Politecnico di Milano
AA 2019/2020

POLITECNICO
MILANO 1863

DD – Design Document

Version 1.0 – 15/12/19

Author: Ines Hafdaoui

Professor: Elisabetta Di Nitto

Table of contents

1. Introduction

1.1 Purpose

This document aims at giving a more technical description of Safestreet application going deeply on functional requirements implementation.

Indeed, in the RASD document is provided a whole view of how the system is expected to act, having obtained a high level view of the structure.

The desidn document purpose is to clarify details about design, providing a guidance to architecture design process.

By the moment the focus is in claryfing how every detail has to be implement, here all components of the system are described, with the related run-time processes and all the design choices are listed and motivated.

In particular, the following topics will be discussed:

- The high-level architecture (non è già stato fatto nel RASD?)
- The main components, their interfaces and their deployment
- The runtime behavior
- The design patterns;
- (Additional details about the user interface)
- A mapping of the requirements on the architecture's components
- Implementation, integration and testing plan

1.2 Scope

Basically Safestreet's scope is to provide users a tool to notify authority parking violations when they occur.
In particular, the application allows user to fulfill a form in which he can send pictures of violations, including their date, time, and position, select the type of violation and optionally give a brief description of vehicle that has committed the violation reported, sending it to authorities.
SafeStreets stores the information provided by users, completing it with suitable metadata. In particular, when it receives a picture, it runs an algorithm to read the license plate , and stores the retrieved information with the violation, including also the type of the violation and the name of the street where the violation occurred (acquired thanks to GPS and so providing the geographical position of the violation). In addition, the application allows both end users and authorities to mine the information that has been received, offering them the possibility to view data about violations and the statistics retrieved.

1.3. Definitions, acronyms, abbreviations

1.3.1. Definitions

- Parking violation

  With parking violation is intended the act of parking a motor vehicle in a inappropriate place or in an unauthorized manner. The system considers among parking violation:

  - parking in a reserved place for example in a bus stop or in handicapped zone without an appropriate permit or in front of a garage entrance or in a parking reserved for employees of a company;
  - parking in areas that are not parking i.e. in places not used for parking for example on pedestrian crossing, on a sidewalk (unless specifically allowed by signs), parking in, too close to or within an intersection, railroad crossing or crosswalk;
  - double parking;
  - parking at a parking meter without paying, or for longer than the paid time;
  - vehicle unauthorized for example with expired or missing license plate or license plate ‘tabs’ without proper safety vehicle inspection decal;
  - parking in a wrong way for example outside marked squares or parking facing against the direction of traffic.

- End user

  End users will be used to refer to common citizens who can be registered or not, who can be interested in consulting statistics

- Authorities

  With authorities we refer to anyone representing an authority

- User

  With user we refer to citizens i.e. to every person who is not part of any authority body and who is already signed up

1.3.2. Acronyms

- GPS: Global Positioning System
- API: Application Programming Interface
- DD: Design Document
- RASD: Requirements Analysis and Specification Document
- UI: User Interface
- TCP: Transmission Control Protocol
- IP: Internet Protocol

1.3.3. Abbreviations

- Stats: Statistics
- i.e.: id est, that is

1.4. Revision History

- Version 1.0 :
  - First Release

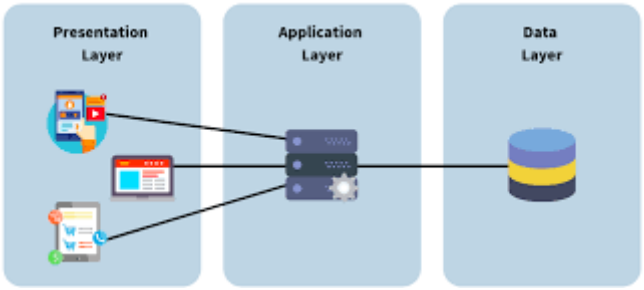
1.5. Reference Documents

- Specification document “SafeStreets Mandatory Project Assignment”
- IEEE Standard for Information Technology - System Design – Software Design Descriptions
- UML Diagrams: https://www.uml-diagrams.org/
- UML Stereotypes: https://www.uml-diagrams.org/stereotype.html/

