POLITECNICO
MILANO 1863

SafeStreets

DESIGN DOCUMENT

Version 1 – 09/12/2019

Authors:

Giulio A. Abbo   10538950

Gianmarco Accordi   10587213

Massimiliano Bonetti   10560496

Professor:

Elisabetta Di Nitto

Academic year:

2019 – 2020

CONTENTS

1 INTRODUCTION

1.A PURPOSE

In this document the main design choices will be discussed, with a functional description of the system. Below the overview of the problem is reported, taken from the RASD.

1.A.1 Problem overview

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

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

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

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

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

1.A.2 Goals

These are the goals of the SafeStreets system:

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

1.B SCOPE

The world in which the system will work is modelled as follows: all the authorities that oversee the traffic conditions or can generate traffic tickets are considered as one for simplicity and are referred to as *municipality*. The municipality is not a mandatory actor, the system can work fine even without any. The user is a person that is subscribed to the system; his identity is verified, for this reason he is

---

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

considered trustworthy: he does not send false or wrong reports and SafeStreets will not check the correctness of the report. The users interact with the system mainly through a mobile device.

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

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

1.C DEFINITIONS, ACRONYMS, ABBREVIATIONS

1.C.1 Definitions
- (SafeStreets) System, Machine: the software to be and all its components
- User: the end user, a registered and logged in individual who can send reports and access statistics
- Municipality: the authority that oversees the road and traffic conditions in the area and generates tickets; to use the functions it must be registered and logged in
- (Traffic) Violations: all types of traffic violations punishable by law, e.g.: parking on bike lanes or reserved lots, double parking
- Report: an alert about a traffic violation
- Effectiveness of the system: the trend of the number of received reports in a given area with reference to the introduction of the SafeStreets' system.
- License Plate Recognizer or Recognition Plate System: is the third part service that identifies the license plate from the user's photo of the report
- Maps Service: is the third part service that identifies the place and position of the report
- Identity Verifier: is the third part service that verifies the identity of the user
- Client: the final costumers, can be a user or a municipality
- User Flow: in a UI, a series of steps that the user can take to achieve a goal

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
- DB: Data Base
- DBMS: Data Base Management System

1.C.3 Abbreviations
- Gn: nth goal
- Dn: nth domain assumption
- Rn: nth requirement
- Un: nth use case

1.D REVISION HISTORY
- Version 1   First version of the document.

1.E REFERENCE DOCUMENTS

- Project assignment: "Mandatory Project Assignment AY 2019-2020"
- Standard IEEE 1016:2009
- OMG UML Revision 2
- SafeStreets RASD

1.F DOCUMENT STRUCTURE

This document is comprised of seven chapters.

The first chapter offers an overview on the problem, with the goals, definitions, acronyms and abbreviations used in this document. For more details refer to the *SafeStreets RASD*.

The second chapter offers a detailed view of the architectural design. Contains the components in which the system is divided and their interfaces, the main classes that are used to model the information, some details on the interaction between components through sequence diagrams, a description of the deployment and the architectural styles and patterns used.

The third chapter expands what already stated in the RASD about the user interfaces, providing user flow diagrams.

Chapter four explains the connections between the requirements and the design elements of the system.

In chapter five are reported the plans for integration, implementation and testing of the System, with details on the priorities of the various components and on the ordering of their implementation.

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 six.

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

2 ARCHITECTURAL DESIGN

2.A OVERVIEW

The high-level architecture of the SafeStreets' system is highlighted in Figure 1 below.
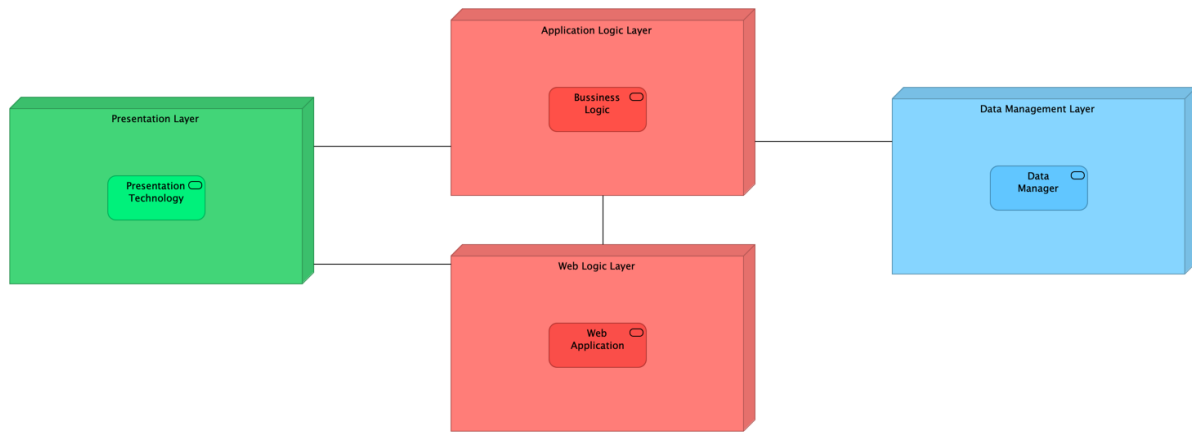

Figure 1 – High-level components and their interaction

The defined architecture is a four-tier architecture in which each component is briefly described below:

- *Presentation Layer*: is the part that manages the visualization of the data and the possible interaction with the system in a human readable and user friendly way, this part knows how to show to the final client the results that come from the *Application* or *Web layer*. The user or municipality interacts with this layer in order to interact with the system by using the application or the web app through a browser.
- *Application Logic Layer*: implements all the business logic of the SafeStreets' system, it receives all the requests from the application on the client's device, and also the request coming from the web, then it elaborates them, by retrieving all the necessary information from the *Data manager*. It takes care of managing the data, by integrating it with the data provided by the municipality.
- *Data Management Layer:* has the task of managing the physical allocation of the data, to answer to the queries that come from the application layer, and to store the data that the application layer wants to memorize.
- *Web Logic Layer*: is used to answer to the web application requests made by the clients, but it doesn't implement any type of business logic, so it's only used to decouple the logic from the web visualization of the data. All the requests made by the clients through a web page are redirected to the application layer.

The different levels of abstraction allow to manage different functionalities offered by the system on different machines, that need only to implement the required interface, allowing the replication of the machines, in order to scale in case of necessity and to be fault tolerant. The user is unaware of the distribution of the levels: he only needs to communicate through a graphic interface.

The data used by the system is obtained directly through the user and from the municipality, so periodically the system will integrate the new data obtained by the municipality with the data already collected.

To allow the communication with services outside the SafeStreets' system the architecture makes use of adapters, this allows all the other component inside the system to use the same sets of operation, but the implementation of the adapter can change based on the implementation of the third parties.

The different levels of abstractions are decoupled as much as possible, this means that they communicate through well-defined interfaces, this allows the developers to extend some layer if necessary or to change them, for example to change the data manager it is sufficient to change the component on the Application Level.

2.B COMPONENT VIEW

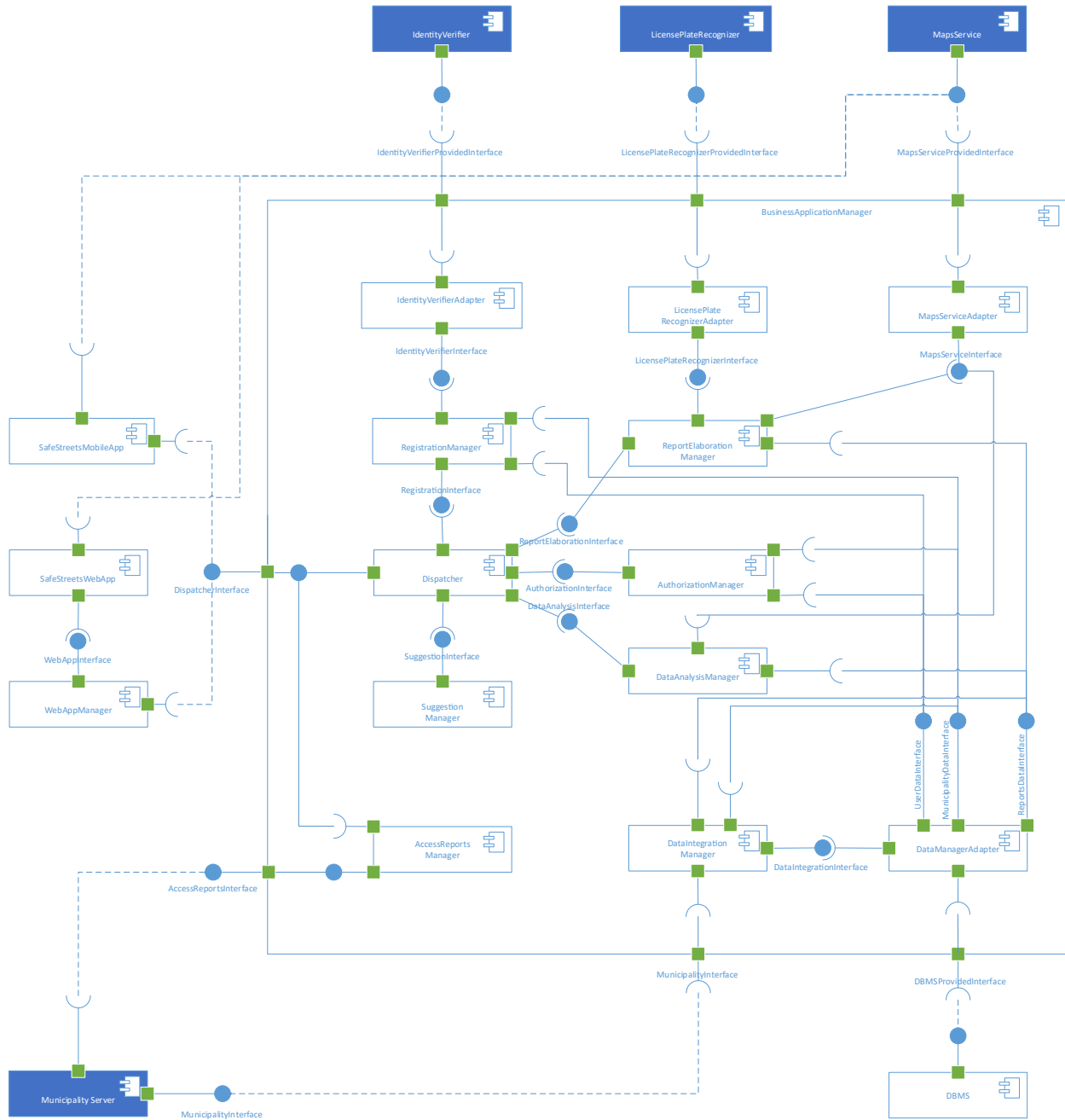SafeStreets' system is composed by a component structure defined in Figure 2 below.

Figure 2 – Component diagram

The figure illustrates how the components are linked together: as already stated in the high-level architecture, the modules communicate through well-defined interfaces that are explained in chapter 2.E.

The blue components indicate which components the system relies on, and that are not developed with the system but by third parties.

The following list define the sets of tasks carried out by each single component that is external to the *Business Application Manager*:

- **Maps Service**: this component provides to the system a geographical meaning to the space the system works with, the Maps Service is implemented by a third party, so an adapter will be used

to connect the Maps Service to the system, this service needs to provide an interface that allows our system to retrieve the address of the violations, and also to get an overlay map that we can modify in order to provide a better graphical representation to the client, and in general it helps the system to have a better understanding of the geographical space.

- **SafeStreets' Web App**: is another method that the client can use to interact with the SafeStreets' system, it has a similar style to the Mobile App, but it can be used in a browser. This will contact the Web App Manager, through the Web App Interface, and it will be provided the required page, then the component's requests will be forwarded to the business logic via the Web App Interface. In order to interpret the coordinates, it uses the methods exposed by the Maps Service Provided Interface.
- **Web App Manager**: takes care of all the requests coming from the Web App of the system, this means that it has to provide all the pages requested and due to the fact that it has no understanding of the business rules, it simply forwards the requests to the Business Application Manager.
- **Municipality Server**: is the component managed and deployed by the municipality. This component can expose the Municipality Interface to use the data integration services and can use the Access Reports Interface to use the Access Reports Service.
- **License Plate Recognizer**: as highlighted in the diagram, this component will not be implemented by the system, but the system will rely on a third party that offers this service. This carries out the task of recognizing the plate from the report's photos provided by the user. As already stated in the RASD, the response that we get from this component will be trusted by the system, so it will not be subject to other verification. In order to communicate with the component, the system implements an adapter.
- **Identity Verifier**: the system relies on this component, developed by a third party, that offers the service of verifying the identity of the user. When the user provides its credentials during the registration, the system uses the Identity Verifier Provided Interface in order to check the user's identity. Similarly, to the License Plate Recognizer, we trust the results obtained through this service.
- **DBMS**: this component is responsible for the physical allocation and management of all the data used by the system, it exposes the DBMS Provided Interface, that will be used by the system to memorize and query the required data. The exposed interface will be wrapped by an adapter, in order to easily change the implementation of the technology.

