POLITECNICO
MILANO 1863

SafeStreets

Academic year 2019-2020

Authors:
Marcer Andrea - 941276
Marchisciana Matteo - 945878
Motta Dennis - 940064

Professor:
Rossi Matteo

RASD

Goals

[G1] A citizen can report a violation.

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

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

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

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

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

[G7] Municipality can visualize statistics about the violations in its territory and the effectiveness of the SafeStreets initiative.

Stakeholders’ needs

	Goals						
	G1	**G2**	**G3**	**G4**	**G5**	**G6**	**G7**
Basic service	X	X	X				
Advanced function 1			X	X			
Advanced function 2			X		X	X	X

Domain assumptions

[D1] A citizen who wishes to report a violation has a mobile phone with the SafeStreets app installed.

[D2] Municipality offers a service to retrieve information about accidents.

[D3] Municipality offers a service to retrieve information about tickets.

[D4] When a device is able to obtain a GPS fix, the location provided has an accuracy of at least 20 meters.

[D5] The municipality checks if approved violation reports can actually represent a traffic violation.

[D6] Data transferred through connections that use modern encryption protocols can not be manipulated.

Requirements

[R13] The system must analyze valid violations report and approve which of them may represent a correct violation.

[R14] The system must be able to elaborate data about violations, accidents, issued tickets and generate useful suggestions about possible interventions.

[R16] The system must offer a service to the municipality for retrieving approved violations report.

[R18] All connections used by the system use modern encryption protocols.

Use case 3: Report a violation, part 1/2

Name	Report a violation
Actor	User
Entry condition	The user is logged in.
Event flow	1. The user chooses the option to report a new violation. 2. The user takes at least one photograph of the violation within the application. 3. The user writes the description. 4. The user selects the type of violation from a given list. 5. The user presses the “Send” button. 6. The system retrieves additional information about the report. 7. The system saves the report data. 8. The system retrieves and checks the consistency of the meta-information. 9. It is communicated to the user that the report has been received correctly.
Exit condition	The violation is saved in the system as “submitted” and it’s ready to be reviewed.

Use case 3: Report violation, part 2/2

Name	Report a violation
Actor	User
Entry condition	The user is logged in.
Exceptions	1. The application cannot retrieve the license plate from one of the photos: the application asks the user to insert it manually. 2. Some of the meta-information of the violation report are incorrect: the user is notified and is asked to correct them. 3. The application cannot access the camera: the user is notified about the problem and the submission of the violation is denied. 4. The application cannot access the GPS location: the user is notified about the problem and the submission of the violation is denied.
Exit condition	The violation is saved in the system as “submitted” and it’s ready to be reviewed.

Use case 6: Verify Report

Name	Verify reports
Actor	Police officer
Entry condition	The police officer is logged in.
Event flow	1. The police officer chooses the option to verify the violations. 2. The system shows on the screen a list of approved violation reports. 3. The police officer chooses a violation report to verify. 4. The system shows in detail all the information of the violation report. 5. The police officer checks the correctness of the violation type, the license plate and the car model. 6. The police officer presses the “Confirm” or “Reject” button accordingly to what he/she has found.
Exit condition	The report is labeled accordingly to what the police officer has chosen.

Alloy

Goal 2: A violation report received by the system must have enough information to be valid: the report, once created, will have **at least one picture**, the **location**, **timestamp**, **type of the violation** and **license plate**.

Goal 6: The **integrity** of the violation report is guaranteed. This means the report, since its creation, will never be in a state in which its integrity can be compromised.

Alloy: signatures

```
sig ViolationReport {
        pictures: set Picture,
        location: lone Location,
        timestamp: lone Timestamp,
        typeOfViolation: lone TypeOfViolation,
        licensePlate: lone LicensePlate,
        state: one ViolationReportLocation,
        createdBy: one Device,
        canBeAltered: one Bool
}
```

```
sig Device {
        hasGPS: one Bool,
        hasInternet: one Bool,
        hasCamera: one Bool
}
```

