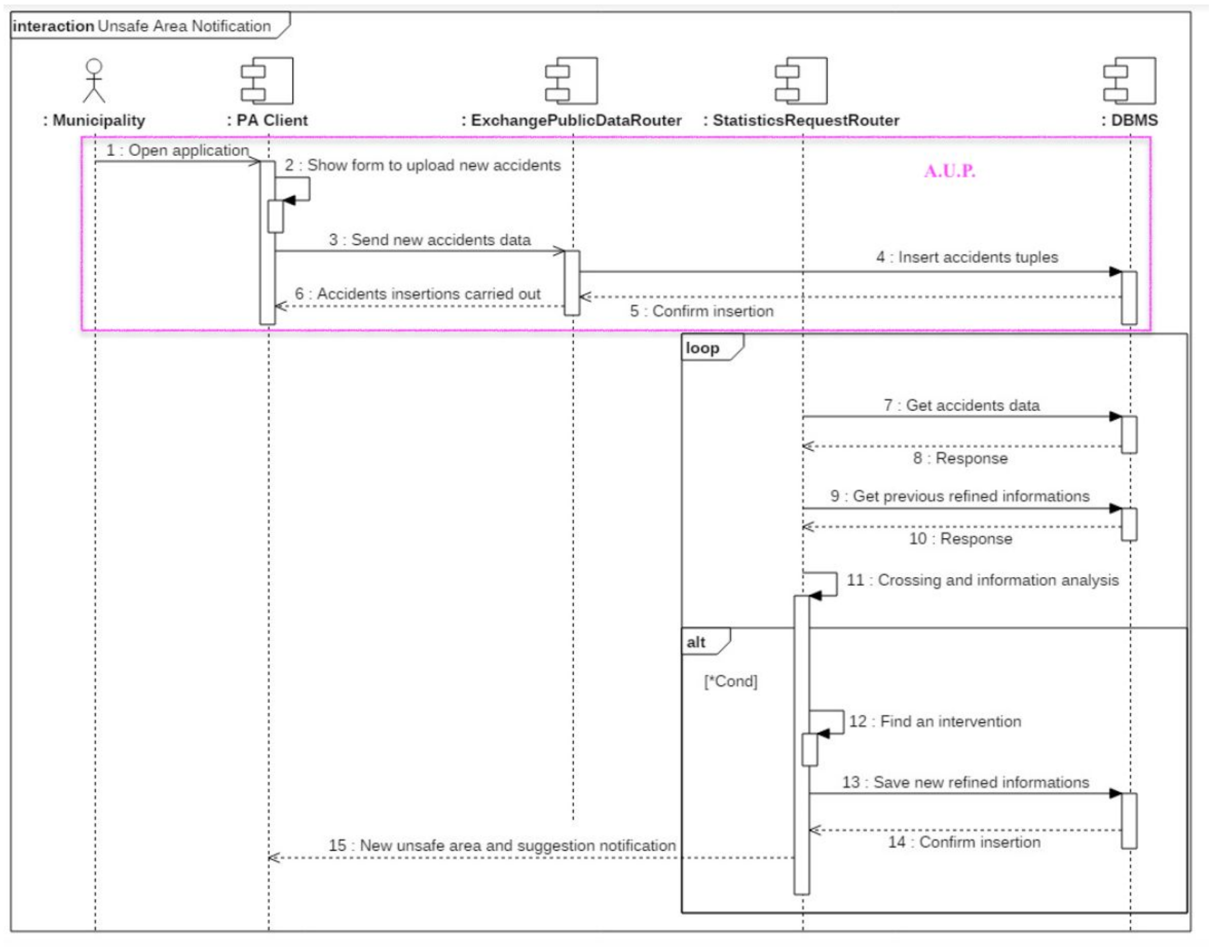
interaction Unsafe Area Notification
: Municipality
: PA Client
: ExchangePublicDataRouter
: StatisticsRequestRouter
: DBMS
A.U.P.
1 : Open application
2 : Show form to upload new accidents
3 : Send new accidents data
4 : Insert accidents tuples
5 : Confirm insertion
6 : Accidents insertions carried out
loop
7 : Get accidents data
8 : Response
9 : Get previous refined informations
10 : Response
11 : Crossing and information analysis
alt
[*Cond]
12 : Find an intervention
13 : Save new refined informations
14 : Confirm insertion
15 : New unsafe area and suggestion notification

Fig.9 Unsafe Areas Notification Diagram

The sequence diagram represented in Fig.9 shows the interaction between the Municipality, through the use of an automated processes, that uploads data concerning accidents occurred in its territory and the consequent notification of a newly identified unsafe. In this instance, the Observer is the Municipality, while the Subjects are newly identified unsafe areas that are pinpointed by the system.

*Cond1 → If is identified a new unsafe area then execute an algorithm to find out an intervention and notify the PA Client, after saving the new refined informations into the Database.

