

SafeStreets

Software Engineering II Project



POLITECNICO DI MILANO

December 20, 2019

Authors:

Matteo PACCIANI
Francesco PIRO

Professor:

Matteo Giovanni ROSSI

RASD

Goals

Boundaries of the system

Meaningful Use-Cases

Relevant Requirements

Relevant Assumptions

Alloy Model

DD

Component View

Components Interfaces

Runtime View

Architectural Styles and Patterns

Deployment View

Implementation Integration and Test Plan



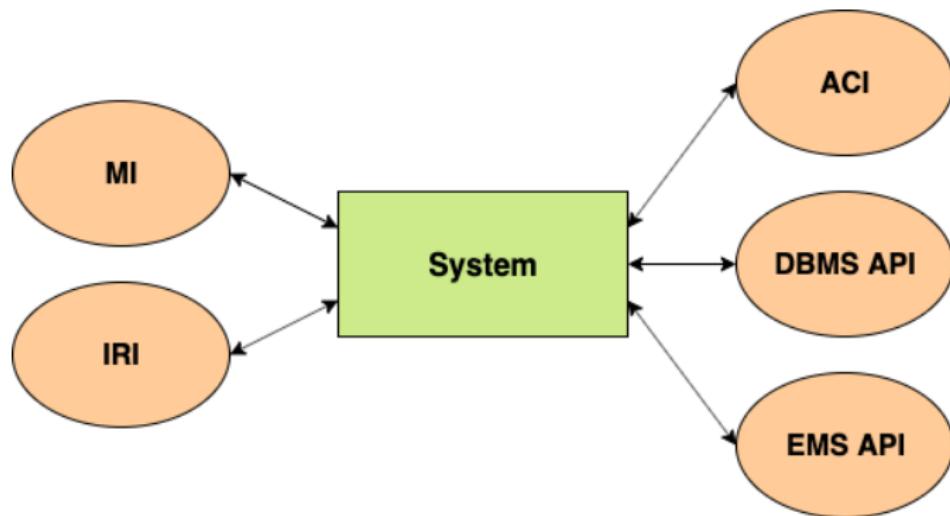
RASD

Requirements Analysis and Specification Document

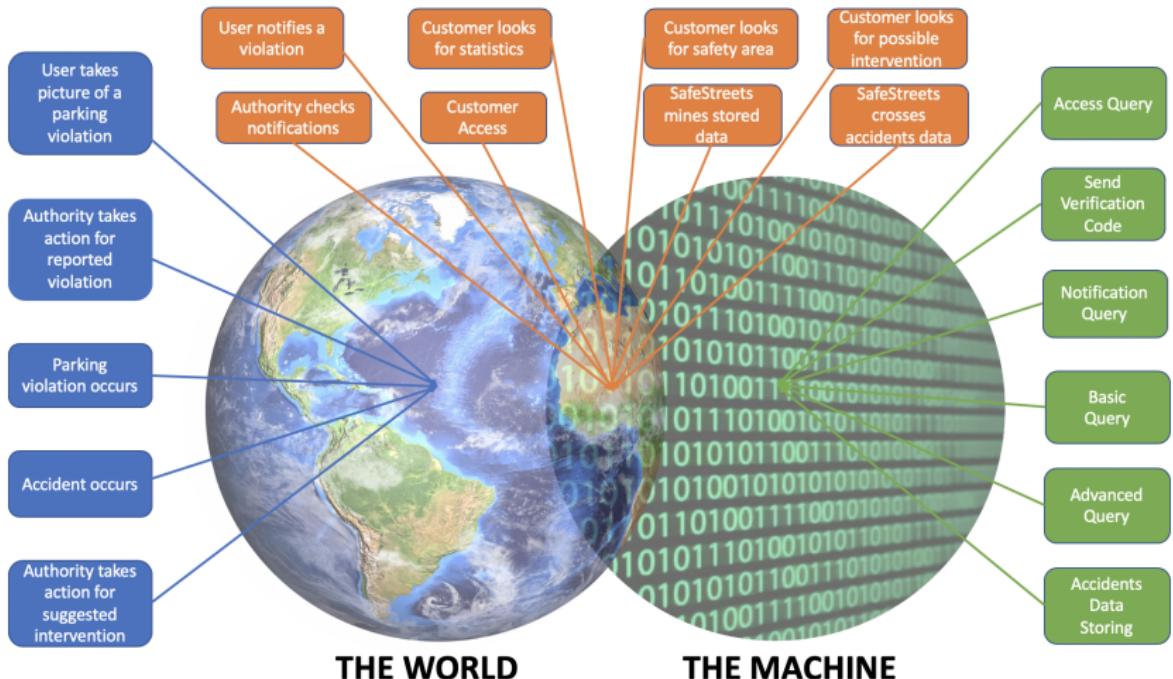
- G1** Users should be able to notify authorities when traffic violations occur, in particular parking violations.
- G2** Users and authorities should be able to mine the information stored by SafeStreets, with different levels of visibility.
 - G2A** Users and authorities should be able to know where the highest number of violations occur.
 - G2B** Users and authorities should be able to know what types of vehicle make the most violations.
 - G2C** Authorities should be able to consult every violation report sent by users.
- G3** Users and authorities should be able to know which streets are safe and which ones are not.
- G4** Users and authorities should be able to know the possible interventions that could be done in a city.