2. Architectural Design

2.1. Overview: High-level components and their interaction

The application to be developed is a distributed application for which a three-tier architecture has been chosen. Three tier refers to three layers, Presentation, Application and Data access that are thought to be divided on three different hardware layers i.e. on three machines (or groups of machines). Presentation layer deals with user interaction with the system, Application is about handling the business logic and functionalities that are expected to be implemented, and data access that manages information with access to the database. This approach grants to any logic layer its own dedicated hardware with the advantages to guarantee scalability, flexibility and low coupling. Thus server side results consisting in two layers, the second layer that contains business logic and the third one that is concerned with data access thus separating client from data and increasing safety in accessing to data stored.

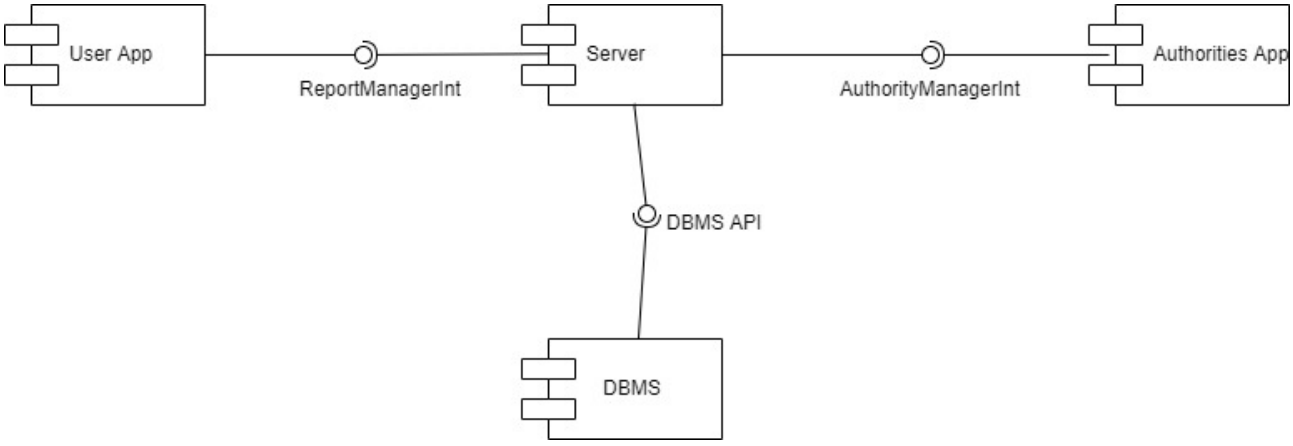


The Server of the Application interacts with clients in an asynchronous way (don't wait the answer) whereas user sends to application layer synchronous messages so it waits for server answer to consult stats and for the report submission that is correctly sent after server checks. Finally, the server in the application level communicates with the database server in synchronously way to retrieve information and in asynchronously way to store information.