Here follows a closer look to the subcomponents that implement the *Business Application Manager*, that provides the business logic of the system:
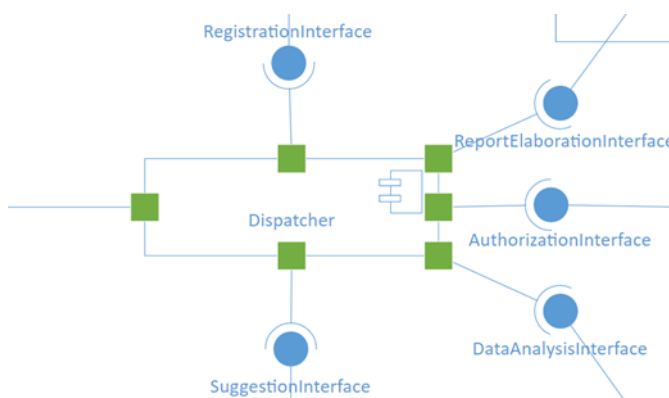

Figure 3 – Interfaces used by the Dispatcher

- **Dispatcher**: the figure makes it clear that the dispatcher is used as an ingress point for the requests coming from outside the system, the component will implement the Dispatcher Interface. The dispatcher redirects the requests to the other components. As in Figure 3, the Dispatcher uses the interfaces provided by the Suggestions Manager to complete the suggestio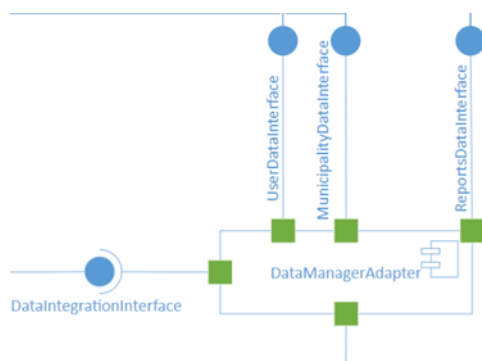n request made by the Municipality, by the Registration Manager to complete the registration requests, by the Report Elaboration Manager that will take

care of elaborate the report sent by the user, by the Data Analysis Manager to retrieve the analysis of the data, by the Authorization Manager that will verify the credentials of the clients and his access rights (this operation is repeated for every request made by the user): the AccessType returned by the Authorization Manager indicates which operations the requester can carry out.

- **Registration Manager**: this component will get from the Dispatcher the registration request of a user: it verifies the user's identity through the service of the Identity Verifier, and if everything is fine it memorizes the user in the system by using the User Data Interface. When it gets the registration request for a municipality, that contains the contract code of the registration, it verifies it and then it memorizes the municipality via the Municipality Data Interface.

- **Identity Verifier Adapter**: as already mentioned, the access to the Identity Verifier Provided Interface is not direct, but through this adapter, that expose the Identify Verifier Interface, of which the implementation is used as a wrapper of the real interface, thus allowing the system to be decoupled from the implementation of the Identity Verifier.

- **License Plate Recognizer Adapter**: similarly to the previous component, this is an adapter, that exposes the License Plate Recognizer Interface, that can be used by all internal components to retrieve the license plate from the photo provided by the user's report.

- **Maps Service Adapter**: also this component is used to expose an interface, the Maps Service Interface, that is used as a wrapper for the real interface Maps Service Provided Interface exposed by the Maps Service. This component is used by the internal components to understand the geographical coordinates and to manipulate the geographical data.

- **Report Elaboration Manager**: it receives the report sent by the user through the Dispatcher, it implements the Report Elaboration Interface. When a report is received, if necessary, it validates the position of the violation and recognizes the license plate present in the report's photo. The component has the task to notify the user if something goes wrong during the elaboration, if there are no problems, the component uses the Report Data Interface to memorize the data.

- **Authorization Manager**: this component is responsible of verifying the credentials provided by the user or the municipality during the login and of setting the correct access rights for the municipality or the user that will be used during its navigation in the system. It gets the login request from the Dispatcher through the Authorization Interface that this component implements, then it accesses the user or municipality data by using respectively the User Data Interface or the Municipality Data Interface. This component will be used by the Dispatcher every time a request arrives in order to verify which actions the requester can perform using the AccessType returned by this component, in all the internal components is implicit that the access rights of the requester are verified.

- **Data Analysis Manager**: has the task to carry out the data analysis requested by the user or by the municipality. When the request comes from the Dispatcher, the component starts to aggregate the data that he can get through the Report Data Interface exposed by the Data Manager Adapter, when the result is available it is sent back to the user or the municipality. The component also uses the Maps Service, through the Maps Service Interface. This component can receive the requests from the municipality to get all the latest reports made by the users, it will get them through the Data Manager Adapter, and it will send the latest reports to the Municipality.

- **Suggestion Manager**: this component identifies the possible intervention that a municipality can carry out in order to reduce the number of violations. When a municipality asks for some suggestions, the request is sent from the Dispatcher to this component, the component will

receive the identity of the municipality that asks for suggestions and the most common violations in its area, that has been previously requested by the Dispatcher to the Data Analysis Manager. Based on the information received, it finds out some action that can be taken, the suggestions found are sent back to the municipality.

- **Access Report Manager**: it is the component that exposes the method used by the municipality to access the violations reported to SafeStreets' system by the users. This component is used by the Municipality to automatically get the latest reports made by the users. The incoming requests will be forwarded to the Dispatcher that will then ask the Data Analysis Manager to return the latest reports. It is important to remark that the municipality can also ask for the latest reports from the mobile or web interface, by sending a request directly to the Dispatcher.
- **Data Integration Manager**: is the component responsible to maintain the data memorized by the system up to date with the municipalities that offer the data integration. It uses the method exposed by the Municipality Interface, implemented by the municipalities. The component periodically or on request asks for the latest violations, then through the method exposed by the Report Data Interface, it memorizes the latest violations in the DB.
- **Data Manager Adapter**: this component is used as an adapter for the DBMS Provided Interface. The interfaces exposed by this component are highlighted in the Figure 4: the component must implement the User Data Interface that allows to access the data of the user, similarly the Municipality Data Interface allows to access the data of the municipalities. The component also implements the Report Data Interface that allows to store and retrieve the reports memorized in the system. It also uses the Data Integration Interface that is the interface exposed by the Data Integration Manager and it is used to update the violations obtained from the Municipality.

Figure 4 – Interfaces of the DataManagerAdapter

2.C DEPLOYMENT VIEW

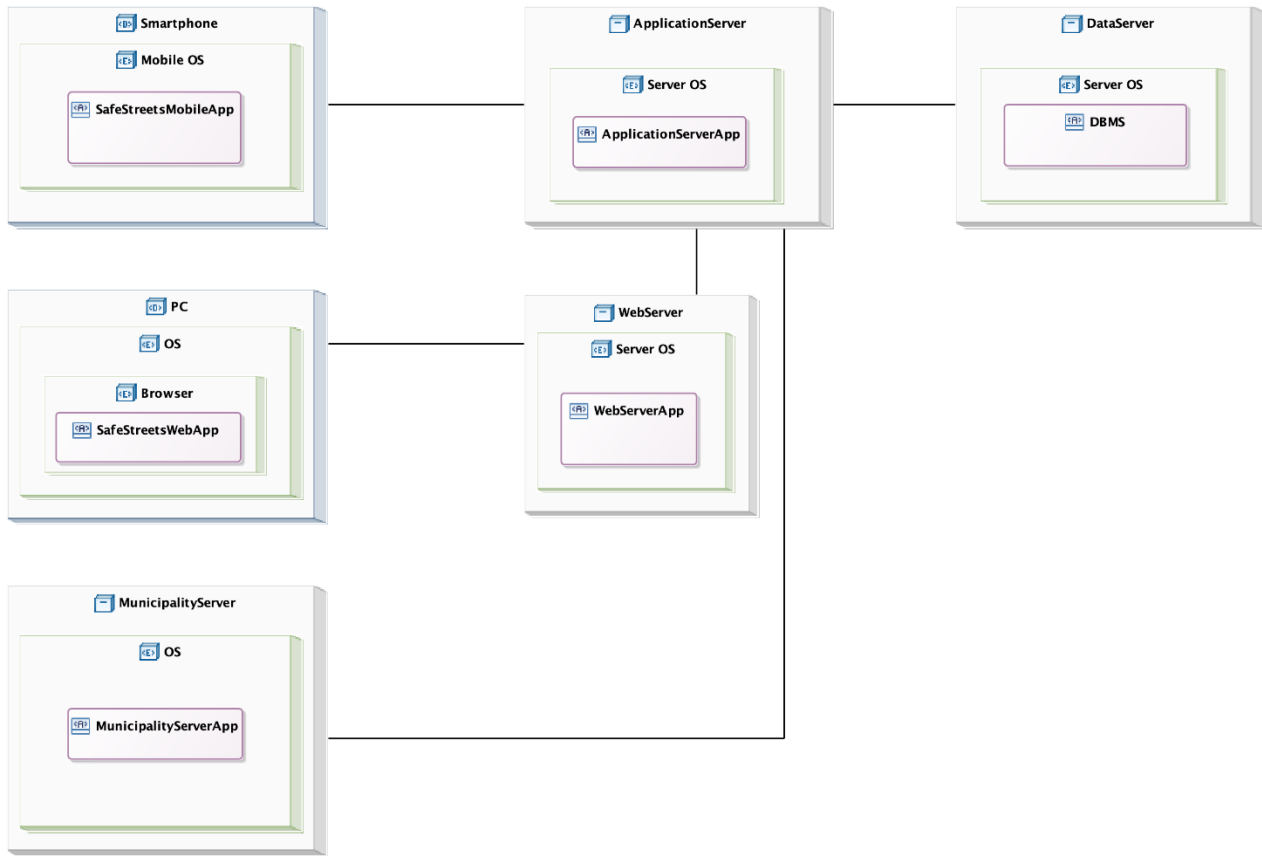The next picture shows how the system will be deployed on the various machines.

Figure 5 – Deployment diagram

The architecture is divided in four tiers, there are the machines of the user and of the municipality, the web server, the application server and the data server.

The presentation layer is split between the machines of the user and of the municipality and the web server. The application layer is on the application server and the data layer is on the data server.

The user and the municipality can use the SafeStreets' mobile app on the smartphone and the SafeStreets' web app on the browser of the personal computer. The municipality can also have a server in which are stored its reports and from which the application server, with the permission of the municipality, can retrieve the reports. The municipality's server can also retrieve from the application server the reports made by the user in its city.

The web server app is on the web server, which communicates with the SafeStreets' web app to provide the web pages and with the application server to forward the requests of the user or of the municipality.

On the application server there is the Application server app, it communicates with the web server, the data server to store and load the data, the SafeStreets' mobile app to answer the requests of the user and the municipality's server.

2.D RUNTIME VIEW

In this section will be better clarified the behaviour of the system at runtime, in terms of function calls, in order to specify the sequence of operations done to carry out a certain task.

The Dispatcher, as said before, acts as an orchestrator for the completion of all the tasks. In all the following sequence diagram, except for the login and the registration, is implicit that in order to use the services exposed by the system, the client must have performed a login operation.

In the following diagrams, the method-calls on the Web Server are made by the Web App, and they are an abstraction of the real exchange of message between the Web App and the Web Server, to keep the diagram simpler. In this case the name of the method is the same as the name of the method that will be called on the Dispatcher with the required parameters. To improve clarity, in some cases the returned value or an error are indicated, followed by the name that represents the object returned. If the returned arrow does not have anything it means that the method does not return anything and completed successfully.

User Login
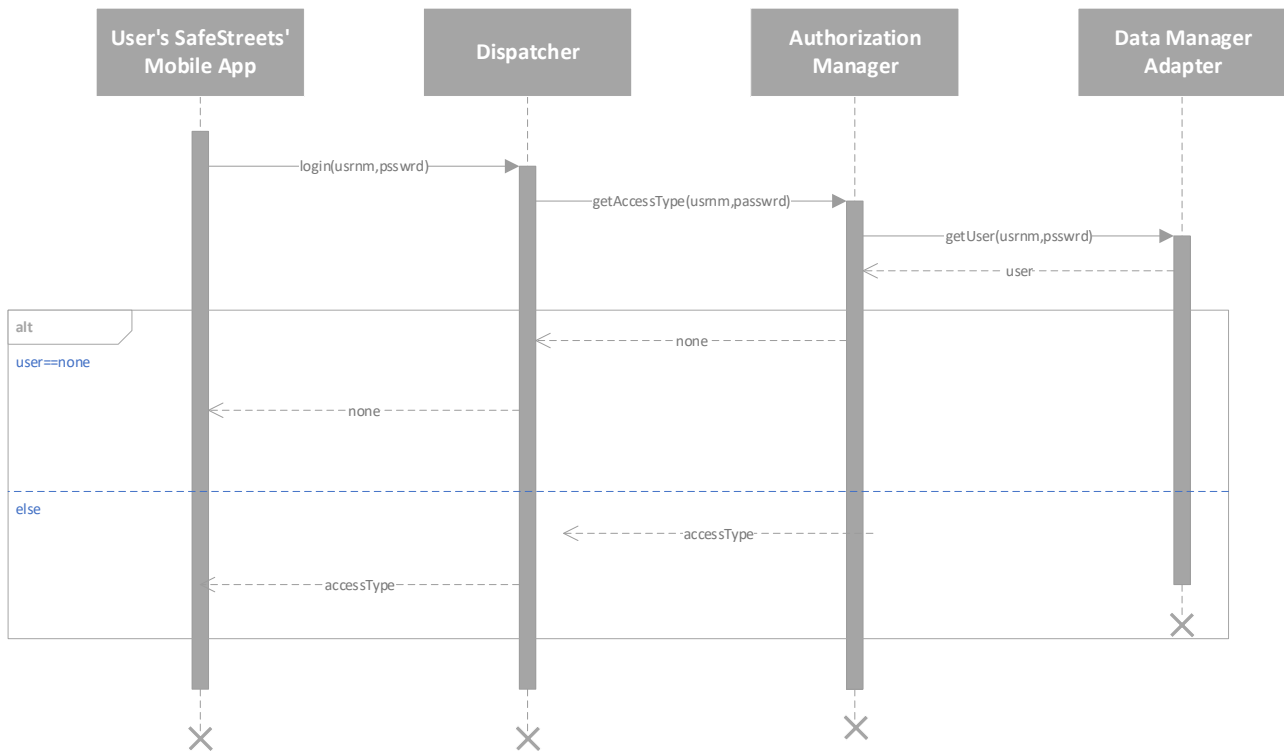
Figure 6 – Sequence diagram of the Login operation