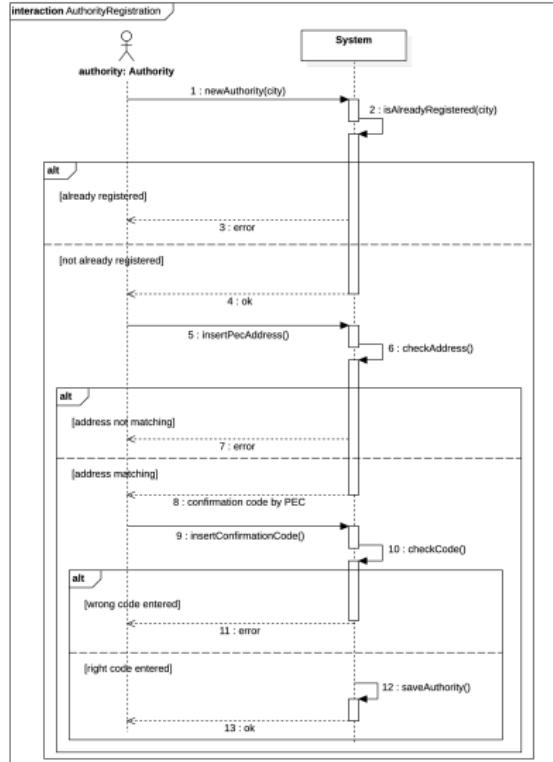
We want now to define the boundaries of our system first with the external interfaces it has to interact with, second with the *World And Machine* that helps to highlight the phenomena.



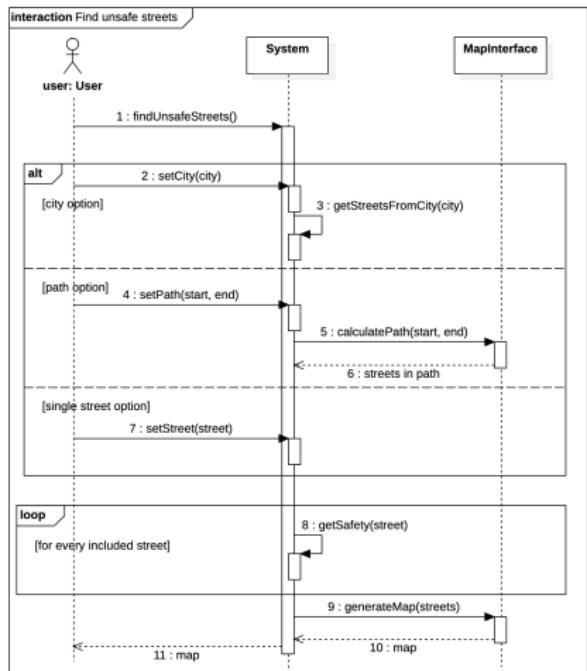
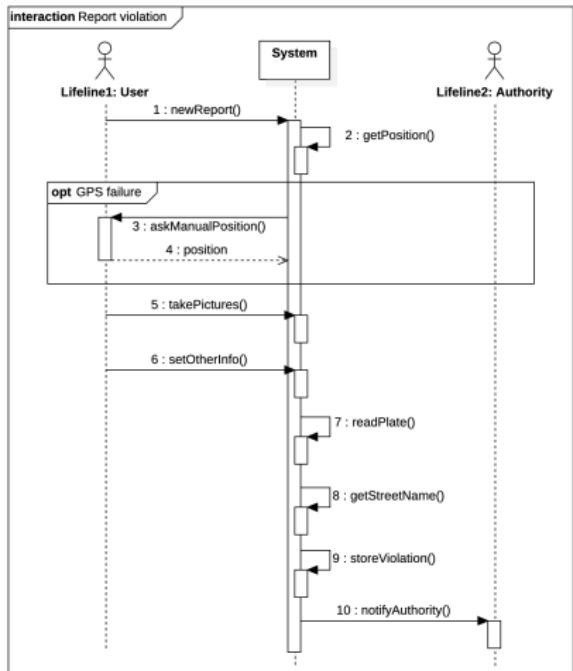
The World and the Machine



Use Case and Authority Registration



Report Violation and Unsafe Streets



R8 The system must allow users to take pictures of the violation they want to report

R17 The system must be able to show authorities all the violation reports sent by users

R18 The system must be able to mine the stored violation reports

R31 The system must be able to determine whether a street is safe or not

R35 The system must be able to determine the most urgent interventions in a street.