2.2. Component view

2.2.1. Main components

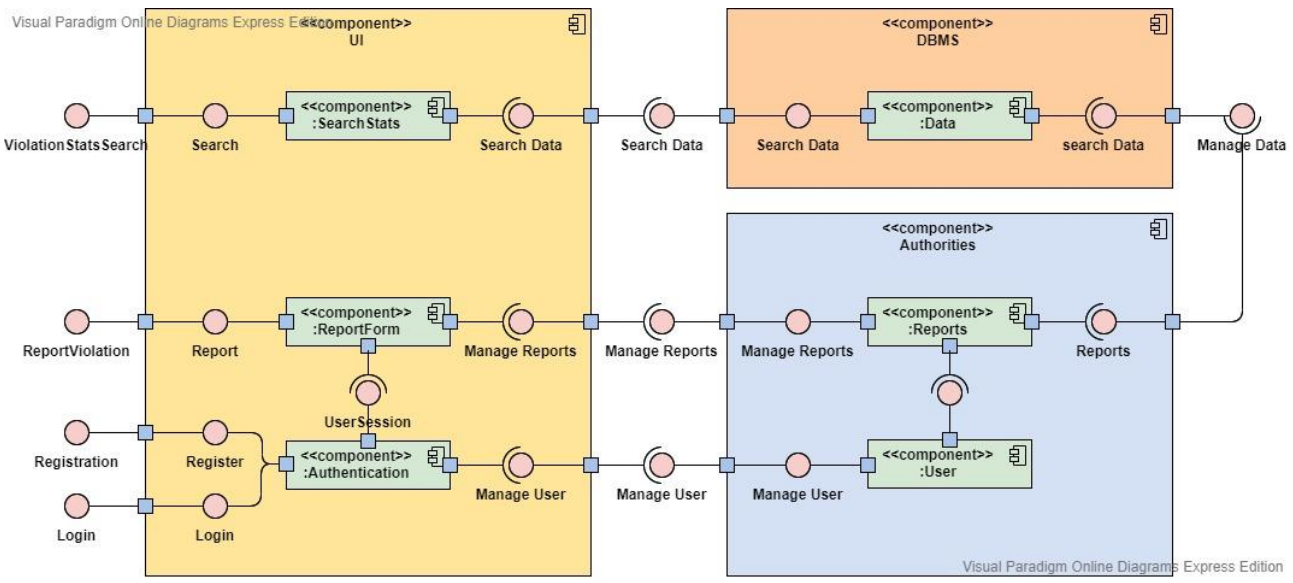
In the perspective of giving a whole view of the main components involved, the following very schematic diagram gives an idea of the general operation and of the structure.



The kind of device that support the application is not more precised as the application, for its purposes, is thought as a mobile application, so we will always refer to a mobile app.

2.2.2. Component Diagram

In the following Component Diagram only logic components are depicted in order to make simple and clear the logic component structure and how to organize source code. This diagram gives a structure to code showing how components will interact through many different interfaces.