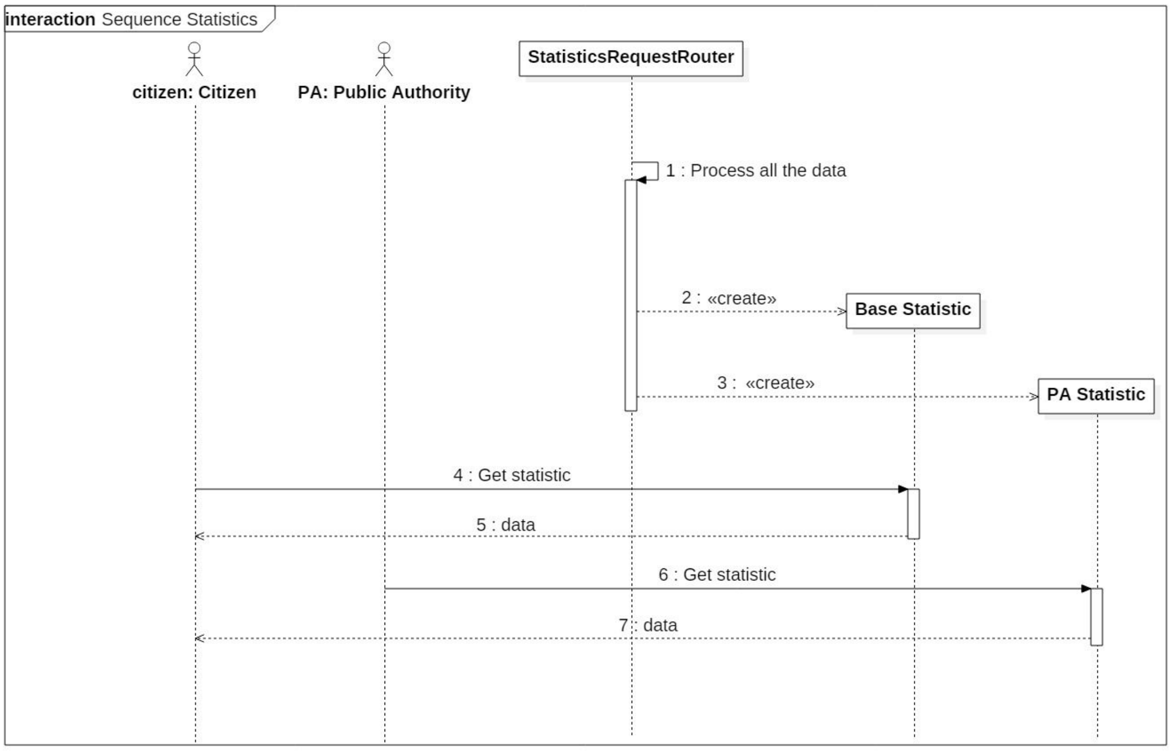

Fig.10 Statistics Creation Diagram

The sequence diagram represented in Fig.10 concerns the production of updated statistics that SafeStreets must be able to provide to all its the different users. Now the Subjects are the Statistics, while the Observer can be one of the two different users of the system previously identified in the R.A.S.D.
To represent the utilization of the Observer pattern with a proper level of abstraction it has been chosen not to consider specific instances, where each one of them would have considered only one user, but a more general process in which either a Citizen or PA could be substituted.
It is worth to notice that Police office is a public authority. Hence in *PA Statistic* is included also all the indexes about the effectiveness of SafeStreets service over the city.

2.4.2 Asynchronous Account Creation/Activation Sequence Diagram

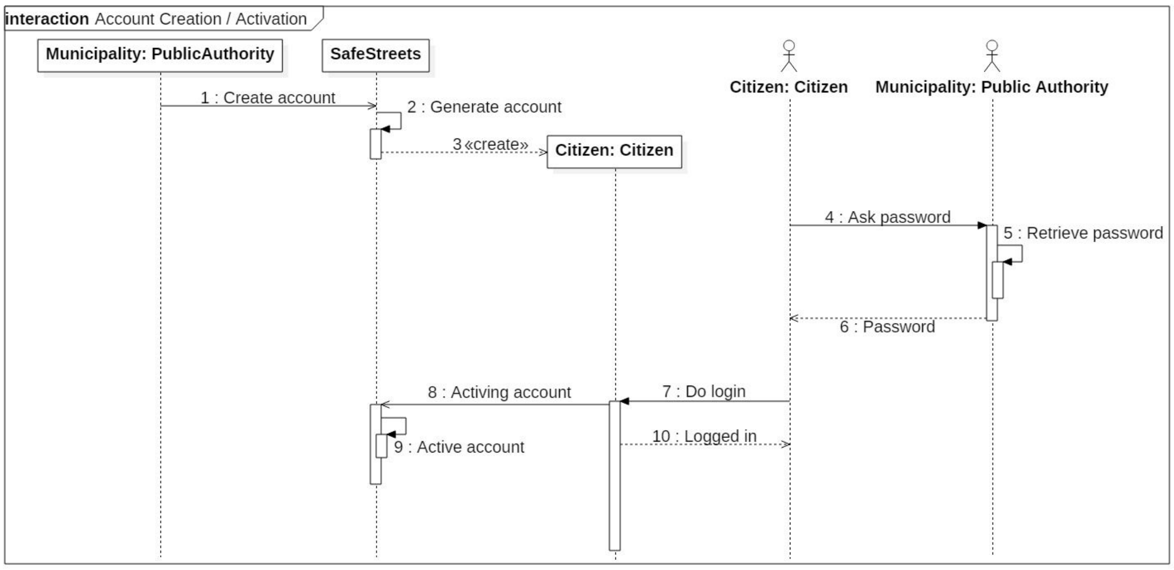

Fig.11 Account Creation / Activation Sequence Diagram

In the sequence diagram below, Fig.11 an important interaction between the municipality and SafeStreets is represented. This important interaction is the exchange of data concerning Citizens that become adults and for which an account must be created, and consequent request by an adults Citizen that wants to activate his/her account that the municipality has to forward to SafeStreets.