DA2 Users always insert correct data while reporting violations to SafeStreets

DA7 The GPS module of the devices on which SafeStreets runs always works correctly and has an accuracy of 2 meters

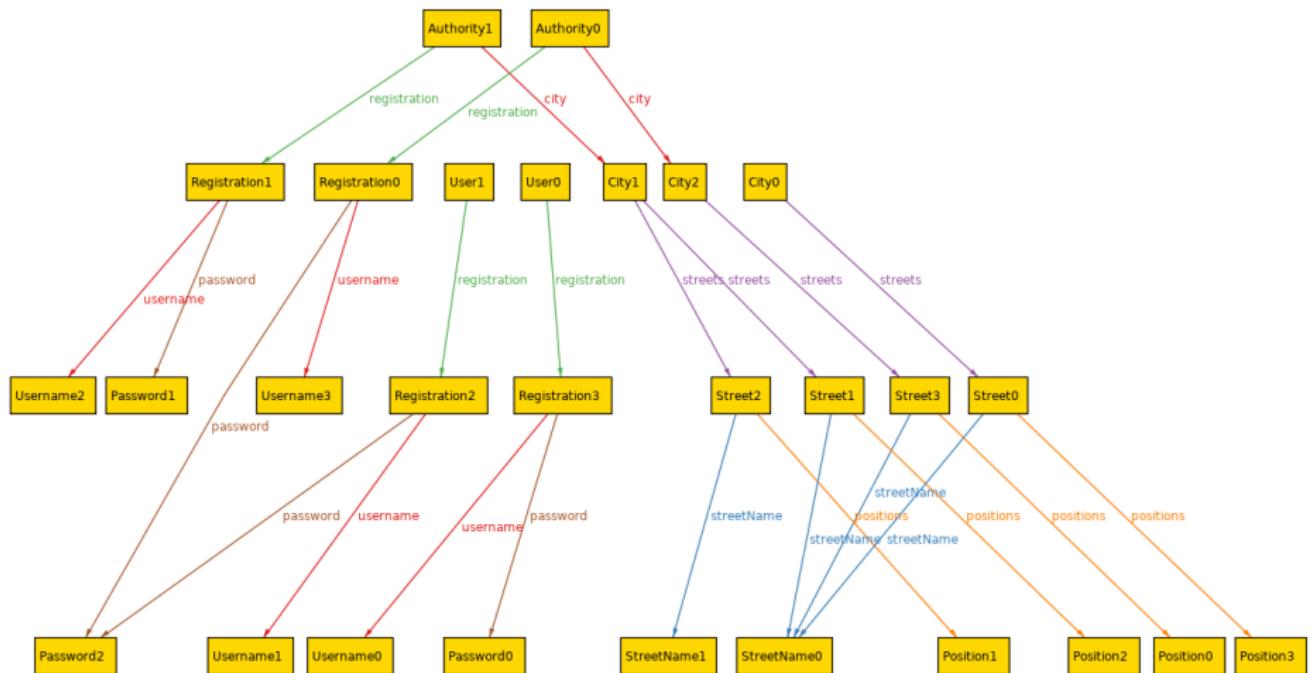
DA8 The camera module of the devices on which SafeStreets runs always works correctly

DA10 The system is assumed to work in Italy

DA13 No one physically and maliciously replaces license plates

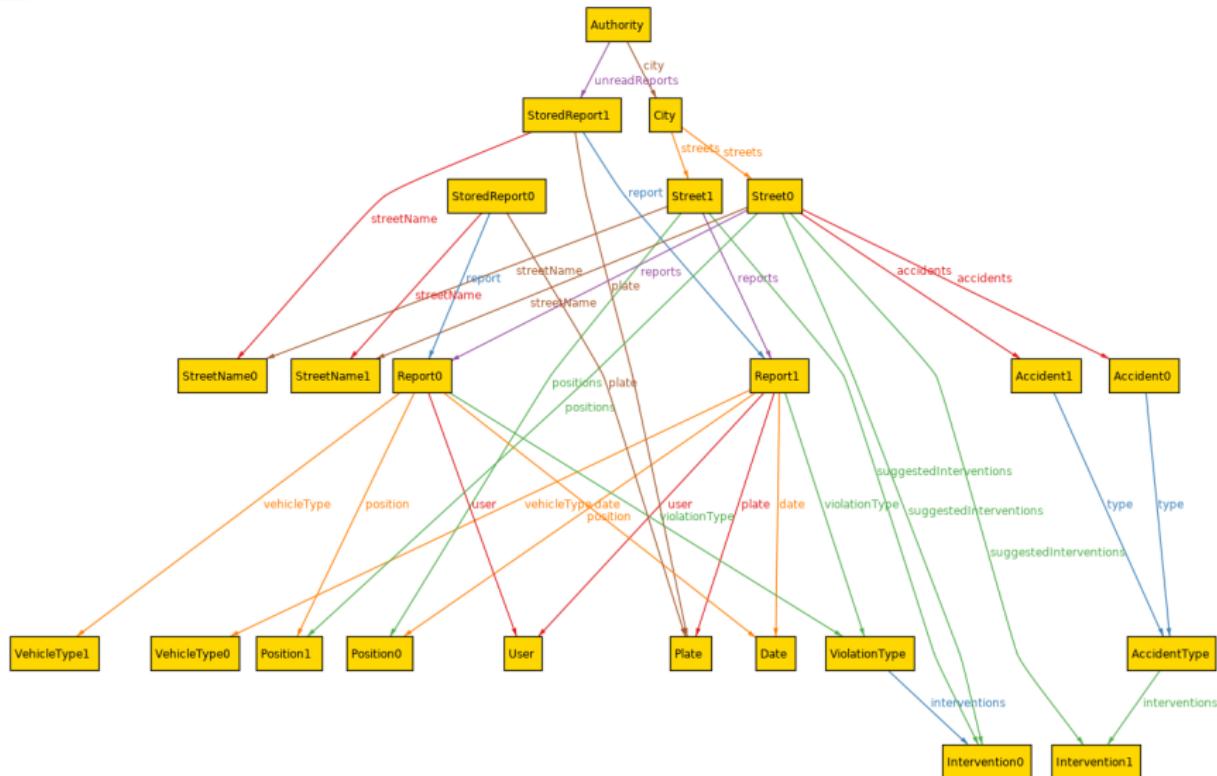
DA17 Accidents data provided by municipalities are always correct