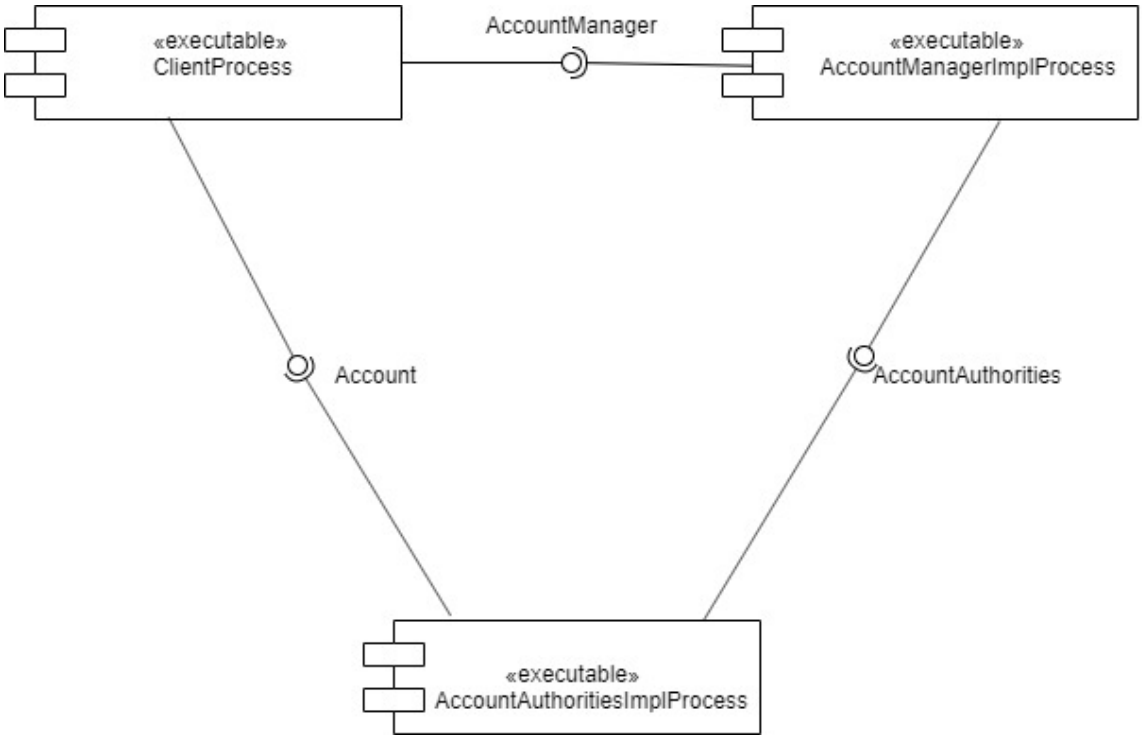


A more detailed description about interfaces functionalities is listed as follows:

- UserSession : it handles procedures to check account is already registered and to provide to user registered with the possibility to report violation (it checks for the form to be correctly fulfilled by the User)
- ManageUser : this component contains all the procedures to allow a User to register to SafeStreet
- ManageReports : it receives the reports, checks for form to be correctly fulfilled and in case forwards it
- Reports : it contains all the logic about reports management and is concerned with providing functionalities to complete them with suitable metadata and retrieving license plate and geographic position
- ManageData : it deals with report storing as data and notification to authorities
- SearchData : it comes into play when an end user wants to consult stats and more in general when a component wants to retrieve information
- AuthorityManager : handles authorities account

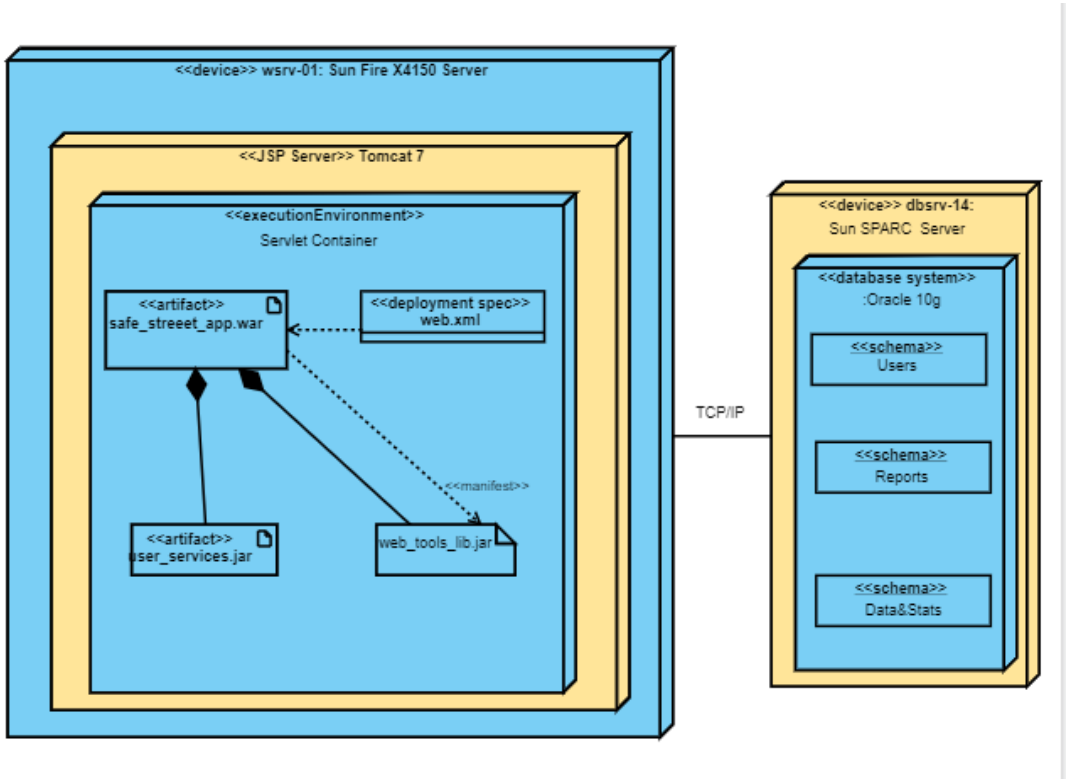
2.3. Deployment view

In the following picture executables and their interaction by means of interfaces, are depicted.