2.4.3 Validation Procedure Sequence Diagram

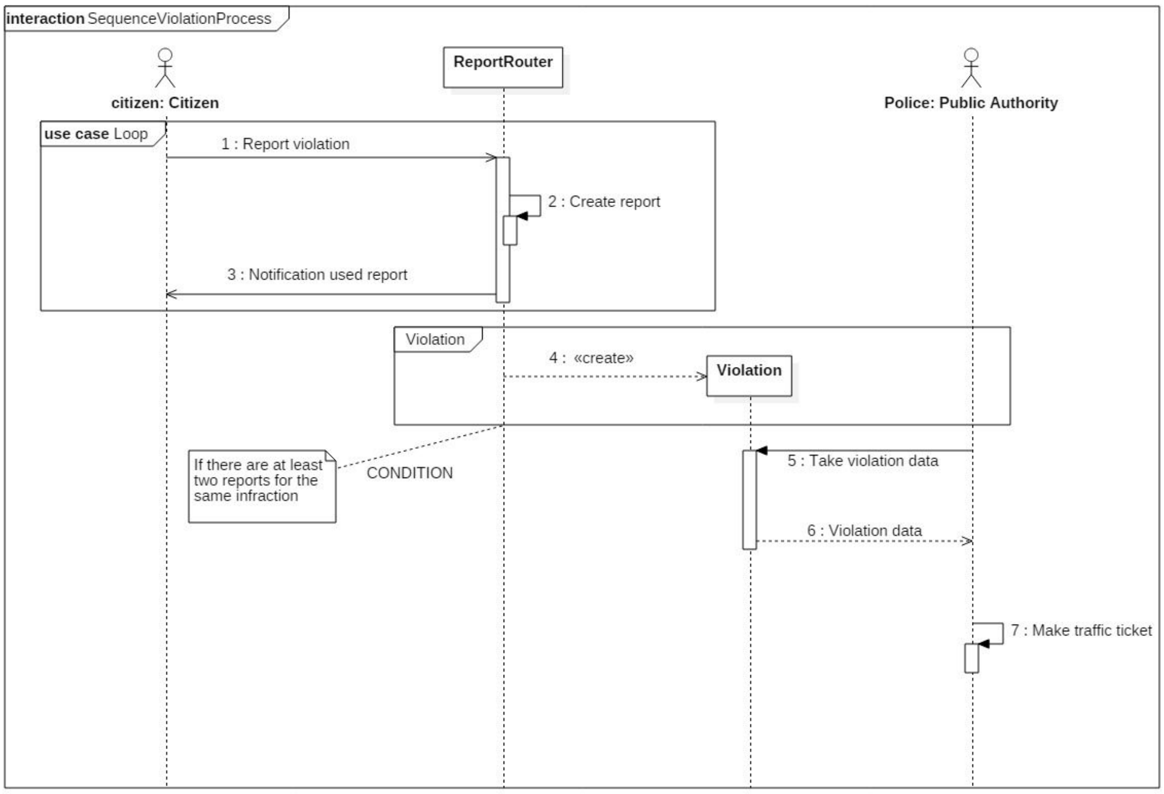

Fig.12 Violation Process Diagram

Considering the importance of the use that will be done by Local Police of the data provided by Citizens, a strict validation procedure has been put in place to guarantee the correctness of the data upon which authorities will perform their action, as it has been described in the R.A.S.D.
The sequence diagram reported in Fig.8 represent the interactions that take place during the validation procedure.

2.5 Component Interfaces

The interfaces presented below are the one used to link the components introduced in section 2.2 Component View.

2.5.1 Manage Data

This interface is provided to applications for allowing them to interact with the D.B.M.S. by offering a database connection to them.

2.5.2 Retrieve PA

Two interfaces for retrieving data by client-side components are designed in order to guarantee different levels of accessibility to data. The interface “Retrieve”, used for citizens, is detailed below and RetrievePA allows Municipality and Local Police to access:

- Statistics computed only for the Municipality itself and the ones containing information about unsafe area and possible solutions.
- TI-Statistics.

The different kinds of Statistics available to Municipality and Local Police have been deeply detailed in section 1.3.1. Definitions of the R.A.S.D.

2.5.3 Retrieve

This interface is aimed for Citizens in order to allow them to visualize Statistics SafeStreets makes available for them. The different kinds of Statistics available to Citizens have been deeply detailed in section 1.3.1. Definitions of the R.A.S.D.

2.5.4 Publish

To allow the exchange, through A.U.P. , of information, coming from client-side, between Authorities and SafeStreets the Publish interface has been designed. It is involved both in the processes through which:

- The Municipality provides data concerning incidents occurred in its territory.
- The Municipality communicates SafeStreets the creation\activation will of a Citizen.
- The Police provides data concerning traffic ticket issued.

2.5.5 Create Report

This interface provides an online service for Citizens with active account for sending data to the system.

The main purpose is the creation and the submission of reports, with helping mechanism for License Plate recognition.
The mechanism for the License Plate recognition, with the fulfillment of the optional field that allow Citizens to manually insert the license plate has been previously described in the R.A.S.D.

2.5.6 System Gate

The SystemGate interface exchanges credential data, adopting a secure protocol, with the DBMS through the SystemGate component in order to log each actor inside the system and grant them the access to SafeStreets resources based on their role. In particular it must:

- Allow Citizens to perform the login
- Allow Municipality to access the platform
- Allow Local police to access the platform

2.6 Selected Architectural Styles and Patterns

2.6.1 Model View Controller

Considering the process of visualization of different informations available to the different kinds of users of the system, it has been chosen to implement the Model-View-Controller pattern to handle in an efficient way the task of visualizing the data elaborated by Safestreet.
Plus, its structure makes it coherent with the logical division of the system. This pattern, employed also to separate internal representations of information from the ways information is presented to and accepted from the user, has been chosen to decouples major components allowing for code reuse and parallel development. As it is possible to see from Fig.13, the pattern divides the application in three distinct elements:

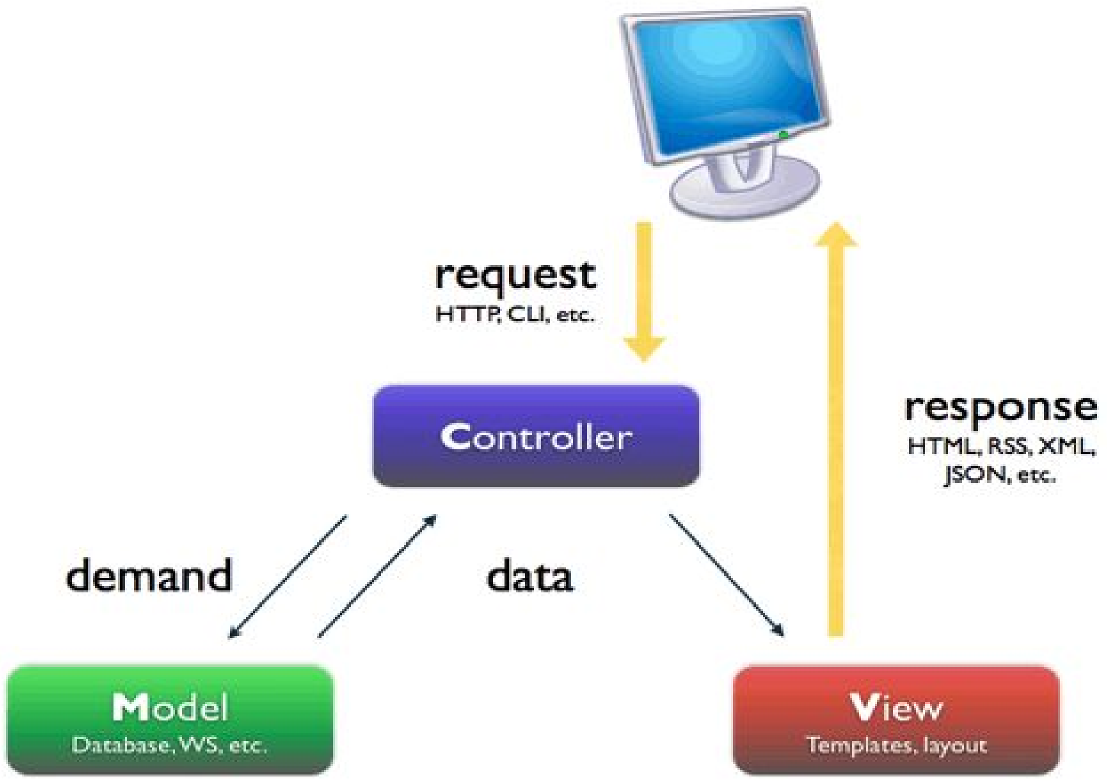

Fig.13 Model View Controller

- **Model:** It encapsulates the state of the application and defines the data and the operations that can be performed on them. It defines also the rules for the interaction with the data, exposing the access and update functions to the View and the Controller respectively. The **Model** is also responsible for notifying the components of the View of any *updates* that may have occurred following requests from the Controller, in order to allow the Views to present the users with updated data.
- **View:** The data presentation logic is managed only by the View. This implies that it must fundamentally manage the construction of the G.U.I. which represents the means by which users will interact with the system. Each G.U.I. can consist of different screens that have more ways to interact with the application data. To ensure that the data presented is always up to date it is possible to adopt the strategy known as "push model". The push model adopts the Observer pattern, recording the Views as observers of the Model. Views can then request updates to the Model in real time thanks to the latter's notification.
- **Controller:** This component has the responsibility to transform the user interactions of the View into actions performed by the Model. But the **Controller** is not a simple "bridge" between View and Model. By mapping the inputs and processes performed by the model and selecting the required View screen, the Controller implements the control logic.

2.6.2 Observer Pattern

The Observer Pattern models the relationship between one or more Observer who are interested in receive and visualize informations about a certain Subject.

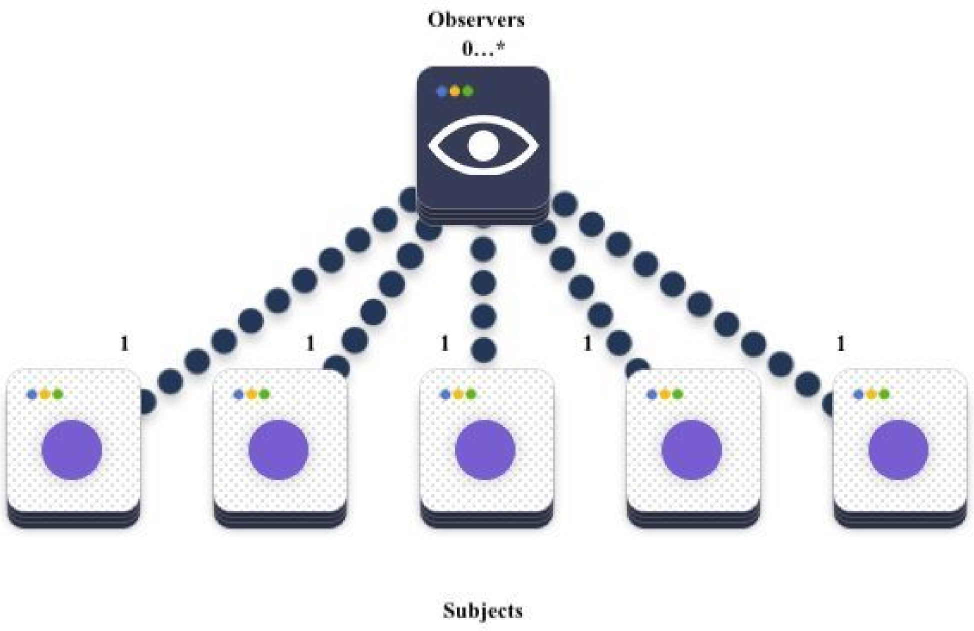

Fig.14 Observer Pattern Overview

In the project the Observer Pattern cover a double role. First, by regulating the relationship between the View and the Model, it ensures that all the data that have to be displayed to the different kinds of users of SafeStreets are up to date. Moreover, by virtue of its structure, it allows also to coordinate the process of notification of users. It must be considered as a pattern that will be used to implement partially the actions performed by the ReportRouter and fully the one performed by the StatisticsRequestRouter. The latter logically include the View-Model relation mentioned before.

2.6.2.1 Comparison with similar design pattern

Other similar design patterns have been considered to model these interactions; a comparison with a design pattern very similar to the Observer is reported below:

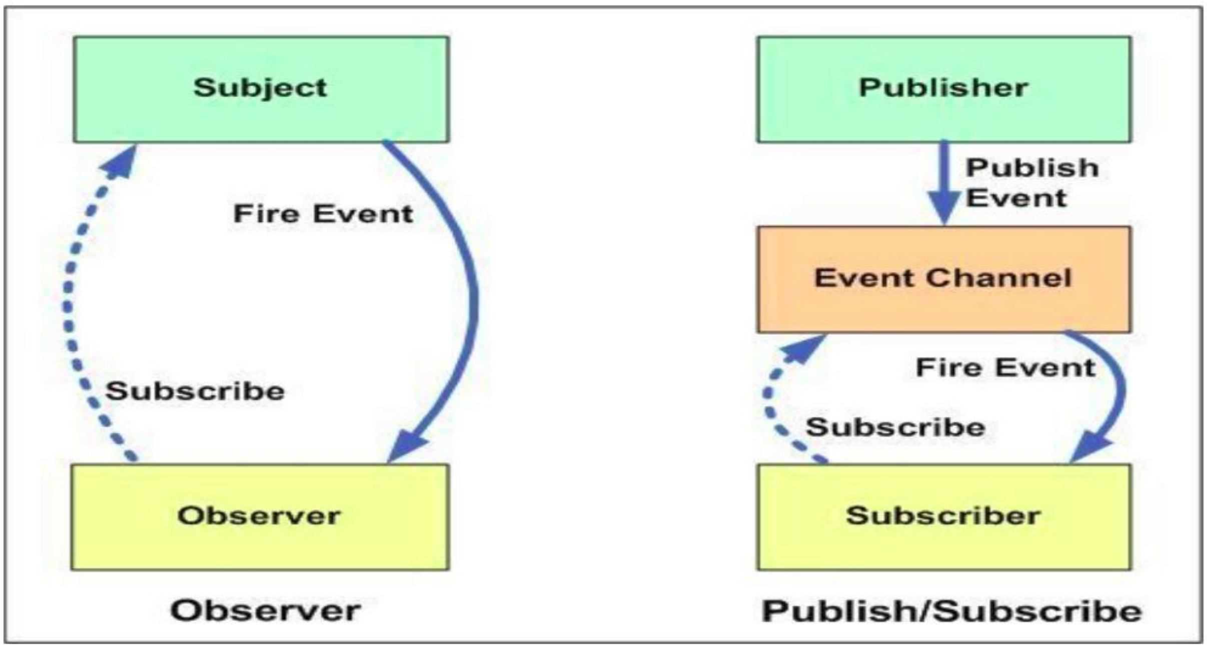

Fig.15 Observer and Publish/Subscribe Patterns

Considering the two design patterns, in the Observer pattern, the Observers are aware of the Subject, also the Subject maintains a record of the Observers. Whereas, in Publisher/Subscriber, publishers and subscribers don't need to know each other. They simply communicate with the help of message queues or broker. Analyzing the Back-end Server tasks, given that each reporter is naturally associated with the reports he/she made, there is no need to introduce a broker that forwards messages. The same consideration, as to say the Model & View relation between, holds for the processes concerning the StatisticsServer.
So, the Observer pattern has been found more appropriate than Publish/Subscribe pattern in this project.

2.6.3 Three Tier Architecture

A three-tier architecture is a client-server architecture in which the domain logic, data access, data storage and user interface are developed and maintained as independent modules on separate platforms.
The layers adopted in this project are presentation (client side), domain logic and data storage layer(server side).

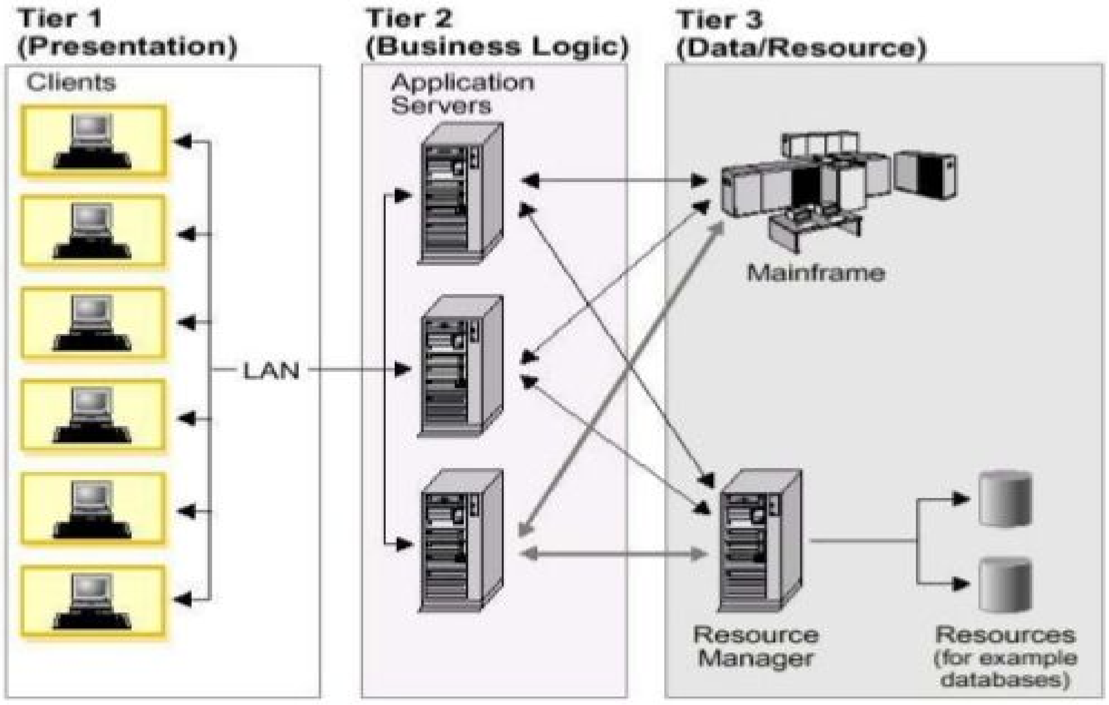

Fig.16 Model View Controller

This choice deserves a better explanation that regards the selection of the number of tiers and logical levels in which the system is divided. According to the state of the art in the multitier architecture science, a layer is a logical structuring mechanism for the elements that make up the software solution, while a tier is a physical structuring mechanism for the system infrastructure. For example, a three-layer solution could easily be deployed on a single tier, such as a personal workstation. Yet, in order to fully exploit the properties of modularity, such as scalability and resistance to obsolescence, it has been decided that a three tiered architecture along with the use of three different logical layers was the most suitable solution for the project.