Alloy: goal 2, part 1/2

```
/*[R4] The application must allow reporting of violations only from devices equipped
 * with a GPS receiver which are in the conditions to obtain a GPS fix. */
[...]

/* R5 and R6 express the same thing but with the camera and internet connection */
[...]

/*[R7] A user has the possibility to specify the type of the reported violation choosing from a list.
 *[R8] The application creates a violation report with at least one picture, exactly one timestamp,
 *    exactly one location, exactly one type of violation and the license plate of the vehicle.*/
fact requirement8 {
      //A valid report is created only if the device has the GPS, a camera and internet
      all v : ViolationReport | (v.state = ON_DEVICE and v.createdBy.hasGPS = TRUE and
                                       v.createdBy.hasInternet = TRUE and v.createdBy.hasCamera = TRUE)
      implies
      (#v.pictures >= 1 and
      #v.location = 1 and
      #v.timestamp = 1 and
      #v.typeOfViolation = 1 and
      #v.licensePlate = 1)
}
```

Alloy: goal 2, part 2/2

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

assert goal2 {
      all v : ViolationReport | v.state = ON_SERVER implies
            (#v.pictures >= 1 and
            #v.location = 1 and
            #v.timestamp = 1 and
            #v.typeOfViolation = 1 and
            #v.licensePlate = 1)
}
```

Alloy: goal 6, part 1/2

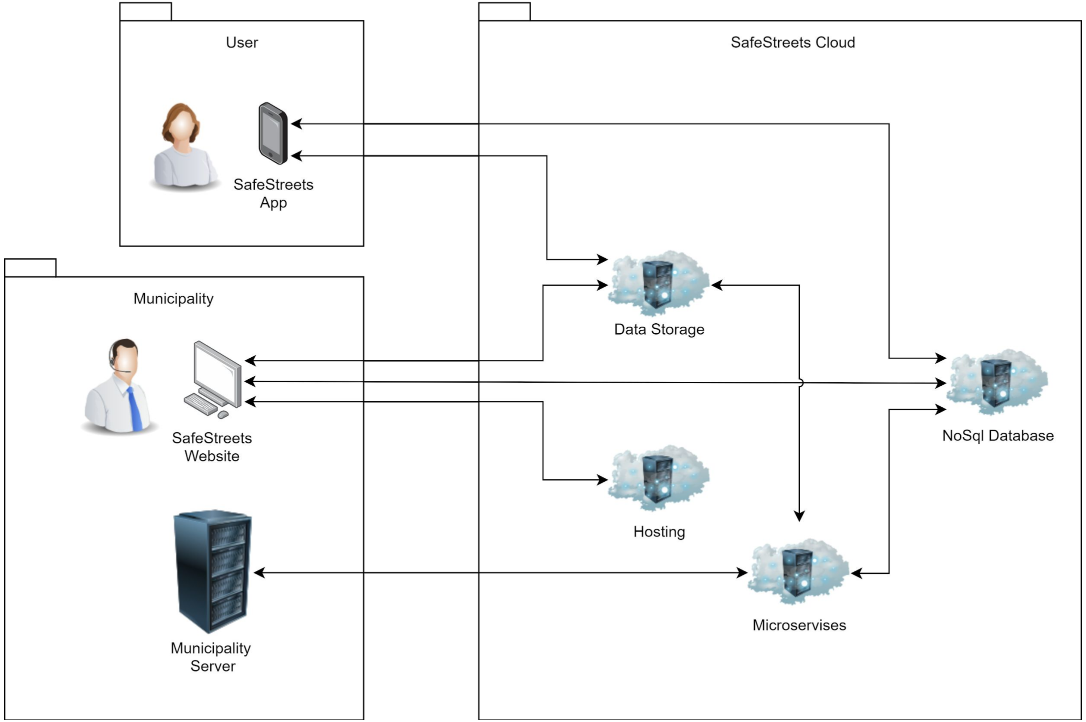
```
/*[D6] Data transferred through connections that use modern encryption protocols
 *cannot be manipulated.*/
fact domainAssumption6 {
	all v : ViolationReport | v.state = ON_NETWORK_ENCRYPTED implies v.canBeAltered = FALSE
	all v : ViolationReport | v.state = ON_NETWORK_NOT_ENCRYPTED implies v.canBeAltered = TRUE
}

/*[R17] The application will allow using pictures in a violation report only if the picture
 * was taken by the application itself, preventing it to be manipulated on the device.*/
[...]