2.3.1. Deployment Diagram

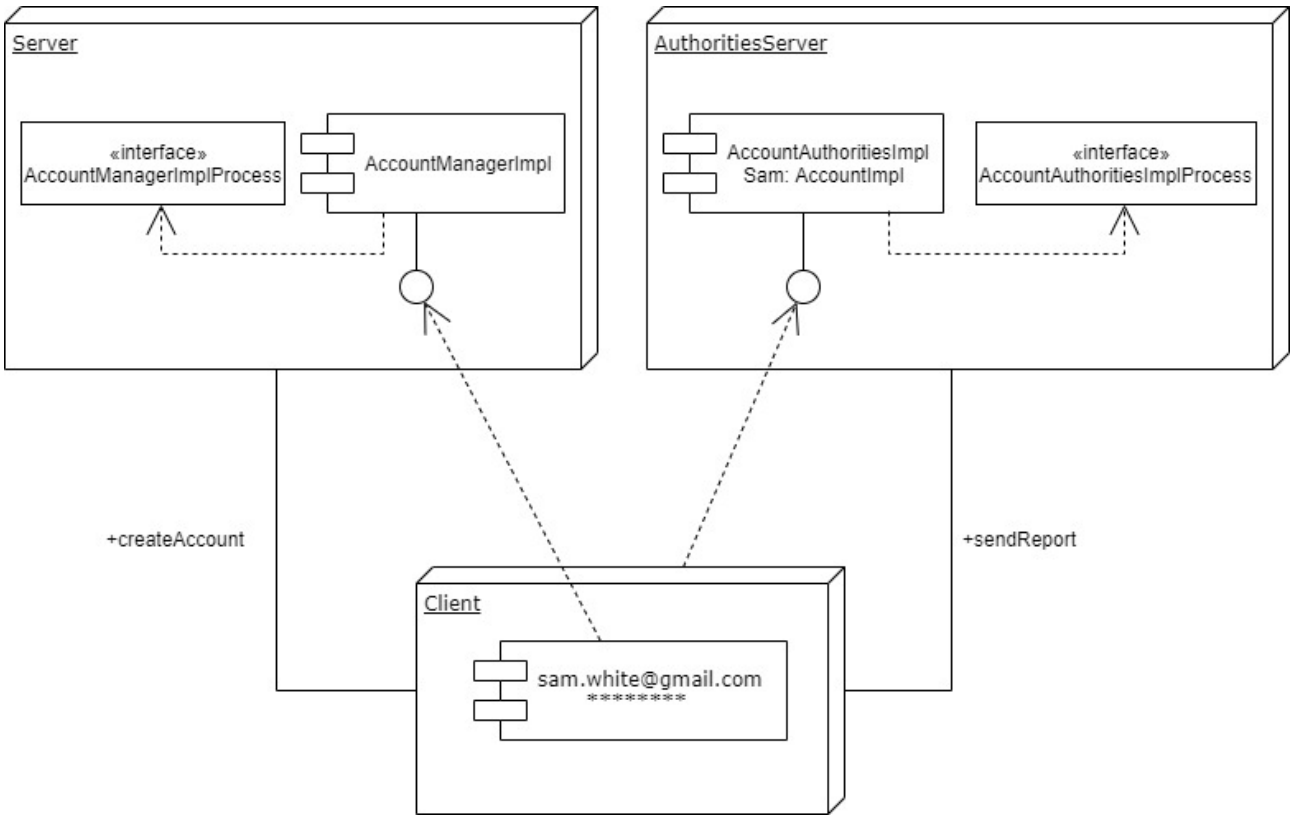
This section goes deep on technical choices about server devices used and protocol interaction between webserver and database server including also environment component, and structure of the different containers.