2.6.4 Data warehouse

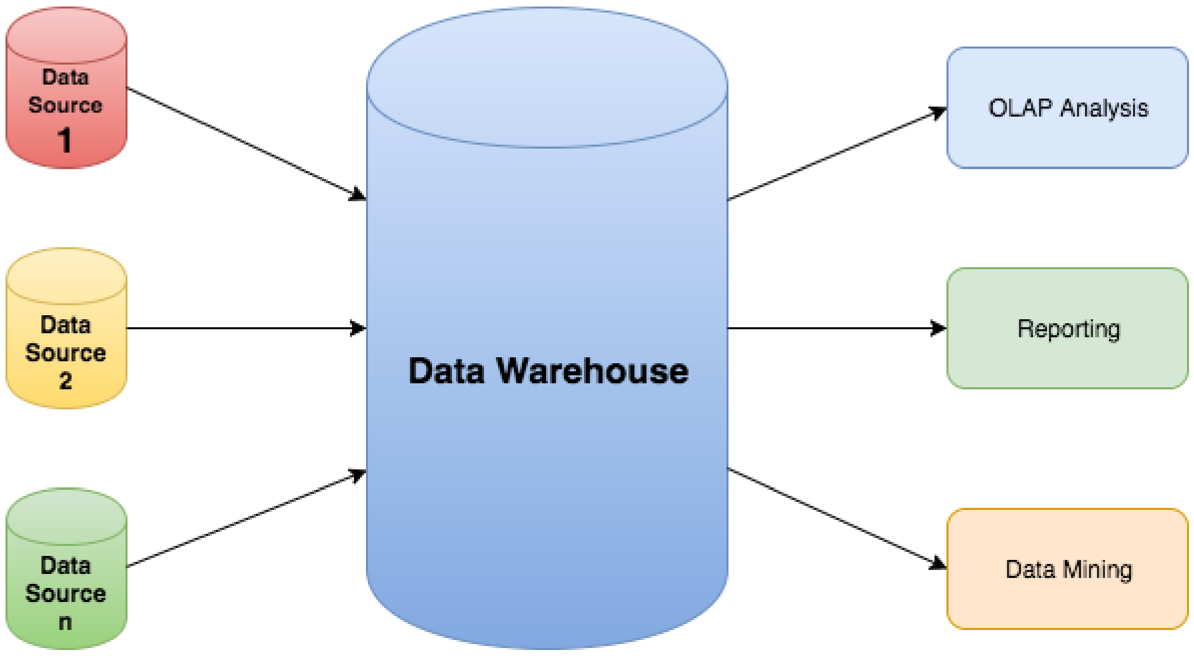

Fig.17 Data Warehouse

Along with the relational database that has been chosen in order to store reports and several others information such as citizens and violations data, a solution involving data warehousing has been adopted for the handling and the production of others kinds of data, as are the statistics computed by SafeStreets. This choice is coherent with the considerations performed in section 2.2.1, where it has been assumed that *"statistics are not assumed as fixed data structures, hence they're not inserted in the database."*
The table reported below explains and justifies in a detailed way why here it is very useful to implement this kind of solution, considered that several functionalities offered by Safestreets, specifically by StatisticsRequestRouter, involve the presentation of

indexes, statistics and decision making processes, as it has been explained in the R.A.S.D.

Characteristic	**Data warehouse**	**Transactional Database**
Suggested workload	Analysis, reporting, big data	Transaction processing
Data origin	Data collected and normalized from numerous sources	Data acquired as they are from a single source, as a transactional system
Data Acquisition	Mass writing of operations, normally in a predefined batch program	Optimized for continuous write operations based on the availability of new data to enhance transactional throughput
Data Normalization	Non-standardized schemes	Static schemes, highly normalized
Data Storage	Optimized for ease of access and high-speed query performance using column storage	Optimized for high-throughput write operations in a single-line physical block
Data Access	Optimized to minimize I / O operations and enhance data throughput	High volumes of small reading operations

Tab.1 Data Warehouse vs Transactional Database

The data warehouse will be carefully populated with refined data by a E.T.L. The latter is in charge of extract dirty data from the general database, transform and refine them depending on the analysis the system has to provide, and load them in the data warehouse. Eventual advanced analysis tools (such as hyper-cubes) can be built on it.

3. User Interface Design

The mock-ups for this project can be found in section 3.1.1 of the RASD and those produced during requirement analysis and specification phase are all the required ones.

4. Requirements Traceability

The mapping between the requirements that have been defined in the RASD and the design elements that have been defined in this document is performed below. The design elements will have to be developed to satisfy each respective requirement.

[G1] it allows citizens to report traffic violation with relative informations.

- [R1] it allows municipality to notify when a citizen becomes adult (account creation).
    - ★ ExchangePublicDataRouter, DBMS
- [R2] it allows municipality to notify when an adult wants to activate his account.
    - ★ ExchangePublicDataRouter, DBMS
- [R3] it supports the login of citizens.
    - ★ CitizenClient,SystemGate, DBMS
- [R4] it supports the access of the platform by the municipality.
    - ★ PAClient, SystemGate
- [R5] it provides a form to be filled with pictures of the violation and its data.
    - ★ ReportRouter, DBMS
- [R6] it provides a special field to be filled with the license plate of the vehicle committing the infraction.
    - ★ ReportRouter, DBMS
- [R7] it notifies the results of the acceptance procedure to the citizen who reported a violation.
    - ★ ReportRouter, DBMS

[G2] it allows both end users to mine informations.

- [R8] it provides charts and statistics for citizens.
    - ★ StatisticsRequestRouter, DBMS

- [R9] it provides charts and statistics for PA.
    - ★ StatisticsRequestRouter, DBMS

[G3] it provides suggestion for unsafe areas.

- [R10] it provides a search interface for PA in order to find unsafe areas.
    - ★ StatisticsRequestRouter, DBMS
- [R11] it allows municipality to upload data about the accidents that occurred in its territory.
    - ★ ExchangePublicDataRouter, DBMS
- [R12] it allows PA to get a possible solution for an unsafe area.
    - ★ StatisticsRequestRouter, DBMS
- [R13] it supports city safeness by notifying PA about a new unsafe area.
    - ★ StatisticsRequestRouter, PAClient, DBMS

[G4] it supplies local police with intelligence to generate traffic ticket.

- [R14] it supports the access of the platform by the local police.
    - ★ PAClient, SystemGate
- [R15] it provides to local police the records of infractions occurred on the territory of municipality.
    - ★ StatisticsRequestRouter, DBMS
- [R16] it allows local police to show an infraction in detail with all available data.
    - ★ StatisticsRequestRouter, DBMS

[G5] it provides TI-statistics in order to evaluate the effectiveness of the service.