Second World

12

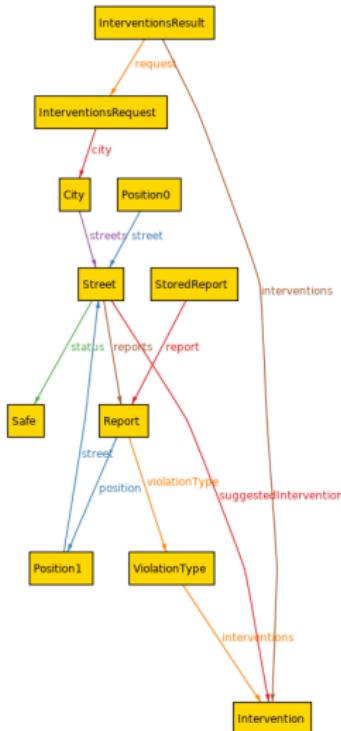


```
147 //an intervention is suggested for a street iff the number
    of reports in
148 //that street that contain violation types linked to that
    intervention
149 //is greater than 0 OR the number of accidents in that
    street linked to
150 //that intervention is greater than 0
151 fact InterventionSuggestedThreshold{
152     all s: Street, i:Intervention| (i in s.
        suggestedInterventions iff
        (#getReportsFromInterventionAndStreet[i, s] > 0 or
        #getAccidentsFromInterventionAndStreet[i, s] > 0))
153 }
154
155
156
157 //gets all the reports for that street, having as violation
    type one linked to the
158 //given intervention
159 fun getReportsFromInterventionAndStreet[i : Intervention, s
    : Street] : set Report{
160     {
161         r : Report | i in r.violationType.interventions and r in
            s.reports
162     }
163 }
```



```
177 --a street is considered unsafe iff
178 --the number of total accidents in that street is greater
179     than 1 OR
180 --if the number of total violations in that street is
181     greater than 1
182
183 fact UnsafeStreetThreshold{
184     all s: Street | s.status = Unsafe iff (
185         #s.reports > 1 or
186         #s.accidents > 1
187     )
188 }
```





```

261 //this is the rule to apply the filter that comes with the
262 //InterventionsRequest
263 //only those interventions that satisfy the following
264 //conditions will be included
265 //in the result
266 fact InterventionsResultRule{
267   all i:Intervention, result:InterventionsResult | i in
268     result.interventions iff
269   (
270     some s:Street | s in result.request.city.streets and
271     i in s.suggestedInterventions
272   )
273 }
  