This Sequence Diagram represents the orchestration of operations carried out in order to perform the login, either by the user or the municipality. The operation of login can be performed both from the mobile app and from the web app through a browser, the sequence of operation is the same, but in the first part we need to insert an intermediate Web Server that forwards the requests coming from the client to the Dispatcher. After the login operation the user is recognized by the system and all the services exposed by SafeStreets system can be accessed.
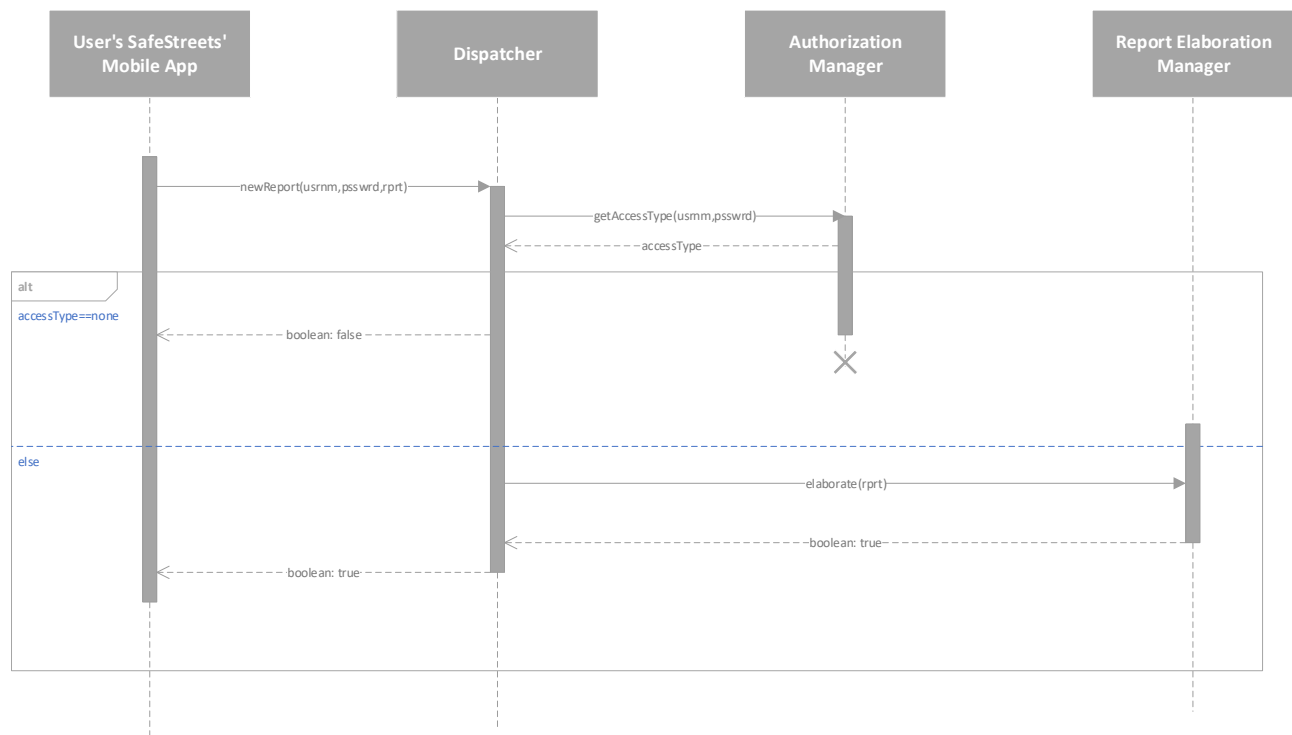
Submit Report

Figure 7 – Sequence diagram of the Submit report operation

This is the sequence of method invocations performed to submit a report violation: first of all the Dispatcher needs to verify the identity of the user, it contacts the Authorization Manager that returns an AccessType based on the credentials of the user. At the end of the tracks there is no cross because the computation goes on – as described in the below sequence diagram – in order to elaborate the report, the components are still alive.

Report Elaboration
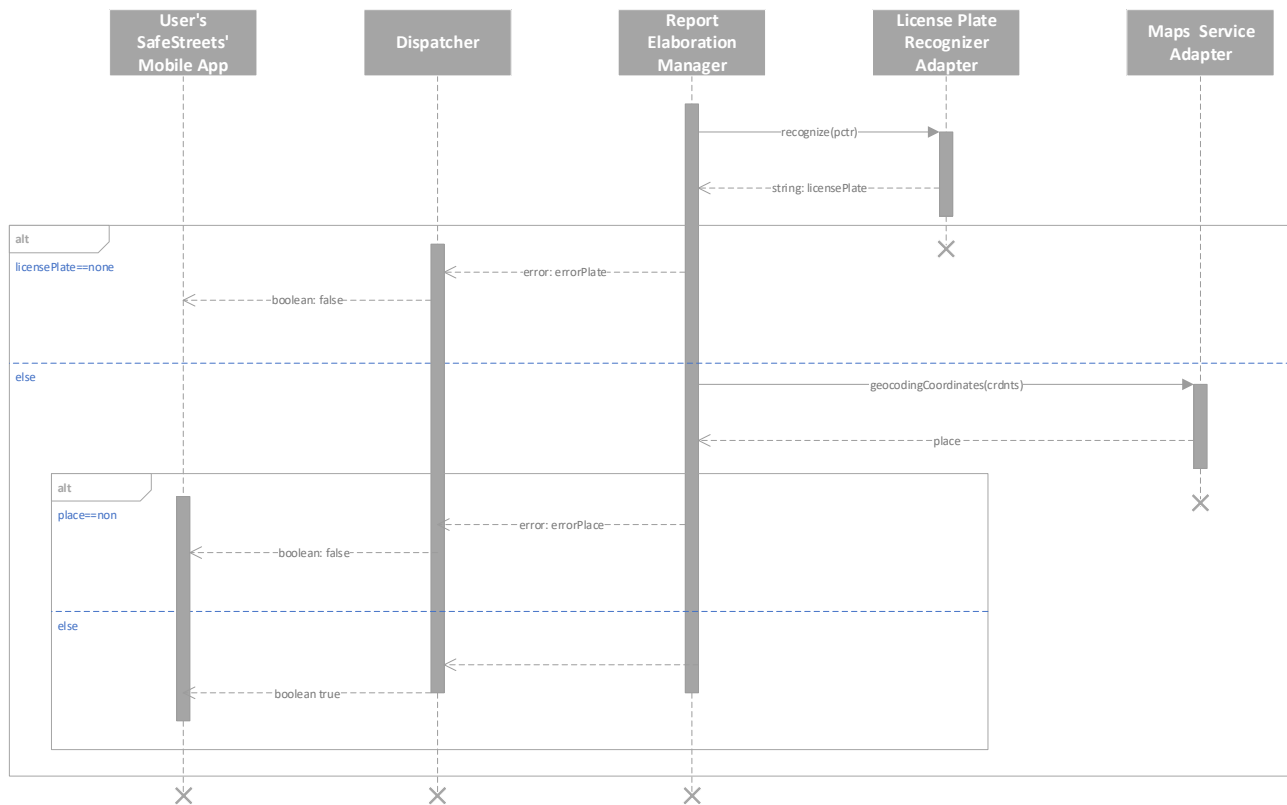
Figure 8 – Sequence diagram of the Report elaboration

This sequence of operations defines how the elaboration of a report is done by the system: after the system has received a report from a user, after what happened in the previous sequence diagram, it tries to complete the report by looking at the license plate and position of the violations In the second case the system will maintain both the coordinates and the address of the position returned by the Maps Service in the Place data structure. The system calls the geocodingCoordinates method if the user sends the coordinates to obtain the Place, otherwise if the system receives the address then it calls the method geocodingString. This is done to help the work of the Data Analysis Manager.

Get Available Statistics
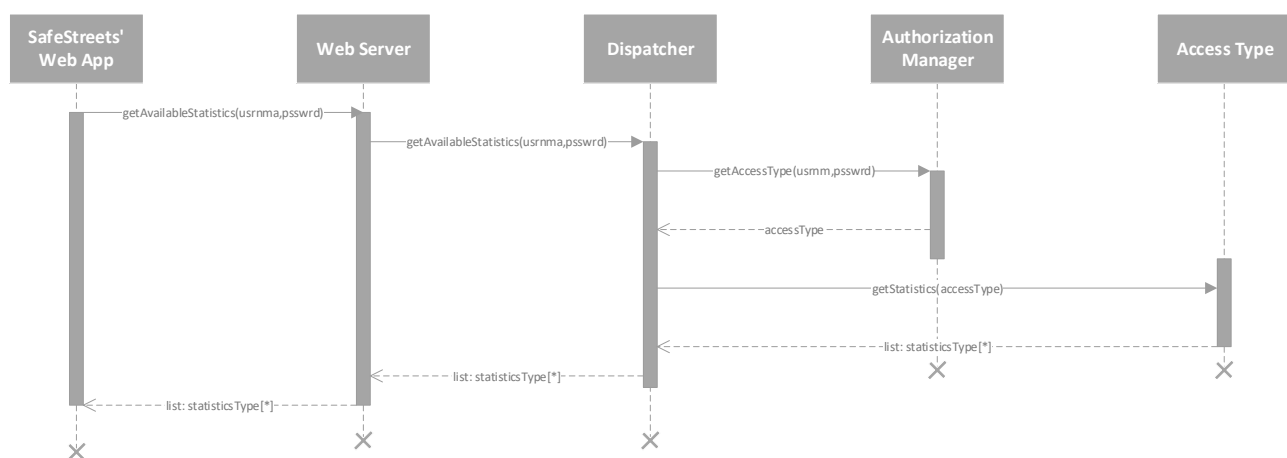
Figure 9 – Sequence diagram of the Get available statistics operation

In this case the sequence diagram highlights how the get statistics operation is performed: the user or the municipality wants to get some statistics from the system, so it asks to the system which are the available statistics according to the AccessType returned by the Authorization Manager. The AccessType will be an object that contains an enumeration of different access type. Each AccessType is associated with a set of operation that can be performed with it: based on the returned AccessType from the Authorization Manager, the system finds out what are the statistics that the user, in this case, can get.

This sequence of operation returns only the available statistics to the user or municipality, the sequence of operation done to get the result of a statistics is defined in the following diagram.

Request Statistics
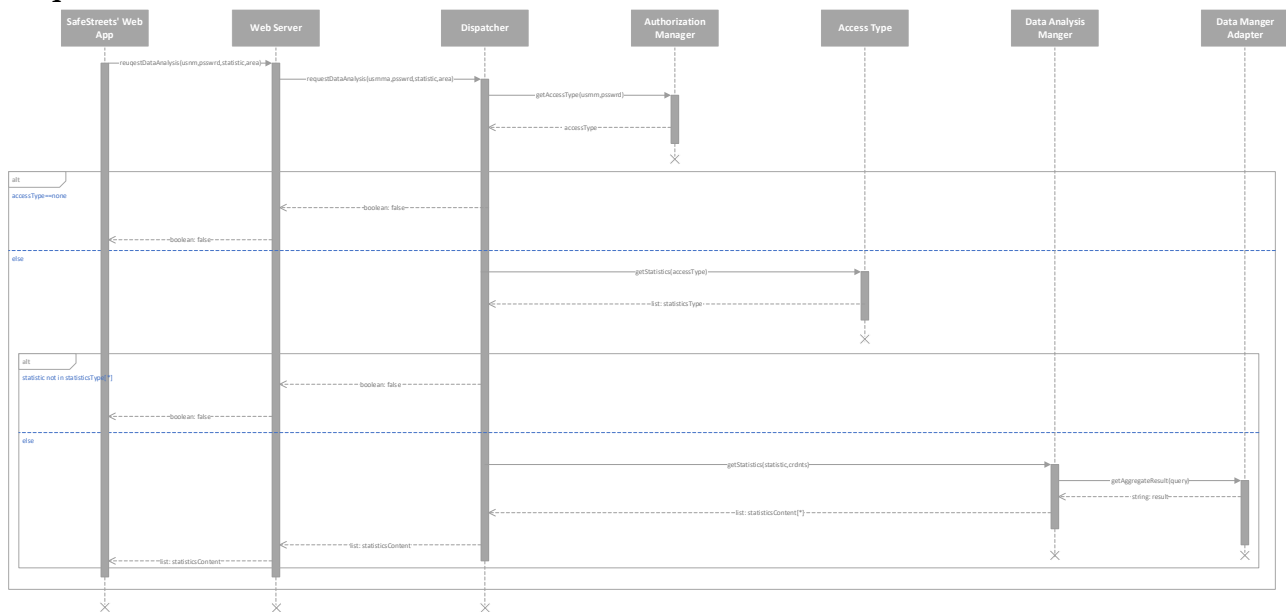
Figure 10 – Sequence diagram of the request statistics operation

This diagram describes how the user or the municipality can get some statistics. In order to get the statistics, the client has logged in and has requested the available statistics and has sent a request for a specific one. When the request is received by the system, the dispatcher verifies that the client can get the required statistics (similar as described in the previous sequence diagram), then the Data Analysis Manager is used to get the required statistics. In the case described in the diagram the statistics can be calculated as an aggregate result of a query (for example the number of violations in a street), but also calling the methods on the Reports Data Interface in order to calculate a more complex statistic.