/*[R18] All connections used by the system use modern encryption protocols.*/
fact requirement18 {
	all v : ViolationReport | v.state != ON_NETWORK_NOT_ENCRYPTED
}

/*[R19] Data saved in the server can not be manipulated.*/
[...]
```

Alloy: goal 6, part 2/2

```
// ########## Modelling the network ##########

pred sendReportToNetwork [vDevice : ViolationReport, vNetwork : ViolationReport] { [...] }
pred receiveReportFromNetwork [vNetwork : ViolationReport, vServer : ViolationReport] { [...] }
pred sendReportToServerFromDevice [vDevice: ViolationReport, vServer : ViolationReport] { [...] }


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

assert goal6 {
       all v : ViolationReport | v.canBeAltered = FALSE
}
```

DD

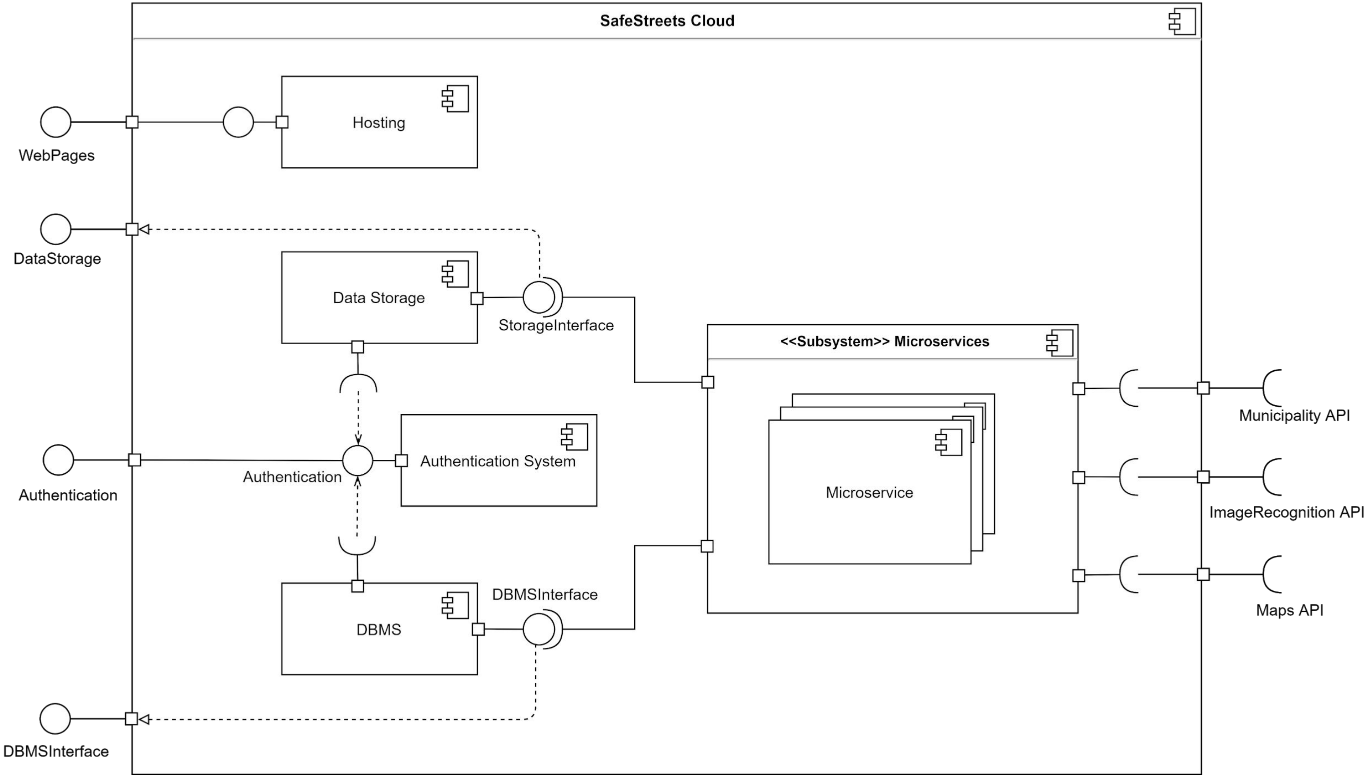
High level architecture view



Subsystems:

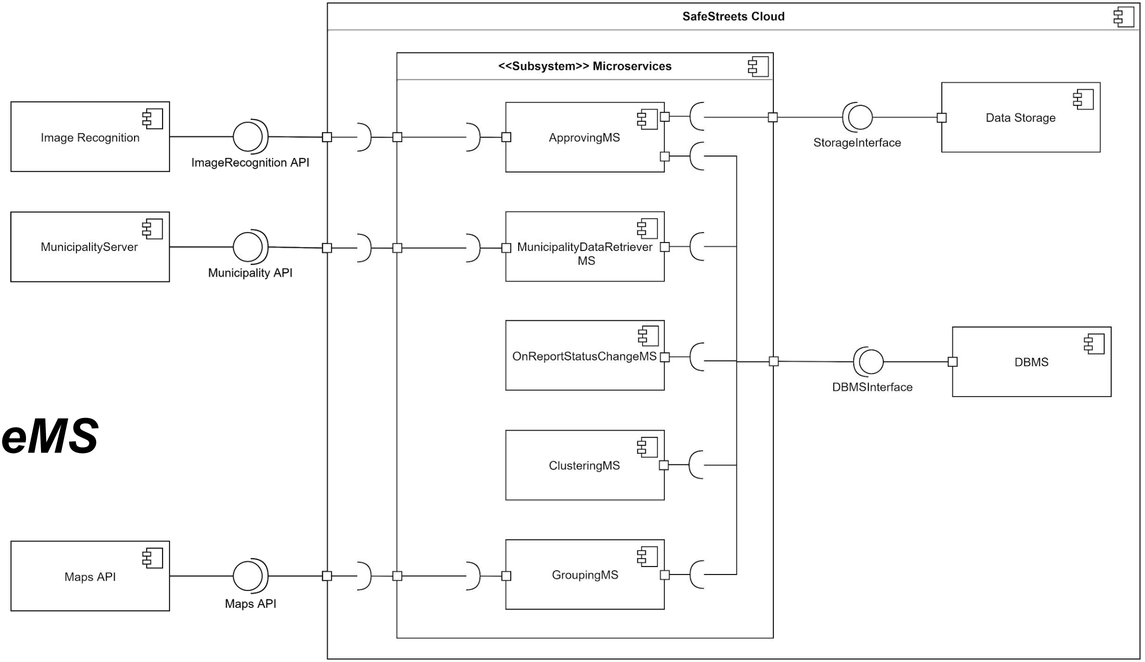
- SafeStreets Cloud
- SafeStreets App
- SafeStreets Web

Cloud component diagram



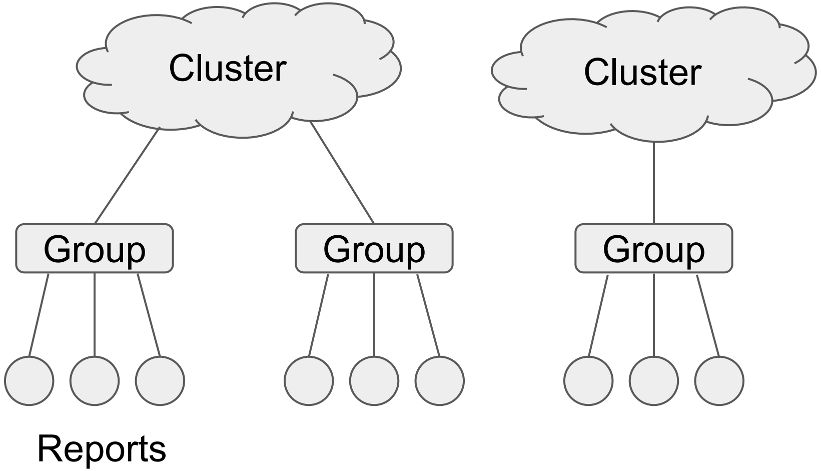
What do Microservices do?

- ***ApprovingMS***
- ***GroupingMS***
- ***ClusteringMS***