- [R17] it allows local police to send feedbacks.
    - ★ ExchangePublicDataRouter, DBMS
- [R18] it allows municipality to obtain the self-evaluated informations.
    - ★ StatisticsRequestRouter, DBMS

5. Implementation, Integration and Test Plan

5.1 Implementation and integration plan

An incremental development is thought for this project in which most logic of the system is included in few interfaces. In order to define an implementation plan, an activity diagram is provided. First of all, three basic sub-components are needed to be developed in parallel: Report Router, System Gate and Exchange

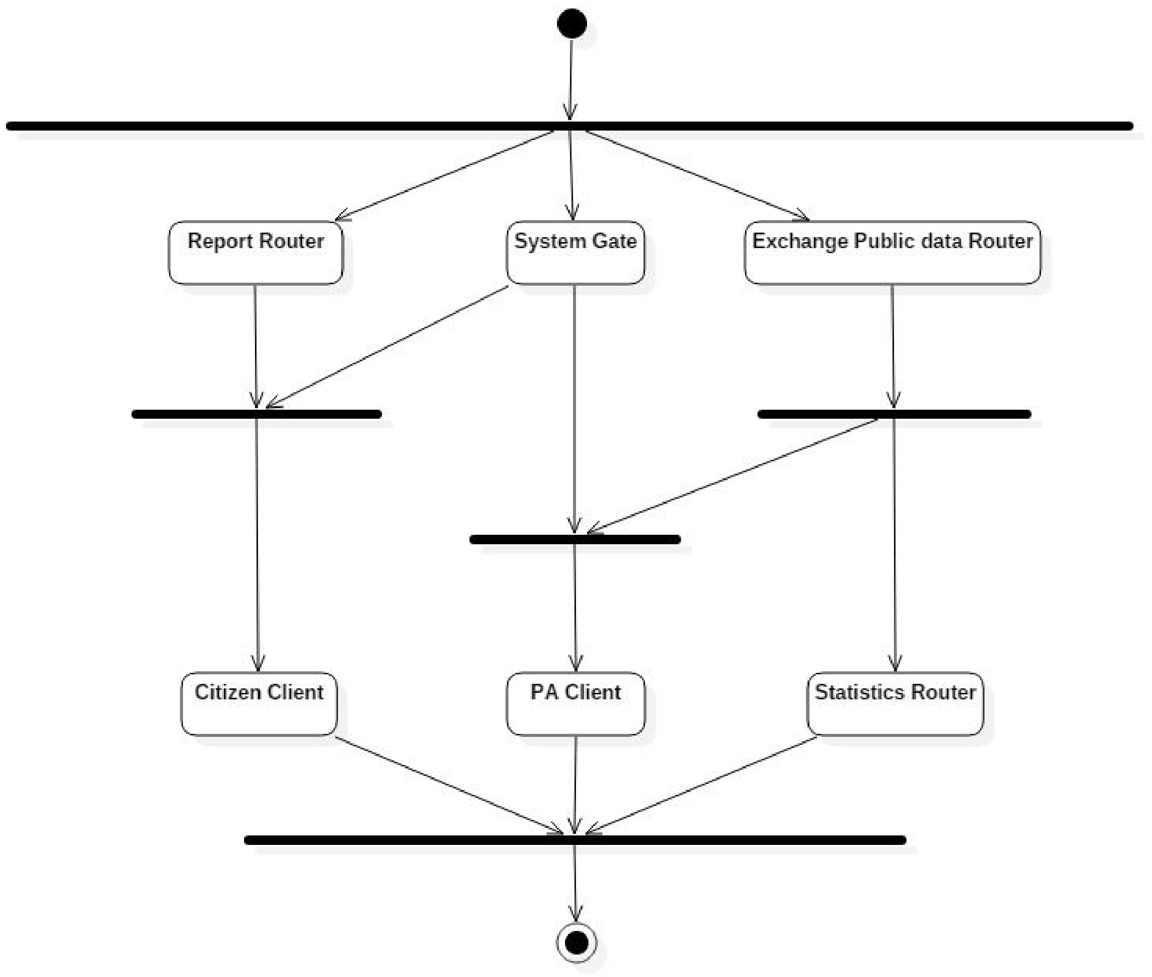

Fig. 18 SafeStreets Implementation & Integration Plan

Public Data System. These are the fundamentals of the whole SafeStreets system. Once they're built, the other sub-components, such as clients and Statistics Router, are implemented over them.

The component-dependency table follows in order to highlight the main structure of the system which the latest diagram is based on.

Component	**Dependencies**	**Interfaces Delivered**
SystemGate	manageData (implemented by the DBMS	API
ReportRouter	manageData (implemented by the DBMS	CreateReport
Exchange Public Data	manageData (implemented by the	Publish

System	DBMS	
StatisticsRouter	manageData (implemented by the DBMS	Retrieve, PARetrieve
CitizenClient	API interface	None
PAClient	API interface	None

Tab.2 Component-Dependencies-Interfaces Delivered

Worthy of note it’s the ReportRouter. It incorporates most logic of the system and handles the reports taking care deeply of the acceptance procedure and validation rules. A breakdown of this component is shown by the following detailed diagram regarding its implementation.