```

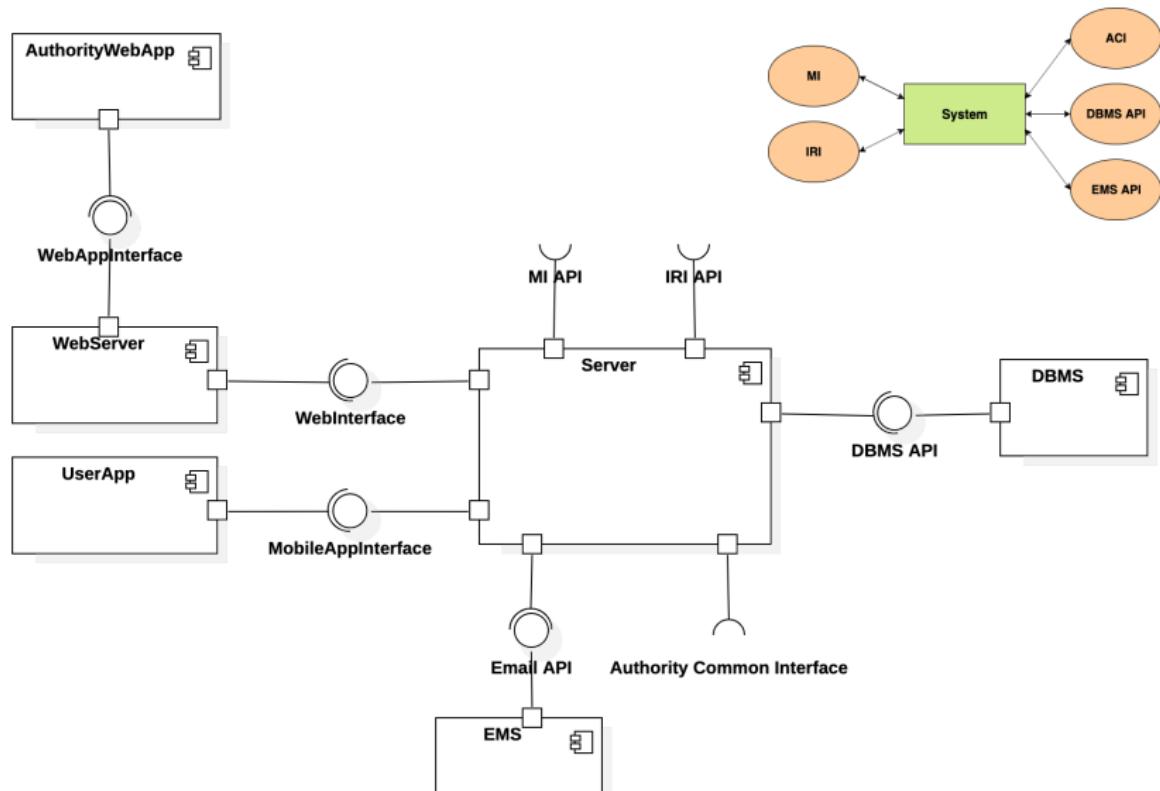
```

299 //this is the rule to apply the filter that comes with the
300 //ReportsRequest
301 //only those reports that satisfy these conditions will be
302 //included in the result
303 fact ReportsResultRule{
304   all sr:StoredReport, result:ReportsResult | sr in result.
305     storedReports iff
306   (
307     sr.report.violationType in result.request.violationTypes
308     and
309     sr.report.date in result.request.dates and
310     sr.report.time in result.request.time and
311     sr.report.vehicleType in result.request.vehicleTypes and
312     (result.request.street = none implies sr.report.position.
313       street.city in
314         result.request.authority.city else sr.report.position.
315       street in
316         result.request.street
317     )
318   )
319 }
  