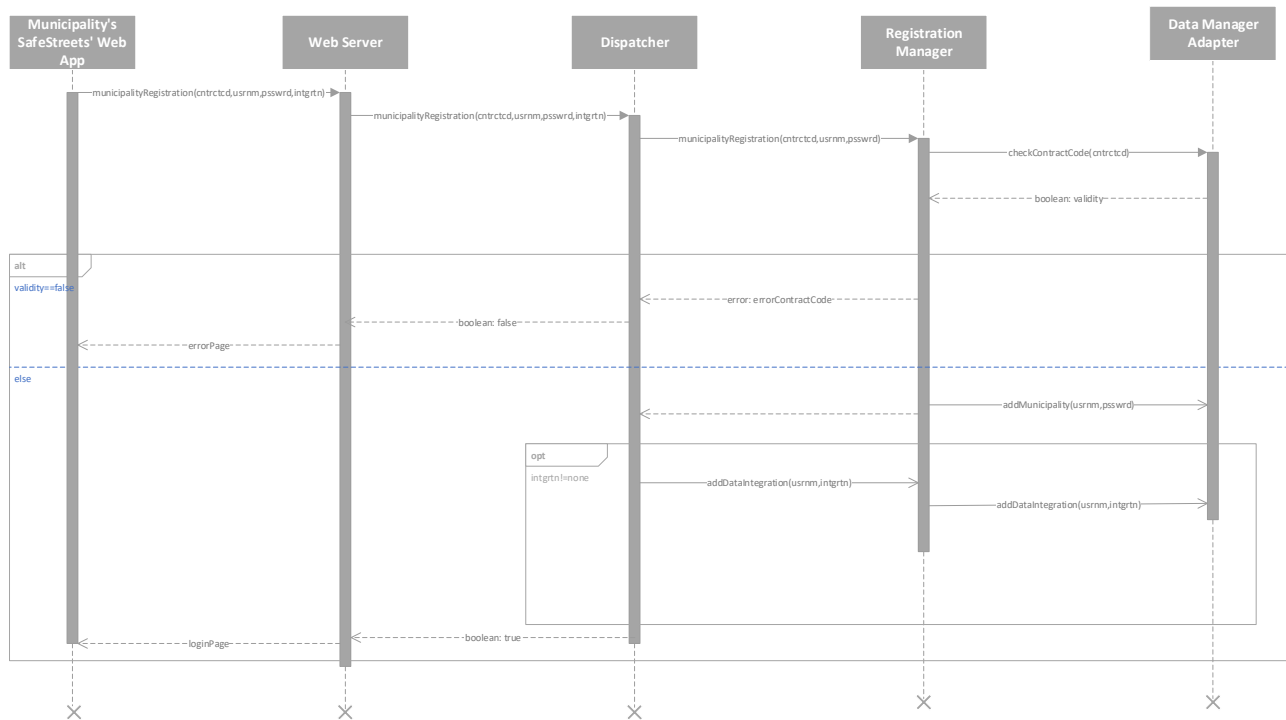**Municipality Registration**

Figure 11 – Sequence diagram of the municipality registration

This sequence diagram describes how the municipality can perform a registration. As stated in the RASD, to perform a registration the municipality must have previously stipulated a contract with SafeStreets, then a contract code and the information about the municipality are memorized in the system, and this information will be used during the registration in order to authenticate the municipality and to retrieve the useful information about them (for example about their jurisdiction area). The methods addMunicipality and addDataIntegration on the Data Manager do not return any value.

User Registration
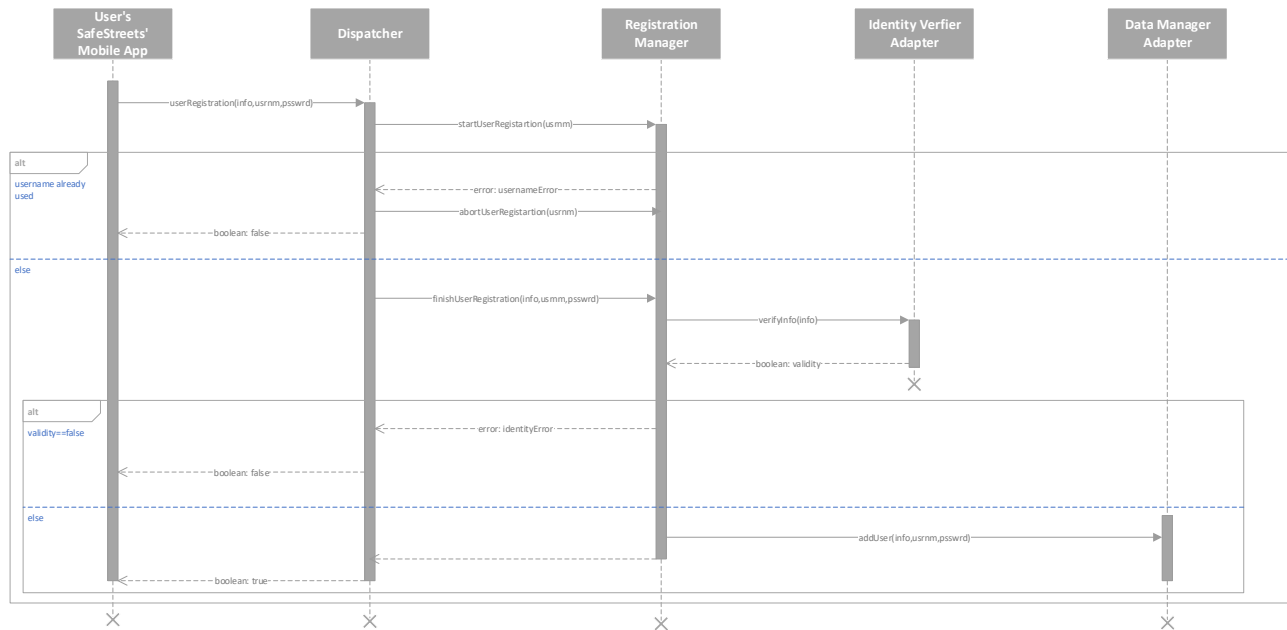
Figure 12 – Sequence diagram of the user registration

The user's registration operation is performed as indicated in this sequence diagram: the registration manager tries to start the registration of the user with startUserRegistration and then to finish it with the invocation of finishUserRegistration, the registration can be interrupted at any time with the invocation of the method abortUserRegistration. The operation can be interrupted by some errors, for example if the username has been already used, or if the identity verifier cannot authenticate the user.

Access Reports Service
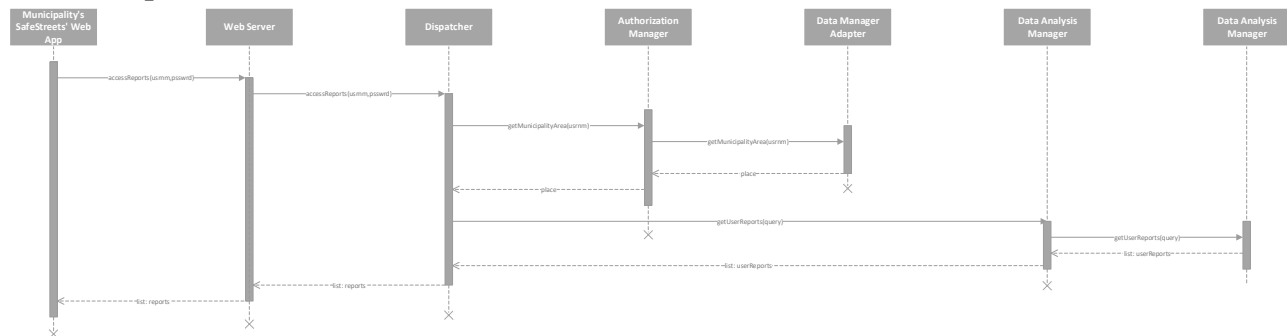
Figure 13 – Sequence diagram of the reports request

In this sequence diagram is better clarified how the municipality can access the reports sent by the user to the SafeStreets' system. In order to get them the municipality can for example use the Web App, the system gets the request and the access rights are verified by the Authorization Manager. The Authorization Manager returns the competence area of the municipality, that will be used by the Data Analysis Manager to return the reports for the municipality.

Data Integration Service

Figure 14 – Sequence diagram of the integration of data from the municipalities

The diagram shows how the data is integrated inside the SafeStreets' system: periodically (for example when the system is not overloaded with work to do) or when a request comes from the Data Analysis Manager (as in this case, where the work of the Data Integration Manager is triggered by the analysis request on a given place), the Data Integration Manager starts to merge the information memorized by the municipality that offers to the SafeStreets' system this possibility.

In this case the Data Analysis Manager needs to perform some statistics on a certain place, so when the Data Manager Adapter receives a query it updates the reports in the DBMS calling the method getLatest on the Data Integration Manager, that will take care of it.

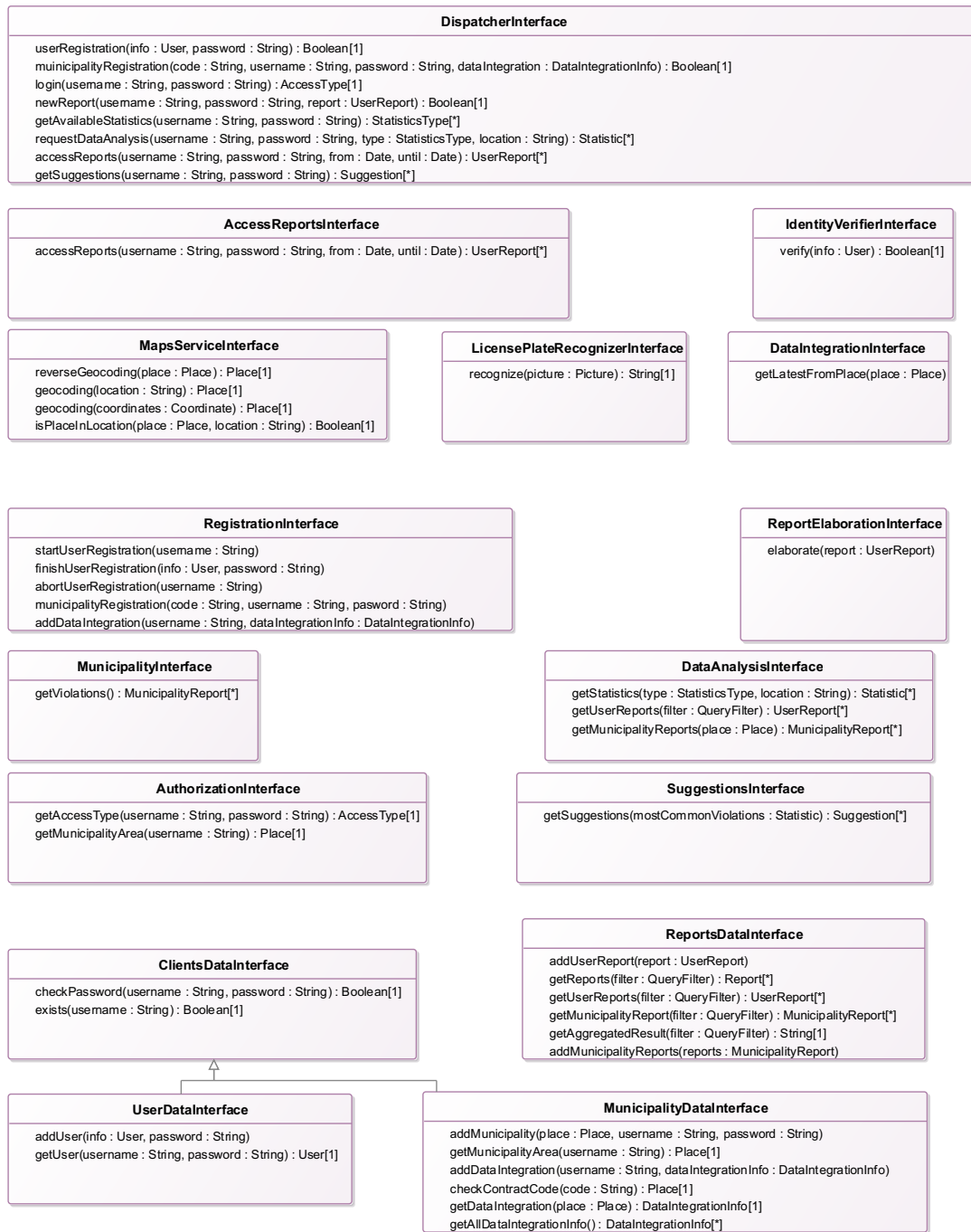## 2.E COMPONENT INTERFACES

Figure 15 – Interfaces used in the component diagram

In Figure 15 the main interfaces used in the component diagram are presented. The diagram also contains some other interfaces – the ones with the tag "Provided" – that are provided by external parties, also missing is the interface through which the web app fetches the code from the web app manager. These interfaces depend heavily on the implementation and are omitted here.

In the image, the first row contains the interfaces that interact somewhat directly with the outside; on the second line the interfaces that allow the internal workings are shown and in the third row there are the interfaces that allow the communication with the data layer. Among these it is worth pointing out that UserDataInterface and MunicipalityDataInterface extend ClientsDataInterface, this was done to

clarify that they have some methods in common and that these methods do not depend on the access type.