2.4. Runtime view

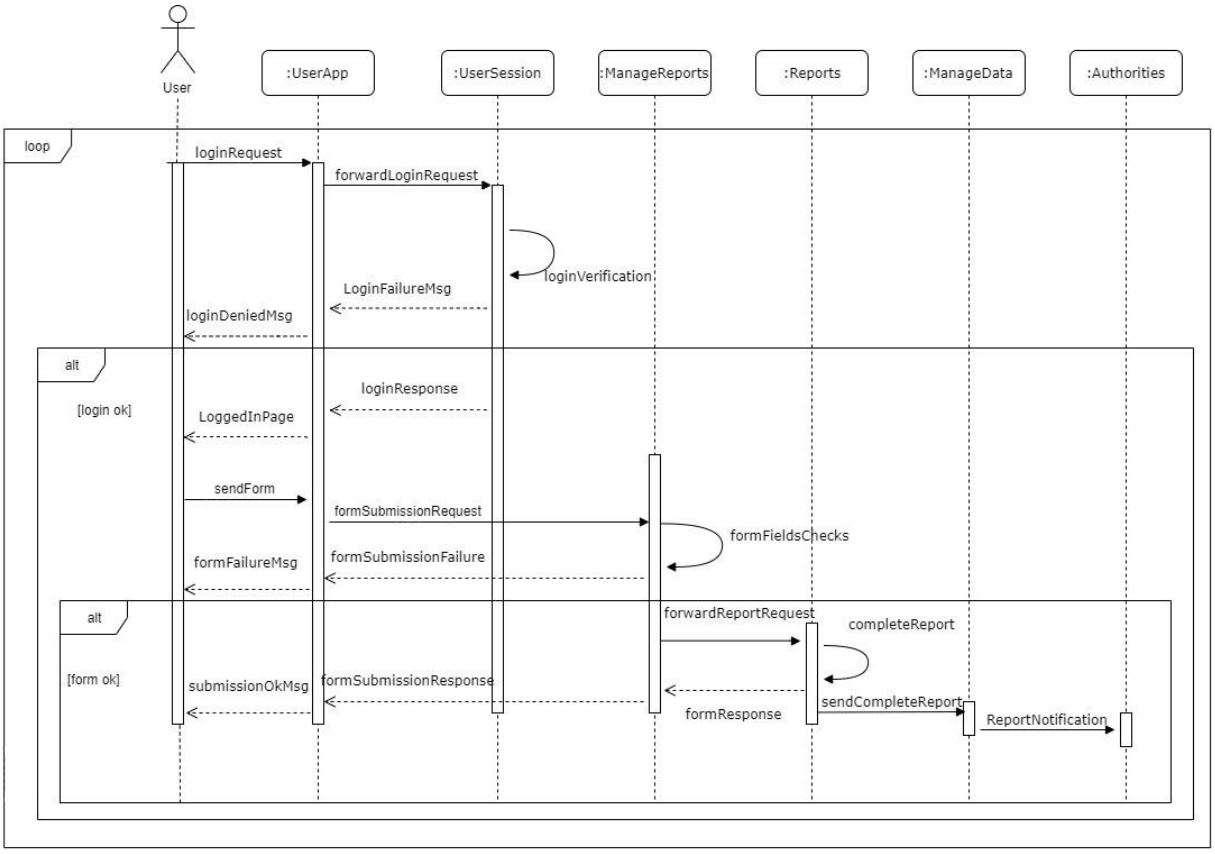
2.4.1. Processes and threads (runtime units)

The following picture shows how runtime units are supposed to behave, so the interactions among processes and threads.