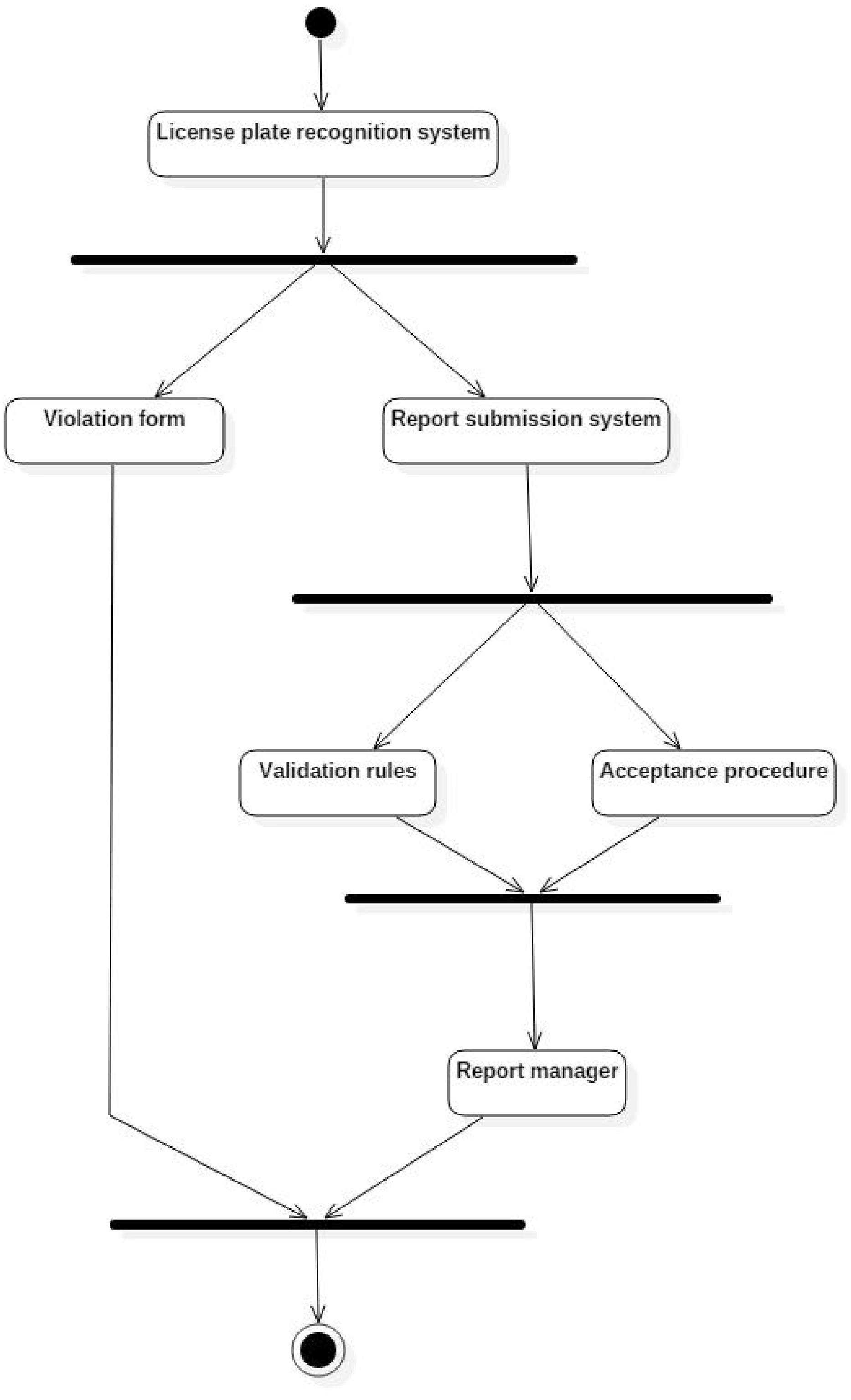

Fig. 19 ReportRouter Implementation & Integration Plan

A table that lists the main features provided by SafeStreets, their importance for the customer and the amount of effort needed for the implementation, is shown below for a better understanding of their relevance upon the decisions took during the project design.

SafeStreets takes care about important data. Hence it’s important to notice that no *low* importance is given to any feature. The scale of it will be then shifted within *medium*, *medium-high*, *high* priority.

Feature	**Importance for the customer**	**Difficulty of implementation**
Log in	*Medium*	*Medium*
Create reports	*High*	*High*
Acceptance procedure for reports	*Medium*	*Low*
Validate report	*High*	*High*
Provide statistics based on the user role	*Medium*	*Medium*
Identify unsafe area	*High*	*Medium*
Provide suggestions	*Medium - High*	*Low*
Exchange data with the municipality	*Medium*	*High*
Support Police office for traffic tickets issuing	*Medium - High*	*Low*

Tab.3 Importance & Difficulty of Features

5.2 Testing Plan

Functional testing examines how your hardware, software or Web site executes its requisite functions, and is vital for assuring the quality of released software. It includes testing of user commands, data manipulation, searches, business processes, user screens, and integrations. Validating an application or Web site in this manner ensures conformity to its specifications and the correct performance of all of its functions. During the development, as soon as the interfaces skeletons have been defined, it is strong recommended to prepare unit test, possibly including some test cases, for each component. All the other components should be then be developed based on these interfaces, checking the consistency at each step.

Giving the fact that the development is incremental and the server-side is first made, a top-down approach can be used to test all the dependencies. Particular attention should be given to ReportRouter. Unit tests with test cases should be prepared as soon as the

component is ready. The acceptance procedure and validation rules should be checked each time because it has a big impact on the service SafeStreets provides.
Performing functional testing at the user interface level is crucial as it can reveal many deficiencies not immediately apparent when conducting a source code review. First priority is given to testing the application's usability rather than the complexity of the application's internal workings.

6. Effort Spent

Here is reported the incremental effort spent by each member of the group on the project, measured in hours.

	Hours			
Date	**Amaranto**	**De Cillis**	**Gumus**	**Subject**
21/11/2019	5	5	5	Component Diagram
22/11/2019	7	8	6:30	Architecture Overview
23/11/2019	11	13	6:30	Applied Pattern
24/11/2019	12:30	13	9:30	General Review
25/11/2019	14	13	11	Sequence Diagrams
26/11/2019	16	19	15:30	Deployment Diagram
29/11/2019	19	19	15:30	Sequence Diagrams & Document Writing
30/11/2019	20:30	23	17:30	Document Review & Writing
1/12/2019	21:30	26	23:30	Component Interfaces
2/12/2019	27:00	31	29:00	Implementation & Testing
4/12/2019	34:30	36	35	Data Warehousing
5/12/2019	37:00	37:00	37:00	Document Review

7. References

- UML diagrams: https://www.uml-diagrams.org/
- UML diagrams: “UML DISTILLED” by Martin Fowler
- Multi-Tier Architecture: https://en.wikipedia.org/wiki/Multitier_architecture
- Observer Design Pattern: https://en.wikipedia.org/wiki/Observer_pattern
- ModelViewController Architectural Pattern: https://en.wikipedia.org/wiki/Observer_pattern
- E-R Model: https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model
- Data Warehouse:https://aws.amazon.com/it/data-warehouse/