The next picture is the class diagram of the model of the application server, it contains the most important data structures that will be used in the implementation phase.
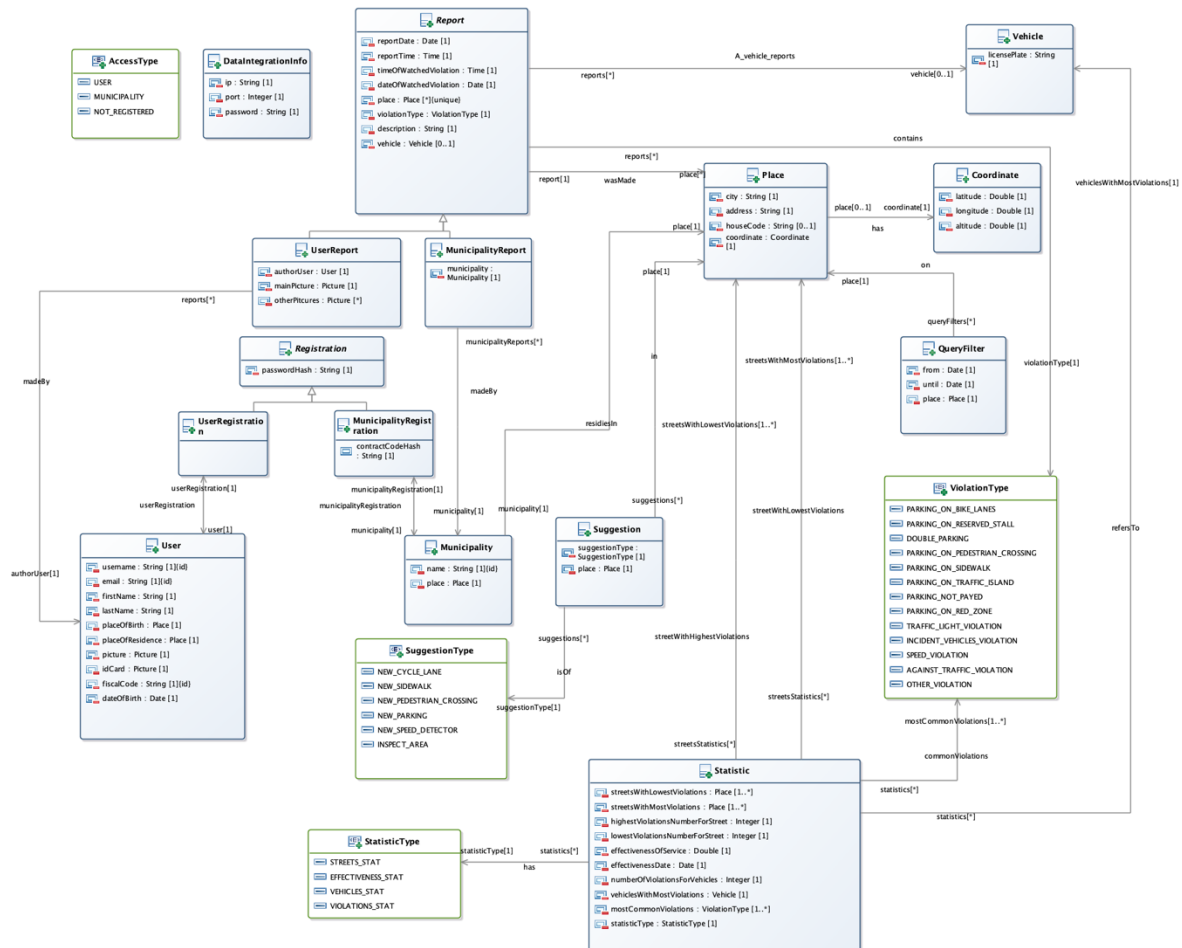

Figure 16 – Class diagram

The ViolationType contains the attribute *isParking: Boolean*, which indicates whether it is a parking violation. The StatisticType contains two attributes: *canBeForUser: Boolean* and *canBeForMunicipality*, which indicate, respectively, whether the statistic can be asked by the user and by the municipality.

2.F   SELECTED ARCHITECTURAL STYLES AND PATTERNS

Four tier architecture
As described in the deployment view the architecture is four-tiered, see Figure 5.

There are: the machines of the user and of the municipality, the web server, the application server and the data server.

The presentation layer is split between the machines of the user and of the municipality and the web server. The application layer is on the application server and the data layer is on the data server.

This architecture allows a separation of the logical layers between the tiers without mixing some logical layers in a single tier. This simplifies the interactions between the tiers, it allows to create redundancy in the servers to improve the performance and the security.

It is possible also to put a firewall between the application server and the web server, given that the web server can be subject to frequent attacks.

Thin Client

The application layer is on the application server. On the clients there is only the presentation layer and the logic associated with it. This architecture allows to have a light client which presents only the information elaborated by the application server.

RESTful Architecture

The RESTful architecture is used in the communications between the clients, the application server and the web server.

The REST architectural has been chosen for the following advantages:

- performance in component interactions;
- scalability, it allows the opportunity to increase the system when the number of clients will increase significantly.
- simplicity;
- modifiability of components to meet new requirements;
- reliability in the presence of failures within components.

To exchange the data between the clients, the application server and the web server will use the protocol HTTPS, which permits to securely transmit the data.

The data will be transferred with files in JSON format. It is one of the most common file format standards with the purpose to transmit the file.

The properties of the Restful architecture are the following.

Client-server architecture

The client asks the server for a service and the server answers the questions done by the clients.

There are two clients: the SafeStreets' mobile app and the SafeStreets' web app. There are also two servers: the web server and the application server.

Statelessness

The application server doesn't store the context of the client between its requests. All the requests from the client contain all the information to provide the service.

Layered system

The client does not know whether it is connected directly to the end server or to an intermediary server. Future intermediary servers can improve the scalability of the system.
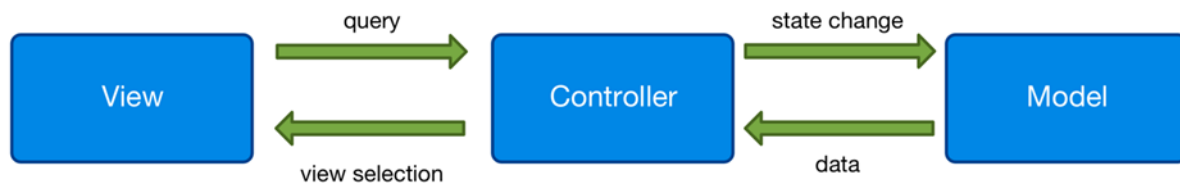
Uniform interface

The server's resources are identified using the Uniform Resource Identifier.

Cacheability

The client stores a cache with the data that refers to its state.

MVC

The Model-View-Controller design pattern will be used. The next picture shows how the pattern is adapted for the system.

Figure 17 – MVC pattern applied to the system

The view, through the gestures of the user or of the municipality, queries the controller. The controller elaborates the request by retrieving the data from the model and by updating the data in model. Finally, the controller selects the view with the new data to show to the user.

The view is present in the SafeStreets' mobile app and in the SafeStreeets' web app. The controller is implemented in the application server. The model is present in the database.

Adapter design pattern

There are four components that act as adapters: DataManagerAdapter, IdentityVerifierAdapter, LicensePlateRecognizerAdapter and MapsServiceAdapter.

These components interact with a system developed by a third part, that provides a fixed interface. The adapters allow to work in the application server without need to worry about the interfaces of the third parties. If there is a need to change an external system, only the corresponding adapter must be changed without touching the other components.

2.G OTHER DESIGN DECISIONS

Relational Database

The data will be stored in a relational database, which is a standard model very reliable. It offers a lot of good properties, that are summarized in the ACID properties: atomicity, consistency, isolation and durability.

The DBMSs of the relational database are provided by a lot of organizations, among which we have chosen the MySQL DBMS. MySQL is an open source DBMS, it is present from the 1995, it is adopted by a lot of organizations and private users and it is currently supported.

Google Maps

For the Maps Service we have chosen the well-known Google Maps. It is a service of very good quality and it is used every day by a lot of people. It has a lot of experience and it is supported by one the largest companies on the world, Google.

Furthermore, Google Maps is easy to use and it has a good user interface, which fits with the design of the SafeStreets' client applications.

3   USER INTERFACE DESIGN

What follows expands what already stated in the RASD[3]: the user flow and the prototypes are represented in detail in Figure 18.

In the figure, arrows represent user flows: the path starts on the element the user interacts with, and points to a destination page – which are named – or to an event that modifies slightly the current page (for example, a pop-up message); a condition can be added on the arrow, to state when the path is active, and arrows can split when the destination depends on a certain condition. Dashed lines represent an interaction with the backend.

The user and the municipality will be presented with the same interface, but with some options not available to both, accordingly with the requirements of the system.

The first page presented to the client is the Login. From there he can sign up or insert his credentials and login. The main page is called *Around me* and shows a map with the streets with the highest number of violations. The Statistics page allows to choose for a statistics type and shows the results of the data analysis for the client's area in a list. The New report process is split in multiple pages: first the user takes a picture, then he can preview it and add others; once he is satisfied, he is prompted to choose a violation type from a list. If the position is unavailable, he is asked to insert one. In the last page the user can insert additional data and send the report. The municipality can also view a page with a list of the reports in its competence area in order of date from the most recent.

The UI adheres to Google's *material design* standards and uses its icons.

---

[3] The RASD refers to UI in the state diagrams in chapter 2.A, in the sections 3.A.1 and 3.D.1.
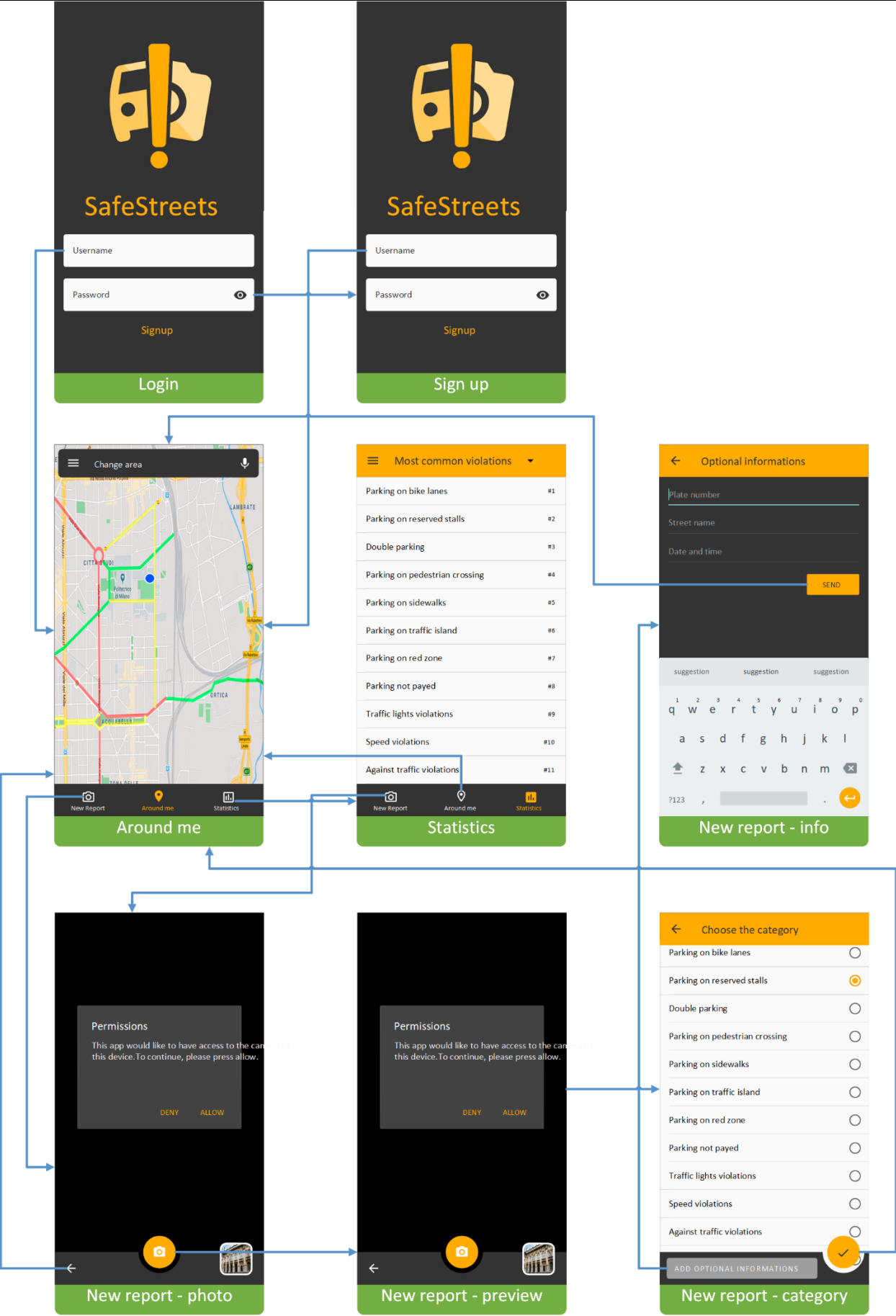
Figure 18 – UX diagram

4 REQUIREMENTS TRACEABILITY

This chapter explains how the functional and the non-functional requirements can be fulfilled through the components of the system.

- **R1:** The reports about the violations are correctly stored.
  - **ReportElaborationManager**, it allows to compile a report about a violation and it forwards the report to the DataManagerAdapter.
  - **DataManagerAdapter**, it interacts with the DBMS by passing the report.
  - **DBMS**, it saves the report in the database.
- **R2**: The user can view the statistics calculated by the system with some exceptions.
  - **DataAnalysisManager**, it calculates the statistics asked by the user.
  - **AuthorizationManager**, it indicates the type of access of the current actor that is interacting with the server.
- **R3**: The municipality can access only the data of the violations of its competence area.
  - **AccessReportsManager**, it offers an interface towards the municipality to retrieve the reports done by the users.
  - **DataAnalysisManager**, it retrieves the user reports from the database.
  - **AuthorizationManager**, it allows access only to the reports that refer to the city of the Municipality.
- **R4**: Violations registered by the municipality can be retrieved by the system.
  - **DataIntegrationManager**, it allows to retrieve the reports done by the Municipality.
- **R5**: The system must avoid the manipulation of the violations.
  - **DBMS**, it ensures the persistency of the data stored in the database, the persistency of the reports done by the users.
  - **DataManagerAdapter**, it is the only way to modify the data stored in the database and it doesn't offer any interface to modify them.
- **R6**: The system must be able to retrieve the position from the user or from the GPS
  - **SafeStreetsMobileApp** or **SafeStreetsWebApp**, it offers a user interface to insert the position of the user and if it is not inserted, the SafeStreetsMobileApp (or SafeStreetsWebApp) asks the GPS the position of the mobile phone.
- **R7**: Only the municipality can access the submitted parking violation of its competence area
  - **AuthorizationManager**, it allows access only to the parking violations that refer to the city of the Municipality.
  - **AccessReportsManager**, it offers an interface towards the municipality to retrieve the reports done by the users.
  - **DataAnalysisManager**, it retrieves the user reports from the database.
- **R8**: The system must allow the user to take a picture or to select one from the device.
  - **SafeStreetsMobileApp** or **SafeStreetsWebApp**, it asks the user if he wants to select one picture from the device or to take a picture, in these cases the SafeStreetsMobileApp (or SafeStreetsWebApp) calls the functionalities of the operating system.
- **R9**: The system accepts reports from the user.
  - **AuthorizationManager**, it allows only the users to make a report.
  - **ReportElaborationManager**, it allows to compile a report about a violation and it forwards the report to the DataManagerAdapter.