```



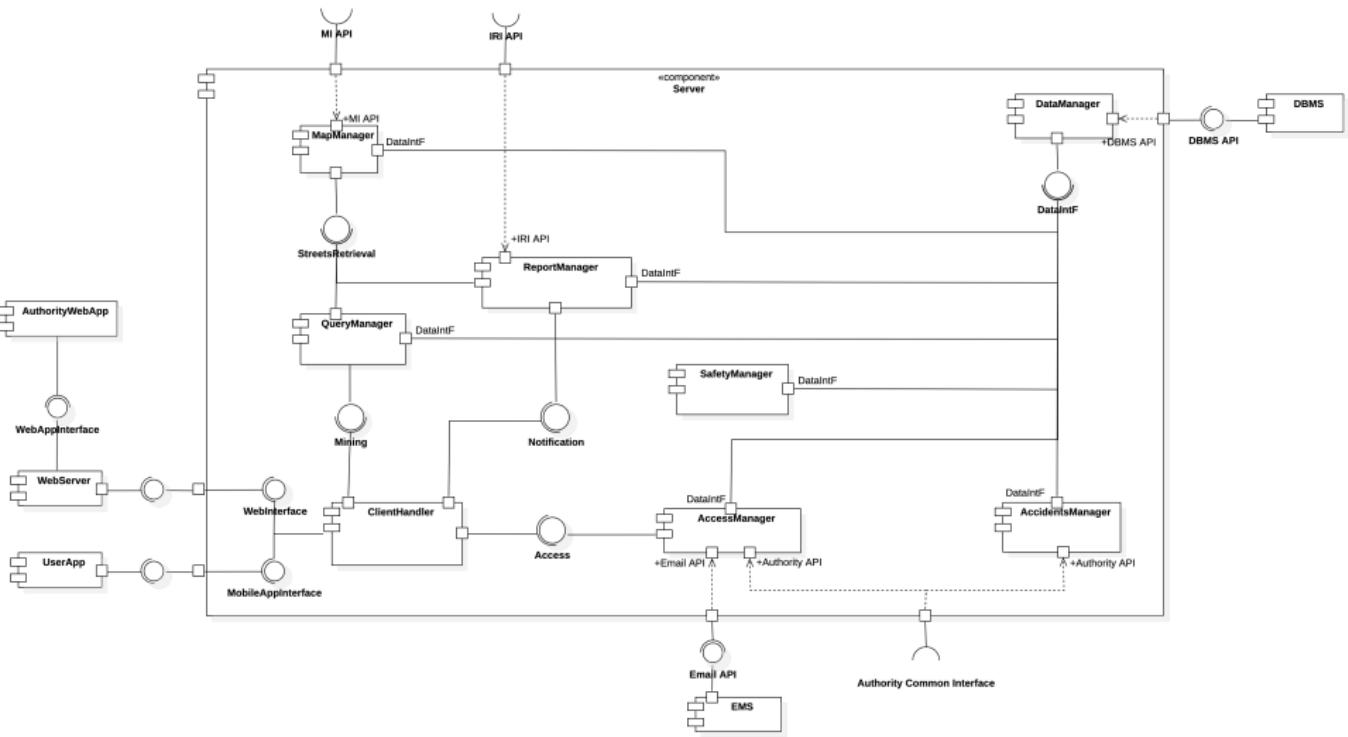
DD

Design Document

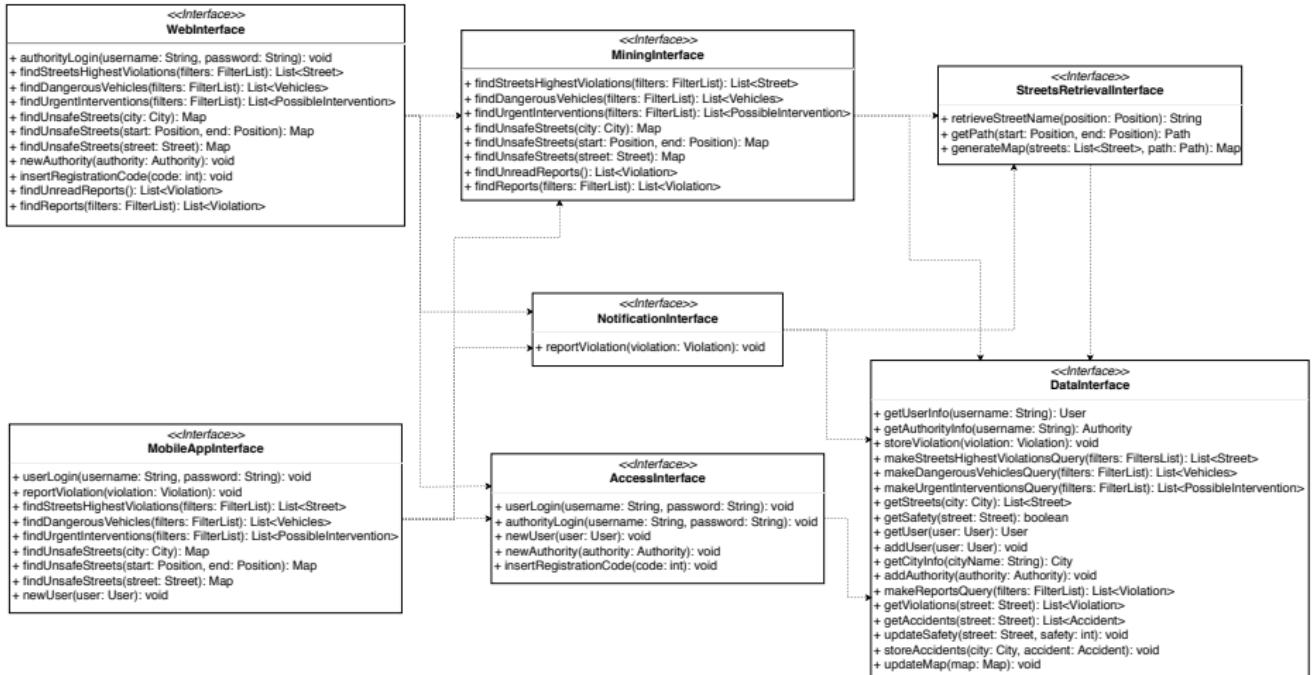


Server Component

18

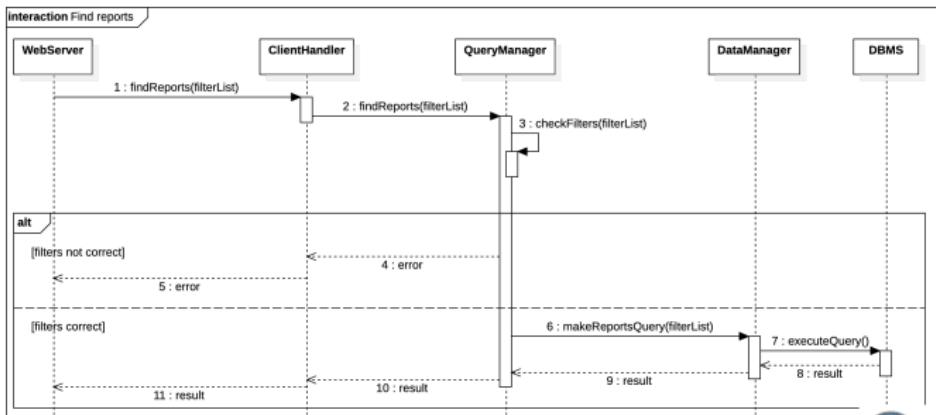
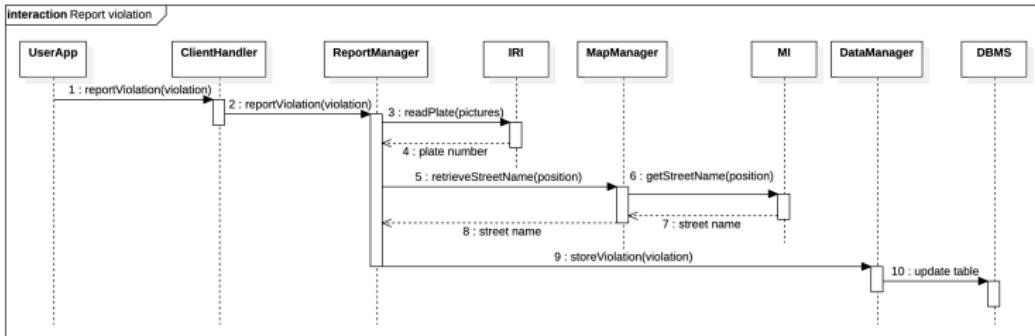