- ***OnReportStatusChangeMS***

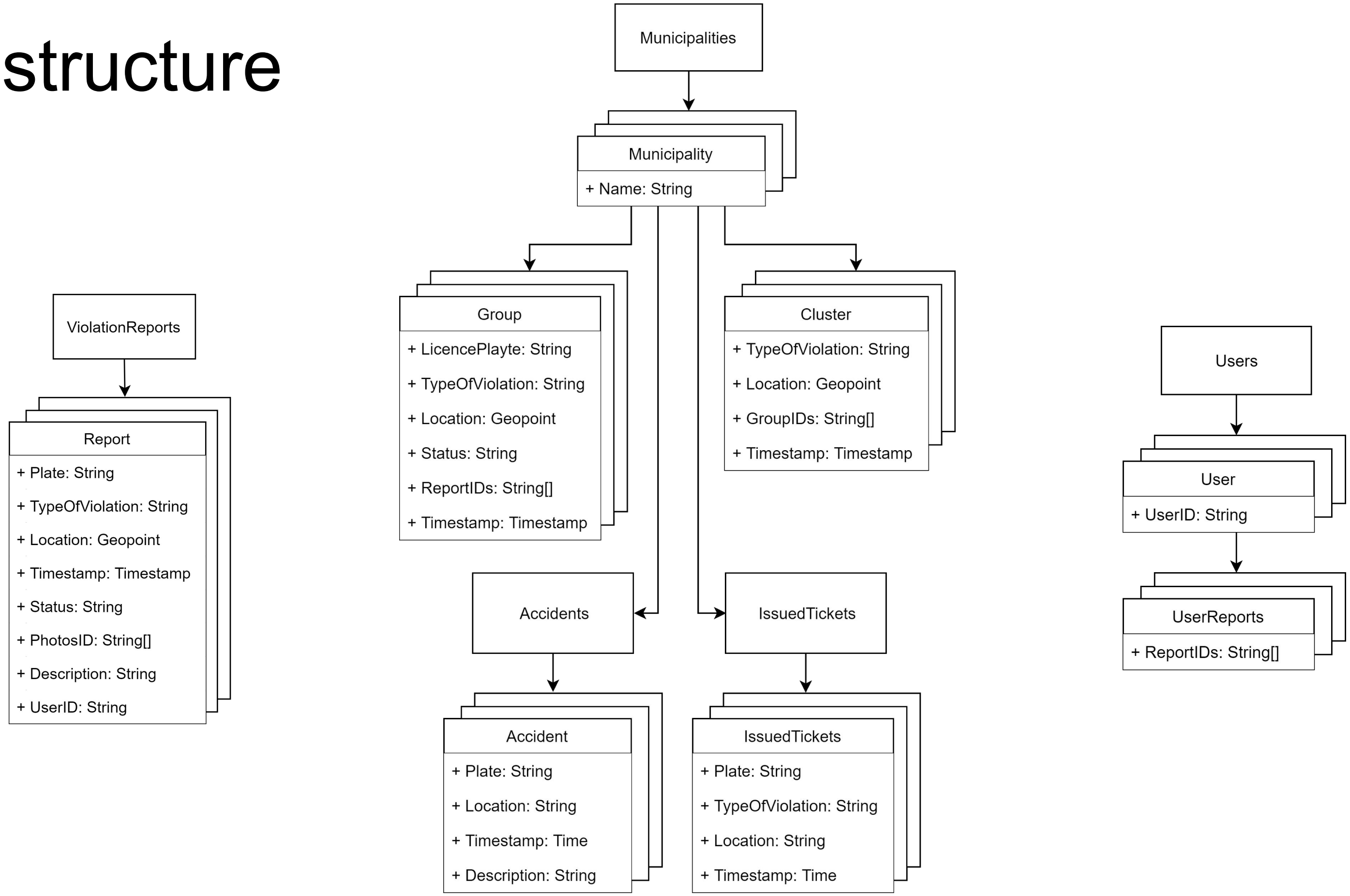


What do Microservices do?

- ***ApprovingMS***
- ***GroupingMS***
- ***ClusteringMS***
- ***OnReportStatusChangeMS***



Data structure



Municipalities

Municipality

+ Name: String

Group

+ LicencePlayte: String

+ TypeOfViolation: String

+ Location: Geopoint

+ Status: String

+ ReportIDs: String[]

+ Timestamp: Timestamp

Cluster

+ TypeOfViolation: String

+ Location: Geopoint

+ GroupIDs: String[]

+ Timestamp: Timestamp

Accidents

Accident

+ Plate: String