- **DataManagerAdapter**, it interacts with the DBMS by passing the report.
- **DBMS**, it saves the report in the database.
- **R10**: The system must calculate some statistics.
    - **DataAnalysisManager**, it calculates the statistics asked by the user or by the municipality.
- **R11**: The municipality can view all the statistics calculated by the system.
    - **DataAnalysisManager**, it calculates the statistics asked by the municipality.
    - **AuthorizationManager**, it allows the municipality to view all the statistics.
- **R12**: The system must suggest interventions to the municipality.
    - **SuggestionManager**, it calculates some possible interventions to do in the city of the municipality.
    - **AuthorizationManager**, it allows the municipality to view the suggested interventions.
- **R13**: The system accepts only reports with a valid plate number and position.
    - **ReportElaborationManager**, it accepts the report done by the user only if the LicensePlateRecognizer returns a plate number and if the MapsService can calculate the right position of the violation.
    - **MapsServiceAdapter**, it allows to communicate with the Maps Service.
    - **LicensePlateRecognizerAdapter**, it allows to communicate with the Plate Recognizer Service.
- **R14**: The system must allow the user to perform the registration and the login.
    - **AuthorizationManager**, it allows the user to make the login.
    - **RegistrationManager**, it allows the user to make the registration.
- **R15**: The system must allow the municipality to perform the registration and the login.
    - **AuthorizationManager**, it allows the municipality to make the login.
    - **RegistrationManager**, it allows the municipality to make the registration.
- **R16**: The system must ask the user the non-mandatory attributes of the report.
    - **SafeStreetsMobileApp** or **SafeStreetsWebApp**, it asks the non-mandatory attributes of the report.
- **R17**: The system must communicate with the Identity Verifier.
    - **IdentityVerifierAdapter**, it allows to communicate with the Identity Verifier.
- **R18**: The system must communicate with the Plate Recognizer Service.
    - **LicensePlateRecognizerAdapter**, it allows to communicate with the Plate Recognizer Service.
- **R19**: The system must communicate with the Maps Service.
    - **MapsServiceAdapter**, it allows to communicate with the Maps Service.

Regarding the other requirements:

- **Performance** requirements
    - **Dispatcher** and **AccessReportsManager**, they offer some interfaces to communicate with the application server at any time.
- **Reliability**
    - All the modules will be tested properly in order to be reliable.
- **Availability**
    - All the modules are designed to be ready at any time, the **Dispatcher** and the **AccessReportsManager** allow to access the application server whenever needed.

- **Security**
  - **DBMS**, it ensures the persistency of the data stored in the database, the persistency of the reports done by the users.
  - **AuthorizationManager**, it manages the accesses to the server and it authorizes, according to certain rules, who can access the various functions.
- **Maintainability**
  - All the modules are designed to be expanded in the future and to be readable with the comments and the documentation.
- **Portability**
  - The various adapters allow to interact with the systems of the third parts by changing only a few modules.

In the mapping, the non-functional requirements are mapped to the most important requirements, but they depend strongly on the entire system, on the hardware and on the technologies that support the system.

In almost all the requirements the component Dispatcher is used, because it redirects the requests of the client to the right component.

5   IMPLEMENTATION, INTEGRATION AND TEST PLAN

5.A   IMPLEMENTATION

The system is divided in the following subsystems:

- Data management, which includes the configuration of the DBMS, and the component DataManagerAdapter;
- Third Parties Adapters, in this part the services exposed by the third parties are integrated in the system, through the use of the adapters: IdentityVerifierAdapter, LicensePlateRecognizerAdapter and MapsServiceAdapter;
- Report Elaboration Manager;
- Registration Manager;
- AuthorizationManager;
- Dispatcher;
- SafeStreetsMobileApp;
- WebAppManager;
- SafeStreetsWebApp;
- DataIntegrationManager;
- DataAnalysisManager;
- SuggestionManager;
- AccessReportsManager.

The subsystems that are not specified are components.

The implementation, integration and test phase will be done using a bottom-up approach and a thread approach. First will be developed the most important components and those that don't depend on other components, for example the adapters, which relies on third-part systems that are already developed.

After the integration and the testing of the most important components, will be integrated and tested those components that can show the user a prototype of the system: the ReportElaborationManager, the Dispatcher, the SafeStreetsMobileApp, the RegistrationManager and the AuthorizationManager.

For the other functionalities again the bottom-up approach will be used: for the DataIntegrationManager, the DataAnalysisManager, the SuggestionManager and the AccessReportsManager.

Accordingly to the previous definitions of subsystems, the following Table 1 highlights the priority (the precedence in the implementation order of this functionalities), the importance (that is how much the services exposed by the functionality are important for the user) and the effort (that indicates the difficulty in the development of the functionality) of the system's functionalities.

Table 1 – Subsystems Analysis

FUNCTIONALITY	IMPORTANCE	EFFORT	PRIORITY
DATA MANAGEMENT	High	High	High
DISPATCHER	High	Medium	High
THIRD PARTIES ADAPTERS	Medium	Medium	Medium
REPORT ELABORATION	Medium	Medium	High
AUTHORIZATION MANAGER	High	Medium	High

SAFESTREETS MOBILE APP	High	High	Medium
SAFESTREETS WEB APP	Medium	Medium	Medium
WEB APP MANAGER	Medium	High	High
DATA INTEGRATION MANAGER	Low	Medium	Low
DATA ANALYSIS MANAGER	Medium	High	Medium
SUGGESTIONS MANAGER	Low	High	Low
ACCESS REPORT MANAGER	Low	Medium	Medium
REGISTRATION MANAGER	Medium	Low	Medium

The table gives a hint on how the different functionalities must be developed and the importance of some functionalities to the development of the system.

RoadMap
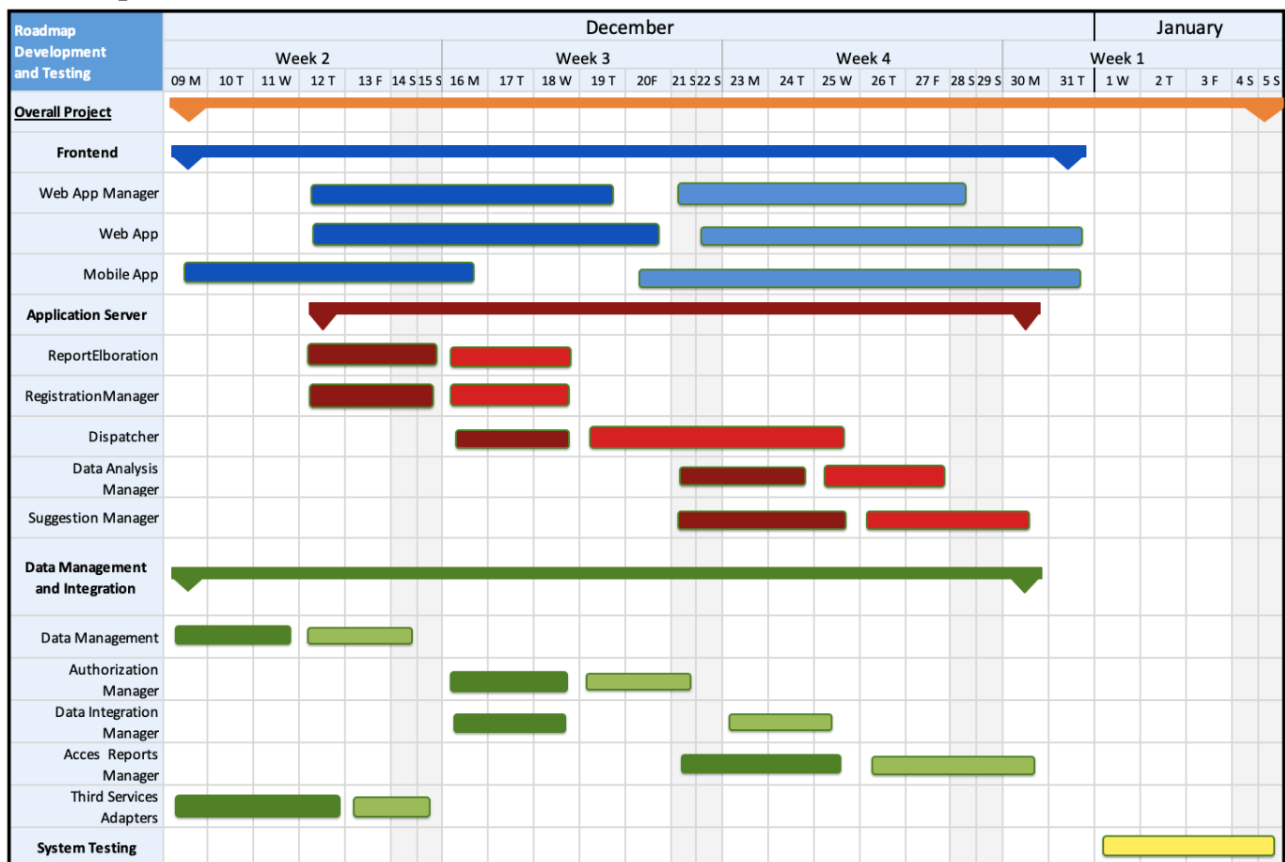
Figure 19 – Project Roadmap

The project roadmap is highlighted in the above picture: the symbol ⬛ indicates the interval of time that is required to complete the main task. The dark symbol ⬛ indicates the implementation phase, which includes also the unit testing, of the corresponding subsystem. The light symbol ⬛ indicates the integration phase and the integration testing phase.

The overall project will take approximative a month of work, and the project is divided in three main tasks: Frontend, Application Server, Data Management and Integration and System Testing. These tasks can be carried out almost in parallel with the synchronization on the interface methods previously defined and, as described above, the Test part will make use of an approach of type bottom up and with thread, to test the different component developed in parallel.

The first two activities that will start are the Data Management part and the Third Service Adapters. Then the Application Server part will follow, that provides the method to manipulate the data and to elaborate them, the Fronted part can be developed in parallel too.

Also, as soon as the functionalities of some components are defined, the test part starts: this part will be carried out all together with the rest of the development and will occupy the development until the end of the project.

The implementation will begin in the order specified in Figure 20.
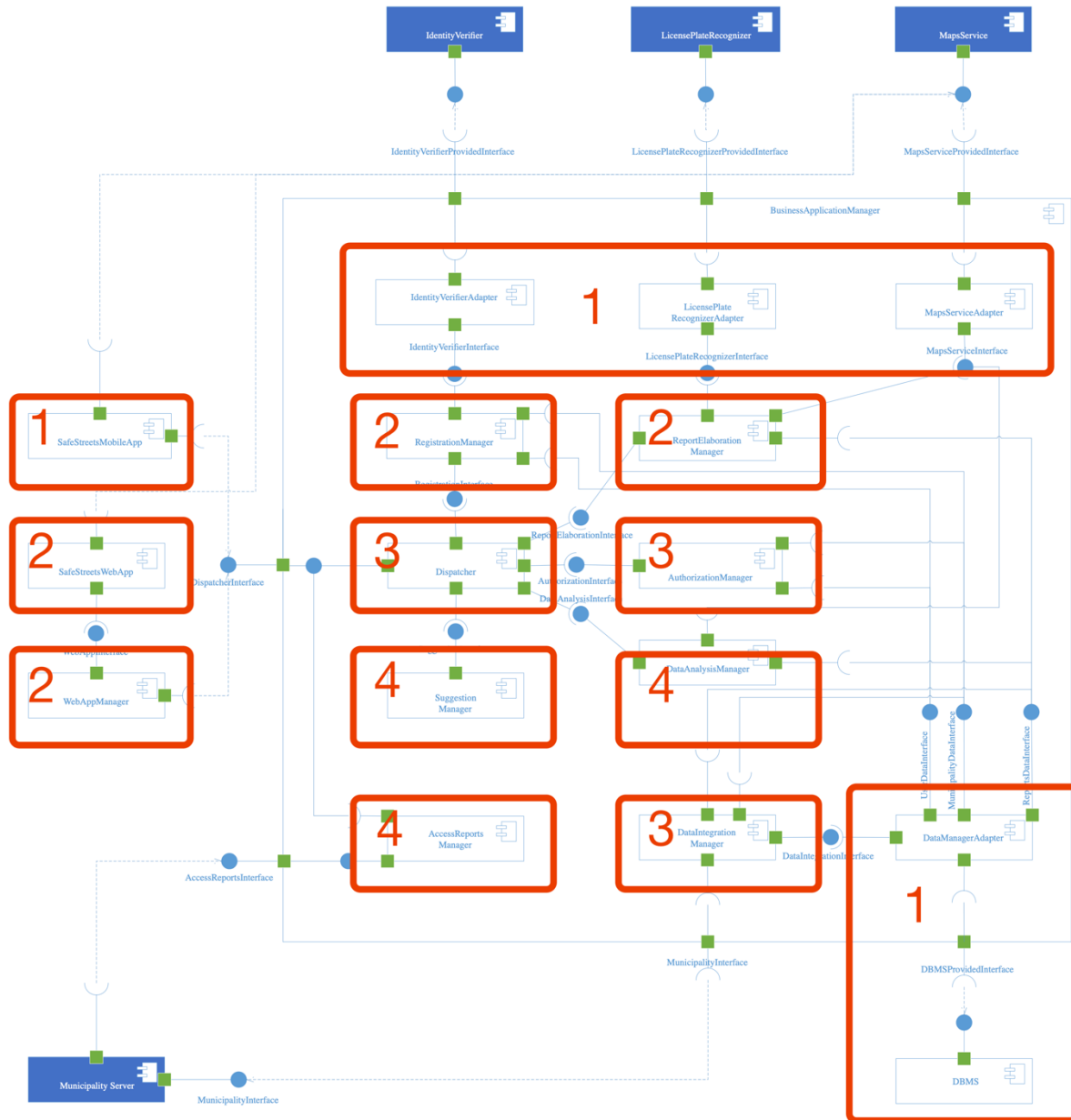
Figure 20 – Implementation phases

The Data management and the Third Parties integration will begin together because following a bottom up approach and because the Data Management is crucial to the whole system. Also, the SafeStreetsMobileApp will be developed at the beginning to show to the user and to the municipality a first prototype of the application and because it requires a lot of effort.