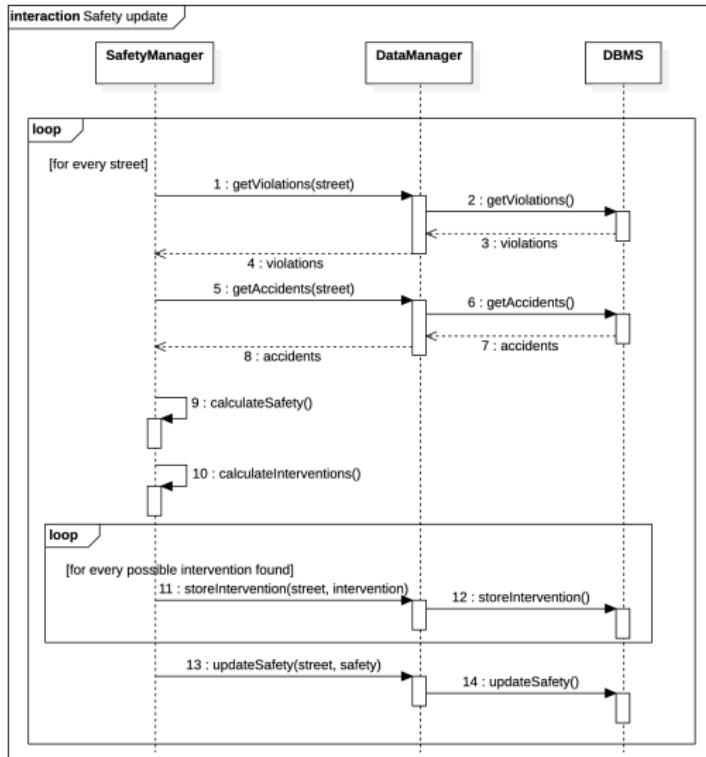
Component Interfaces



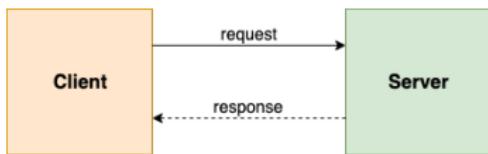
Report Violation and Find Reports

20

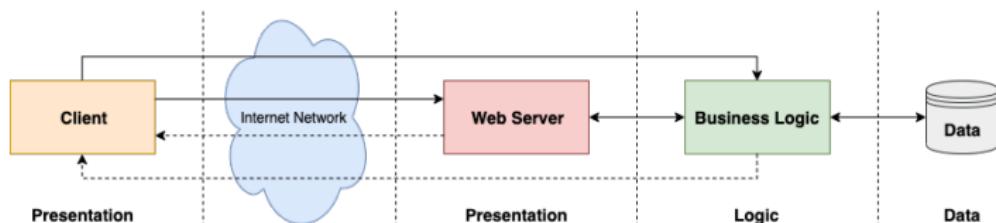




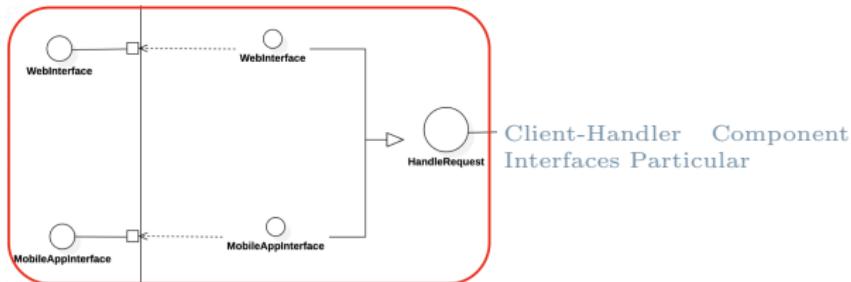
The system is a **crowd-sourced** application based on the client-server paradigm where the *users* represent the crowd on which the functionalities of SafeStreets are funded



The layers of the system are deployed to a *four-tier architecture*



SafeStreets provides its services through **REST** thanks to an interface that is extended twice to define the communication with the **mobile app** for the users and the **web app** for the authorities