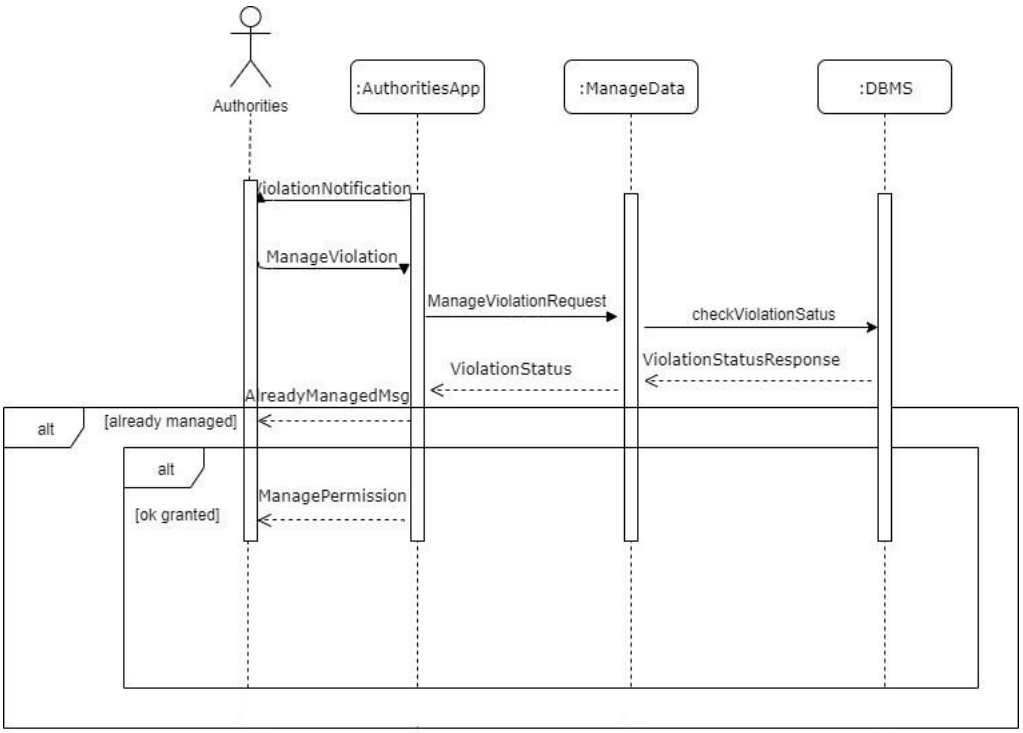


2.4.2. Sequence Diagram

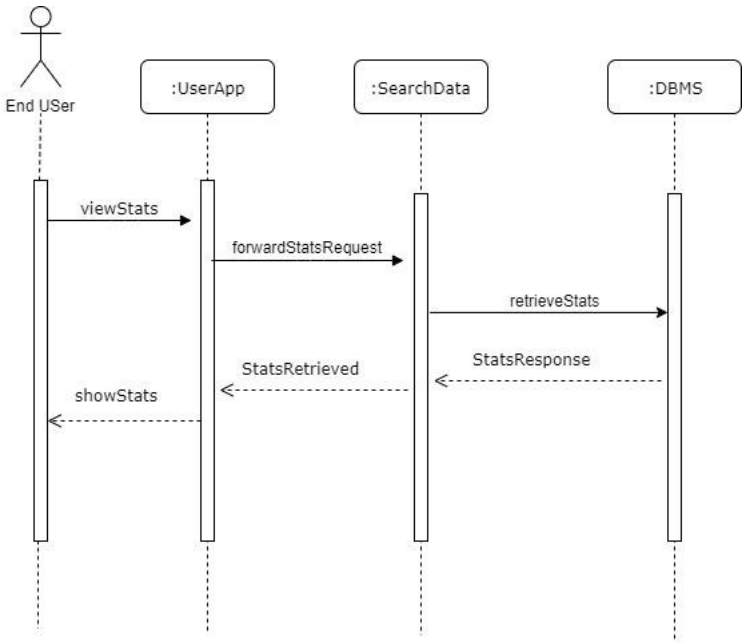
2.4.2.1 Report Violation



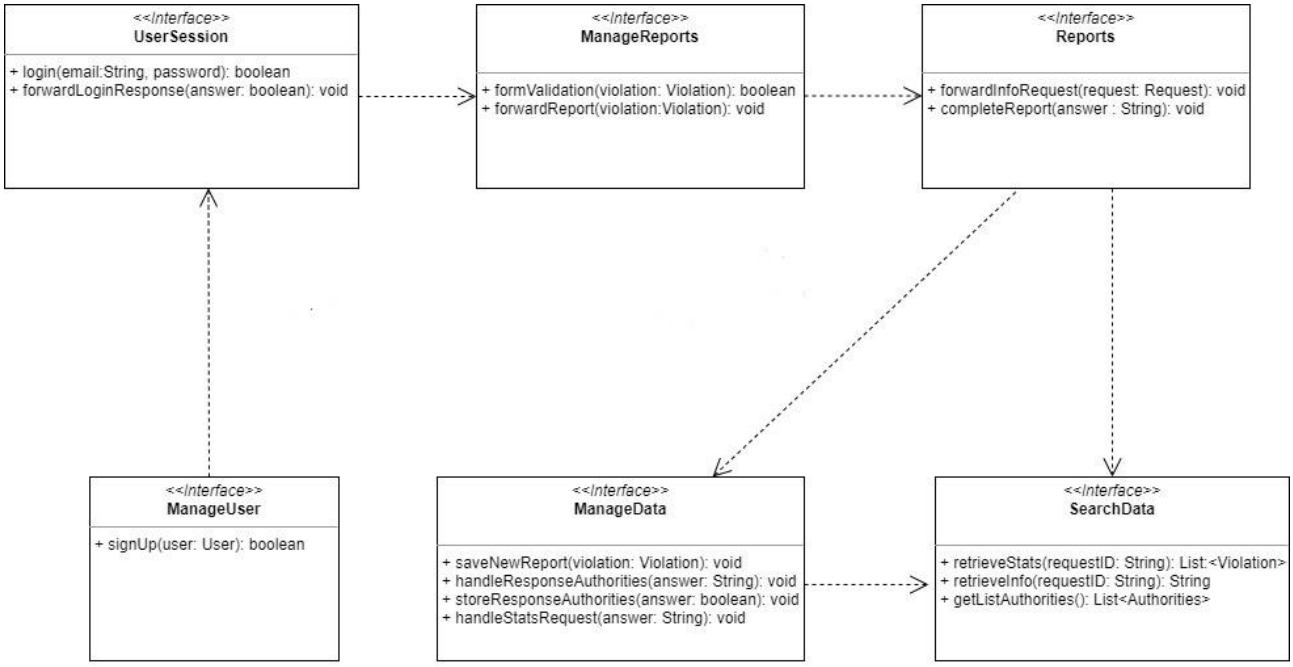
2.4.2.2. Receive Violation Notification



2.4.2.3. Consult Stats



2.5. Component interfaces



2.6. Selected architectural style and patterns

2.6.1 Three Tier Client-Server

The Three Tier Client-Server architecture is a n-tier architecture i.e. functionalities are logically and physically separated in many tiers that communicate with each other. Here the logic layers are: Presentation(P) tier that is concerned with user interface, Application(A) tier that contains the business logic and Data access(D) which deals with data persistence management. This approach makes the system flexible, reusable, safe because of the separation between the access to data and user interface that includes interaction with the customer. Moreover a low coupling is granted by the moment layers are separated and logic and implementation of each layer is independent from the others.

2.7. Other design decisions

2.7.1. Non-relational database

A non-relation database seems to better fit system needs as long as there are not several transactions, neither the necessity to create complex data structures, but gives the system flexibility, scalability and high performance.

3. Requirements tracebility

This section aims at showing how components system meet requirements.

Requirement	*Component*
The system stores the information provided by user	ManageData
The system complete information with type of violation	Reports
The system reports violation to authorities	ManageData
The system has to allow end users and authorities to retrieve stats	SearchData

4. Implementation, integration and test plan

This system includes a bunch of subsystems:

- UserApp
- AuthoritiesApp
- Application Server
- External systems: GPS

For the implementation, integration and test plan a bottom-up approach is exploited, taking into account the order of importance of the different functionalities.

The components of the same subsystem will be implemented, integrated and tested for each subsystem and will take place in a later stage.

Feature	Importance for the customer	Difficulty of implementation
Sign up and login	*low*	*low*
Visualize personal data and stats	*high*	*low*
Report violations	*high*	*high*
Violation reported notified to authorities	*medium*	*medium*

The previous table gives the order of importance to features:

- Report violations
- Visualize personal data and stats
- Violation reported notified to authorities
- Sign up and login