Then the development proceeds with the RegistrationManager and the ReportElaborationManager. At this step the development of the WepAppManager and of the SafeStreetsWepApp will begin. The second step will begin when the mobile app will not be completely implemented yet, as it is possible to see in the roadmap.

At the third step, the Dispatcher and the AuthorizationManager will be developed, with them will begin also the DataIntegrationManager.

In the final step there are the DataAnalysisManager, the SuggestionManager and the AccessReportsManager, which are functionalities that require the other components to be integrated with.

5.B INTEGRATION AND TESTING

Each functionality of the single components will be tested with unit testing during the development of the components. The components will be integrated in the following order.
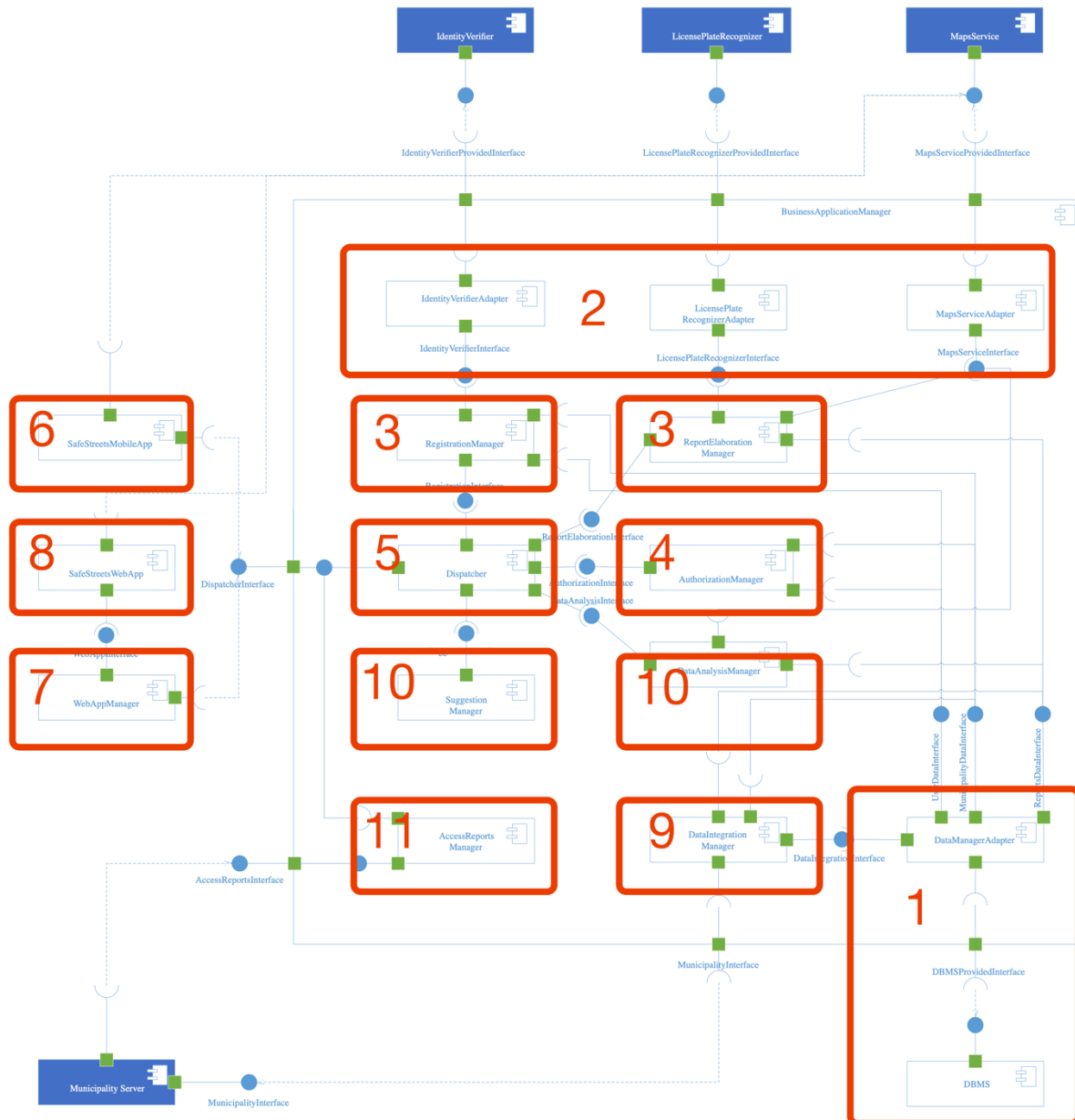
Figure 21 – Integration phases

After the integration of a component, an interface test will be performed.

1: DataManagerAdapter testing
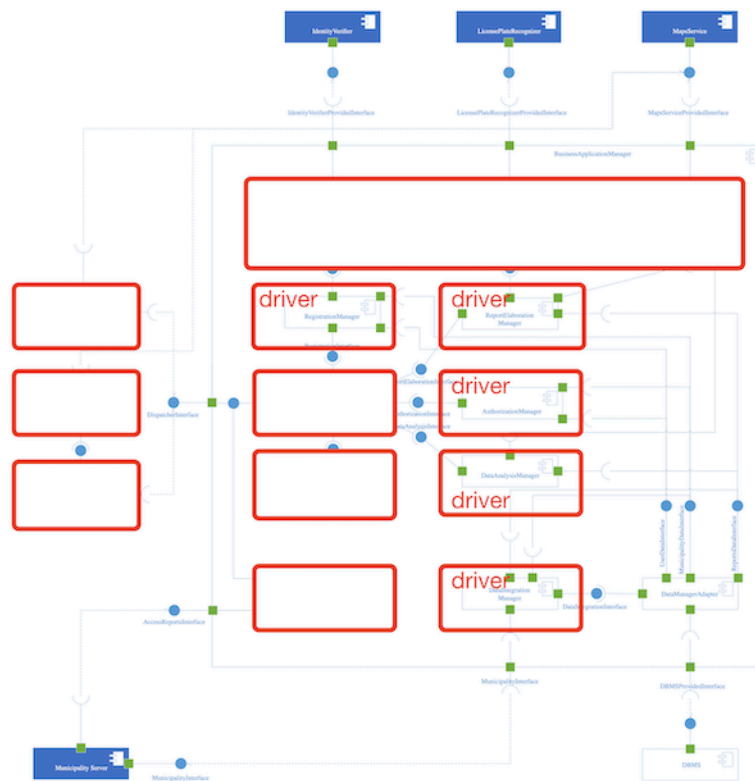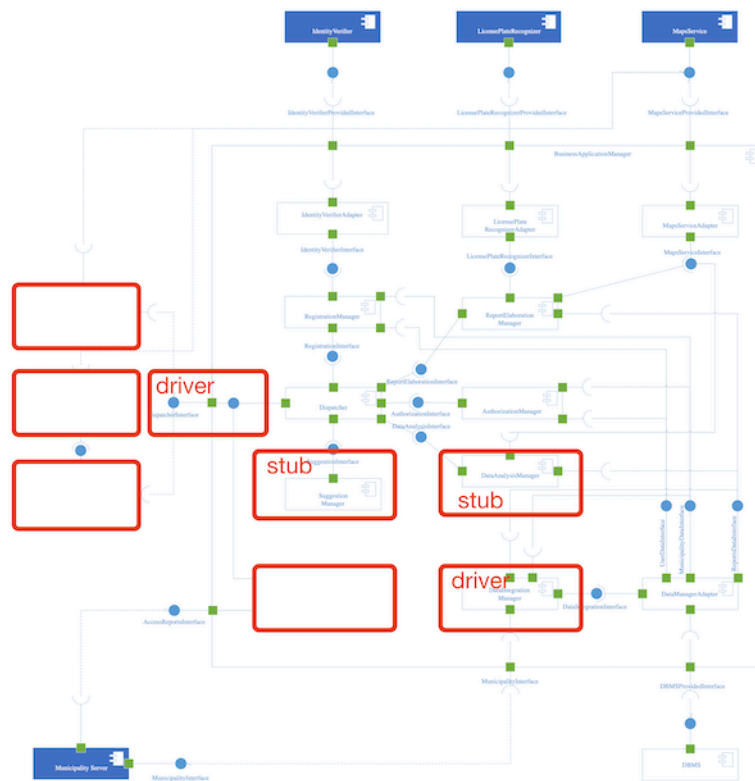

2: Third Parties Adapters testing
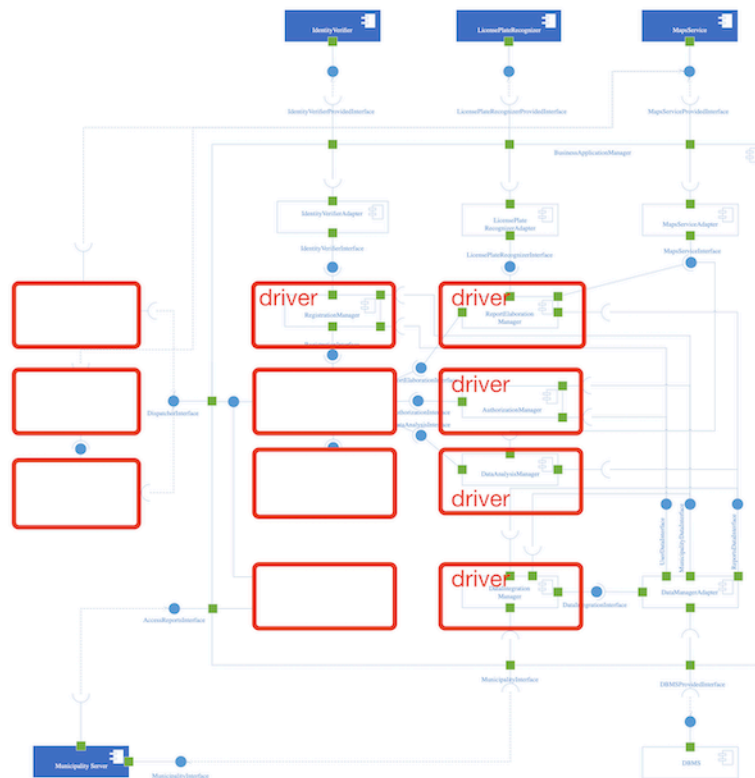
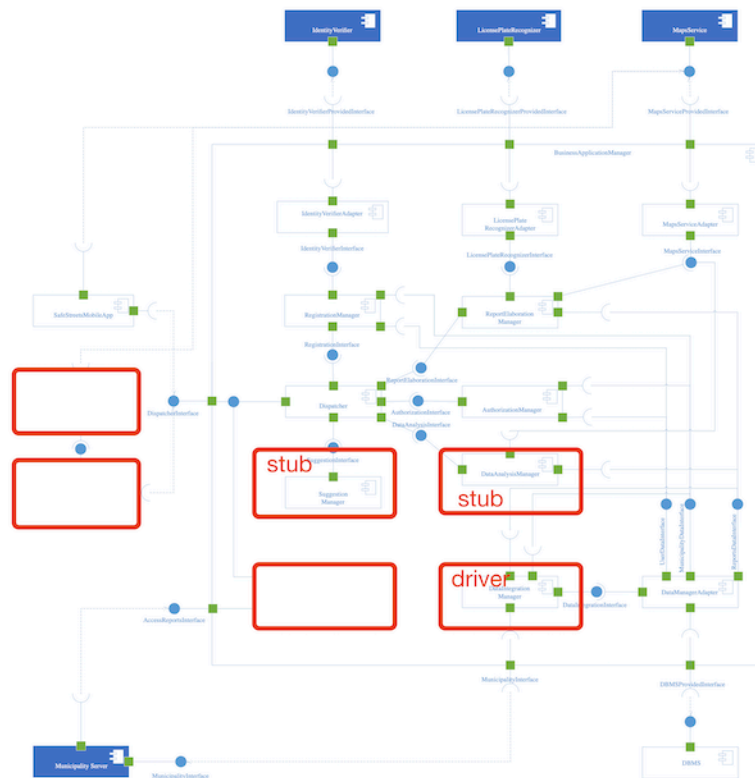### 3: ReportElaborationManager and RegistrationManager
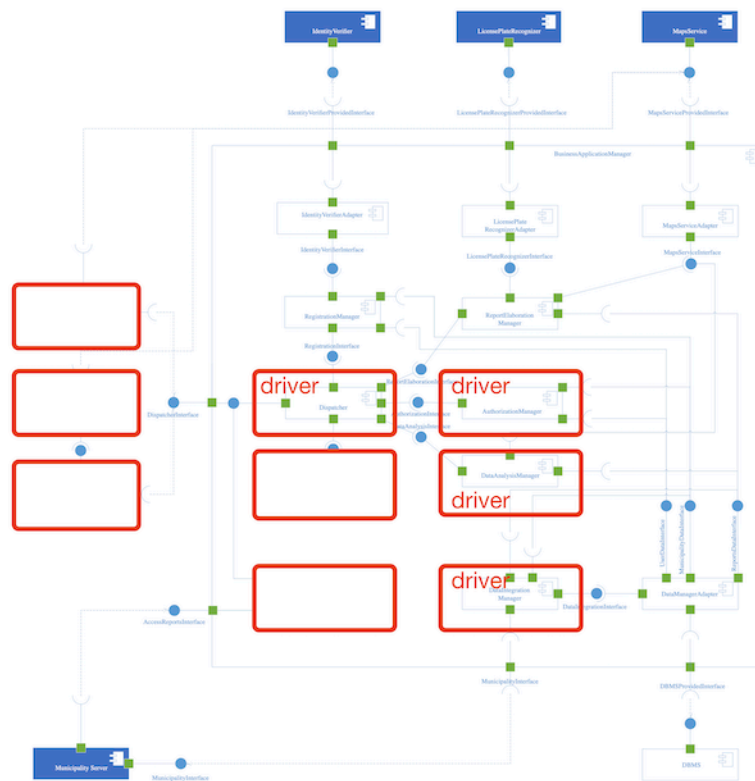
4: AuthorizationManager
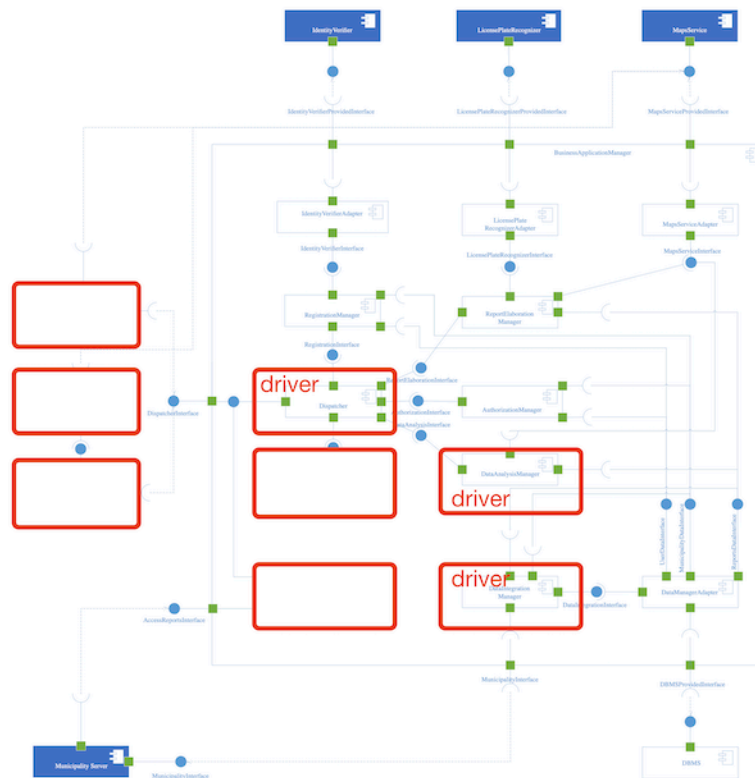
5: Dispatcher