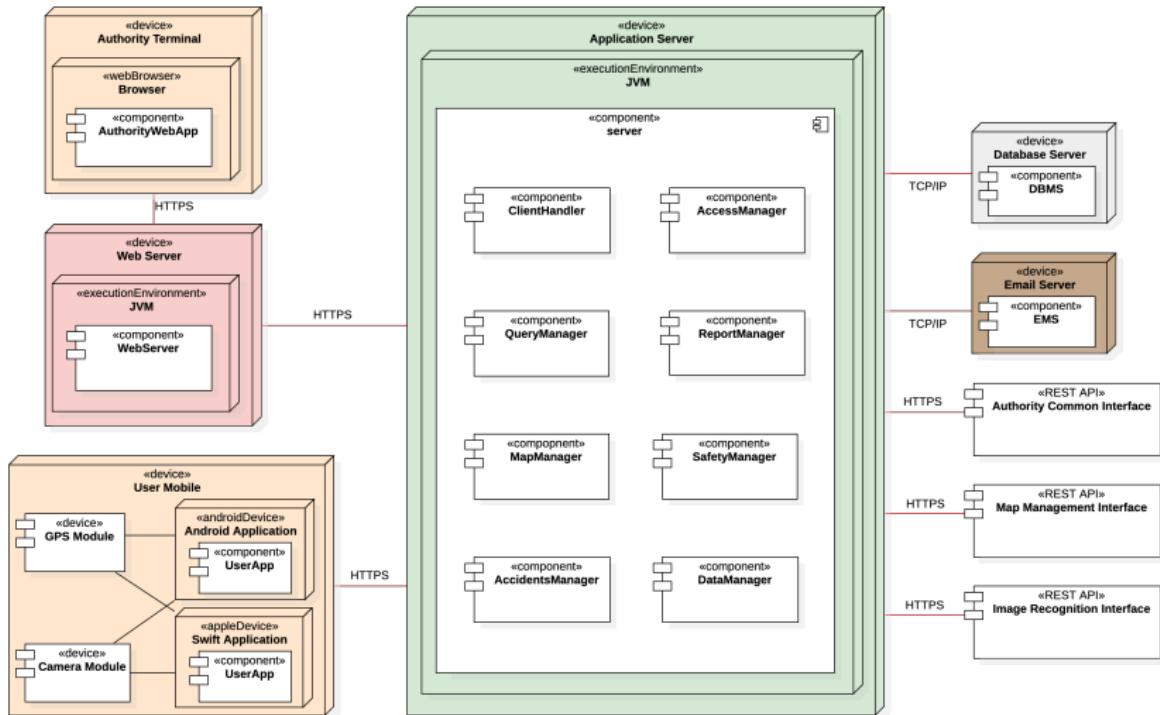


Moreover basic communication with the TCP/IP protocol is also used in particular for the external components that allows the system to manage its data: the *DBMS* and the *EMS*



Deployment View

24



The strategy on which the *Implementation, Integration and Test Plan* is developed, and thus the use relation hierarchy diagram, is based on the following 3 main features:

1. **Bottom-Up and Top-Down approach**



The strategy on which the *Implementation, Integration and Test Plan* is developed, and thus the use relation hierarchy diagram, is based on the following 3 main features:

1. **Bottom-Up and Top-Down approach**
2. **Core Functionality**

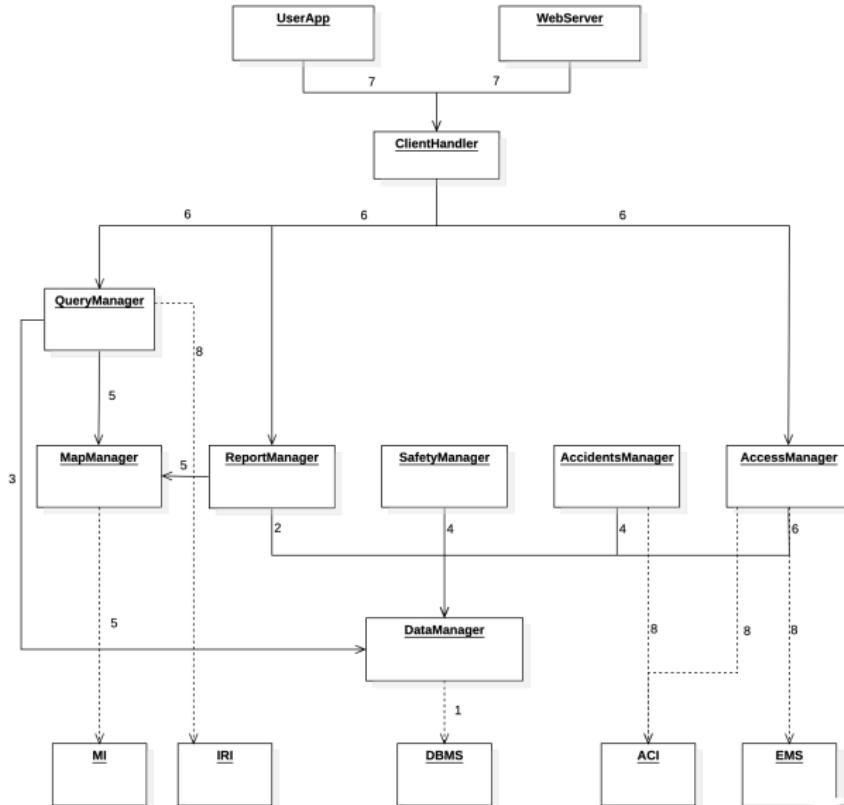


The strategy on which the *Implementation, Integration and Test Plan* is developed, and thus the use relation hierarchy diagram, is based on the following 3 main features:

1. **Bottom-Up and Top-Down approach**
2. **Core Functionality**
3. **Reliability of Eternal Systems**



Use Relation Hierarchy Diagram



Thanks for your Attention