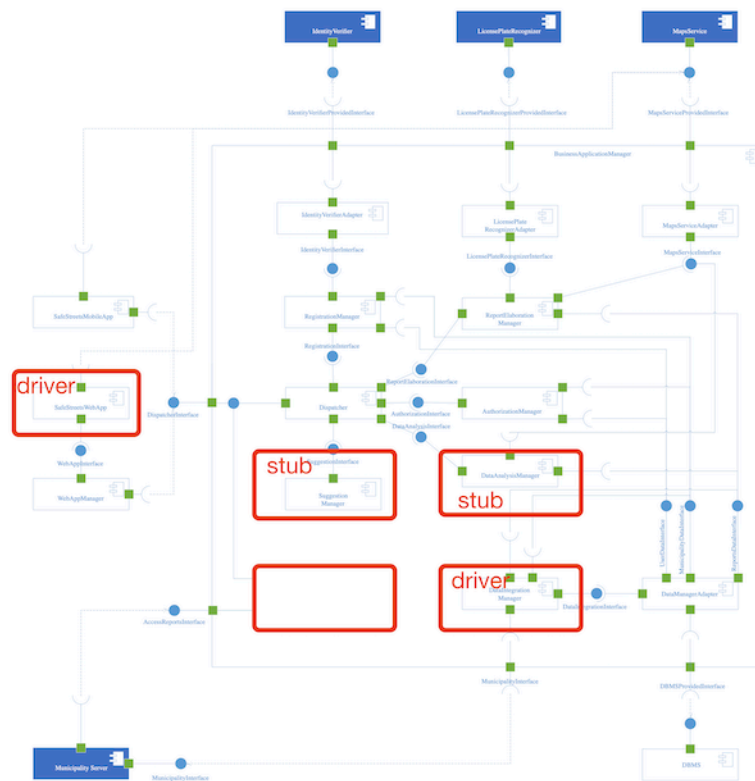**6: SafeStreetsMobileApp**

7: WebAppManager
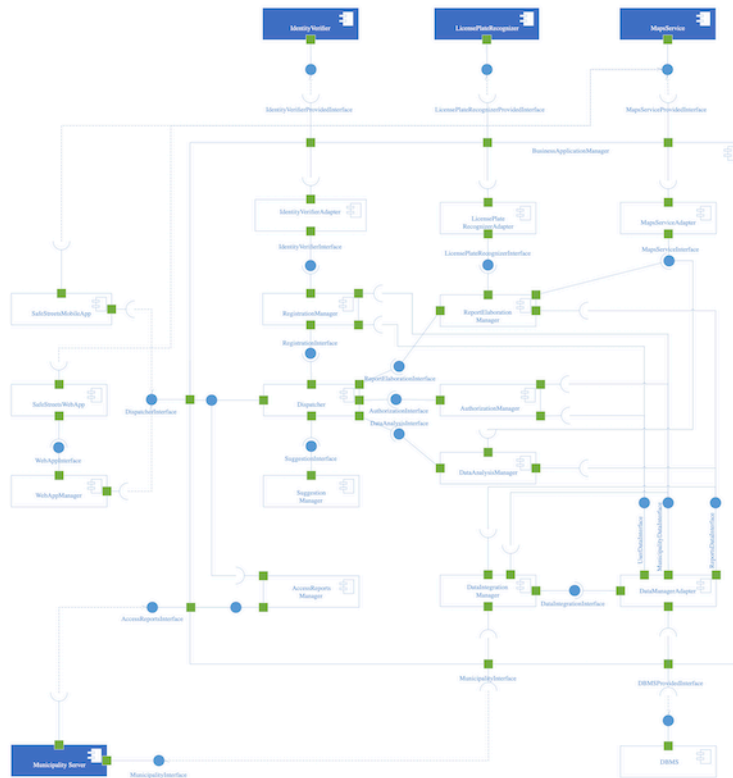

8: SafeStreetsWebApp
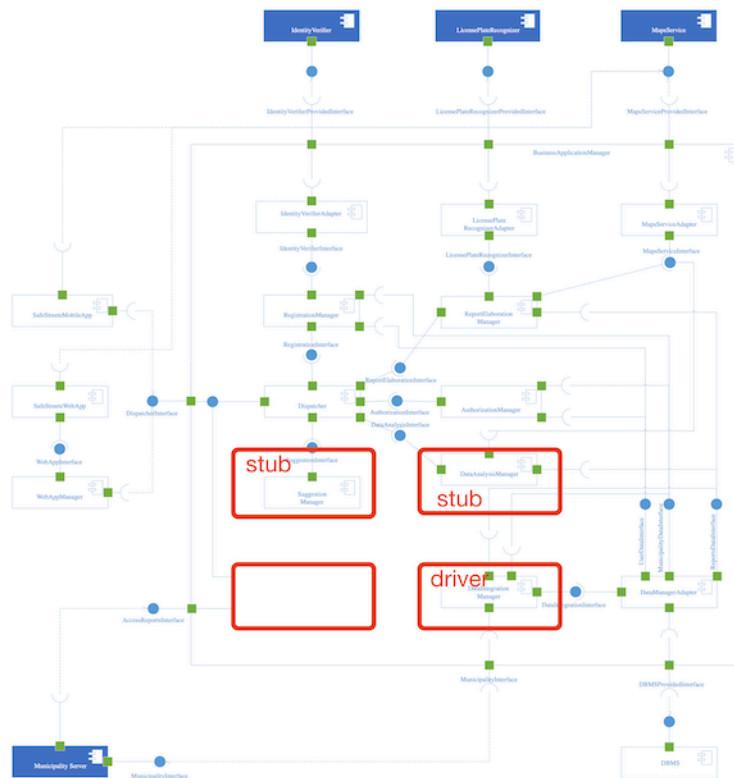
9: DataIntegrationManager
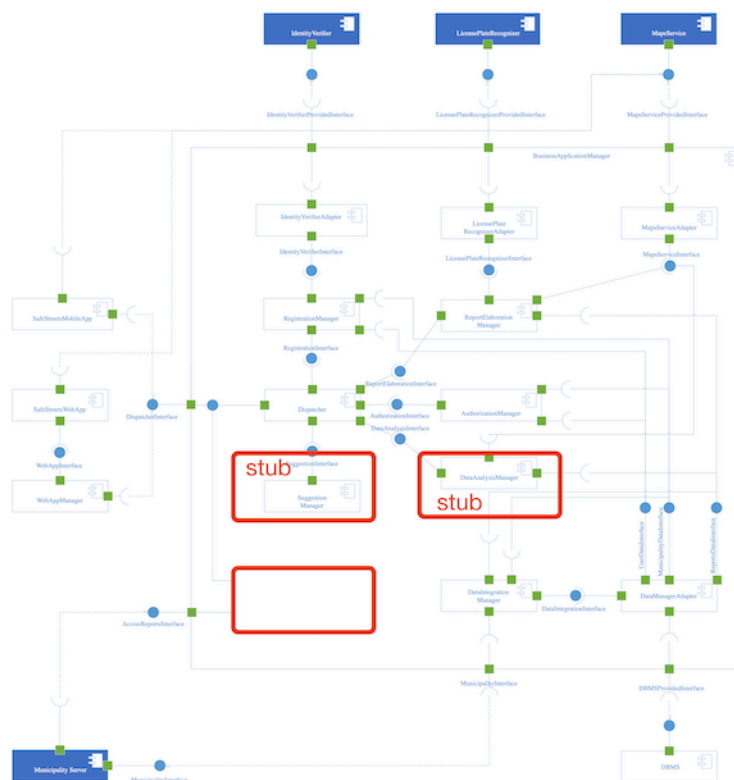
10: DataAnalysisManager and SuggestionManager
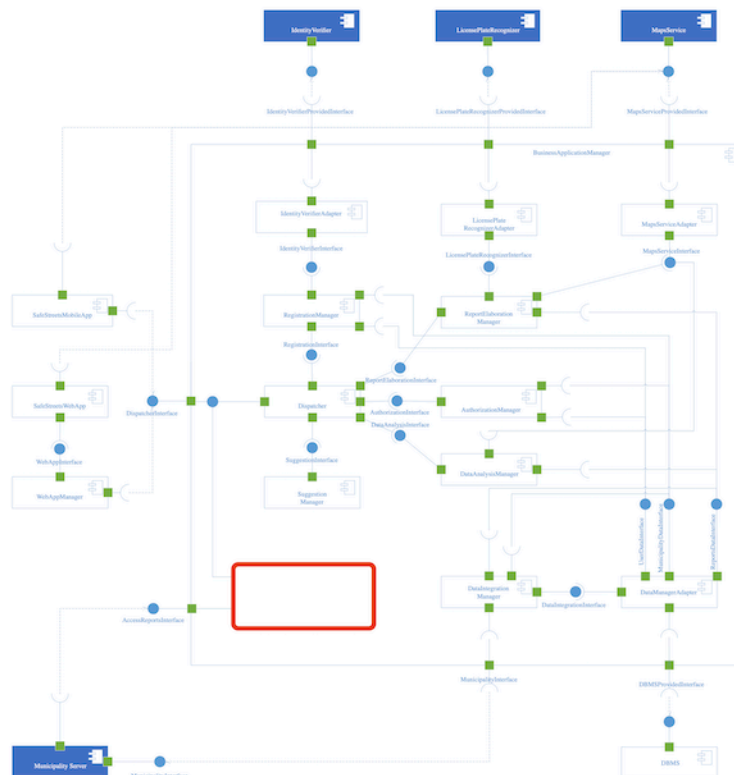
11: AccessReportsManager

Finally, the system test can be done. At this stage all the system is tested in order to evaluate the compliance with the requirements.

6 EFFORT SPENT

- Abbo Giulio Antonio

DESCRIPTION OF THE TASK	HOURS
Architectural design	18
User interface design	20
Requirements traceability	0,5
Implementation, integration and test plan	0,5

- Accordi Gianmarco

DESCRIPTION OF THE TASK	HOURS
Architectural design	33
User interface design	0,5
Requirements traceability	1,5
Implementation, integration and test plan	2,5

- Bonetti Massimiliano

DESCRIPTION OF THE TASK	HOURS
Architectural design	18
User interface design	0,5
Requirements traceability	6,5
Implementation, integration and test plan	10

7 REFERENCES

- Slides of the Software Engineering 2 course by prof. Di Nitto at Politecnico di Milano
- GOOGLE. Material Design Guidelines.
  [Accessed 8 November 2019]. Available from: https://material.io/