+ Location: String

+ Timestamp: Time

+ Description: String

IssuedTickets

IssuedTickets

+ Plate: String

+ TypeOfViolation: String

+ Location: String

+ Timestamp: Time

ViolationReports

Report

+ Plate: String

+ TypeOfViolation: String

+ Location: Geopoint

+ Timestamp: Timestamp

+ Status: String

+ PhotosID: String[]

+ Description: String

+ UserID: String

Users

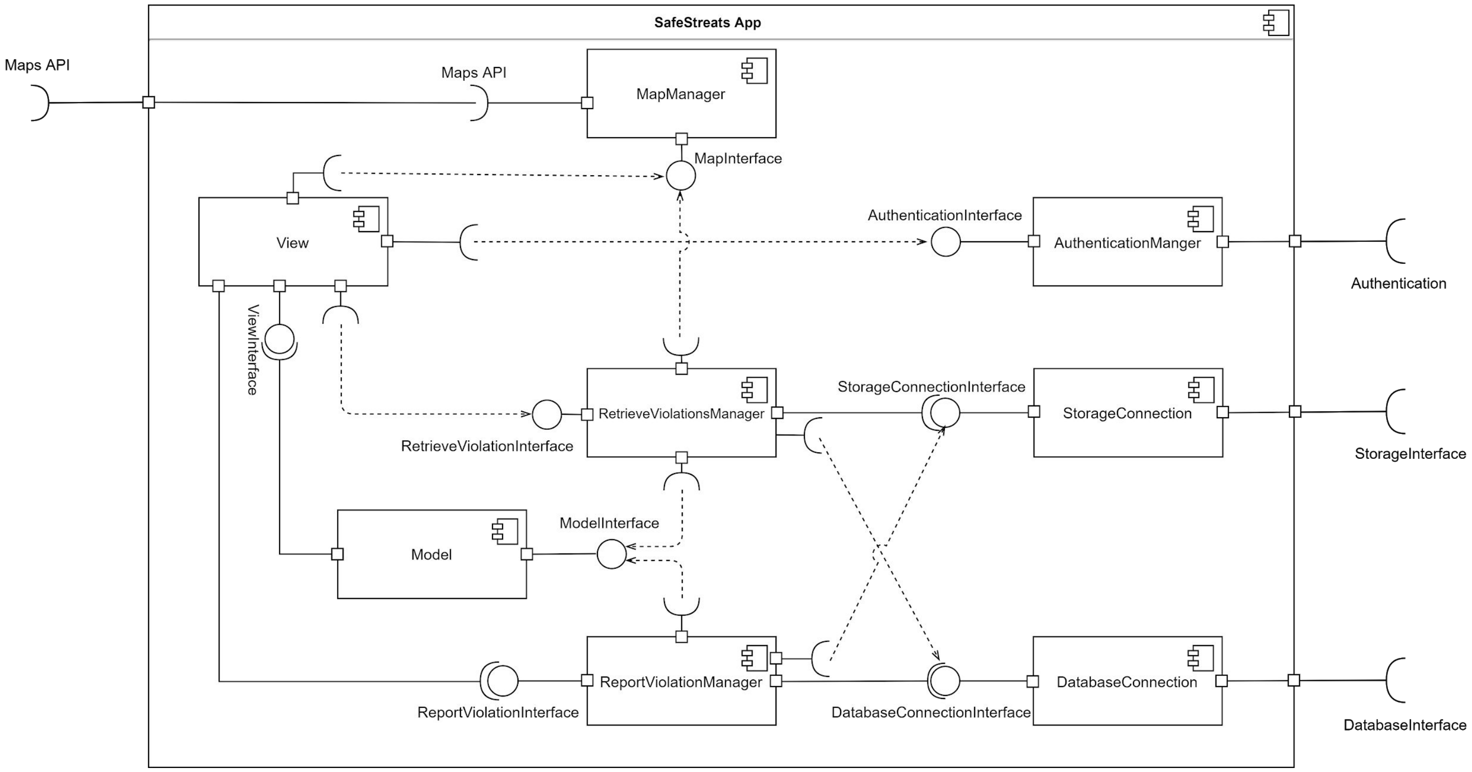
User

+ UserID: String

UserReports

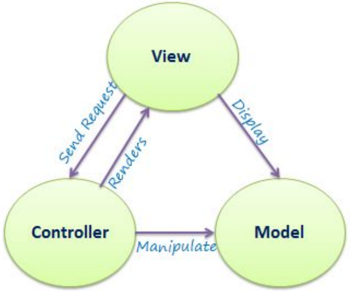
+ ReportIDs: String[]

Application component diagram

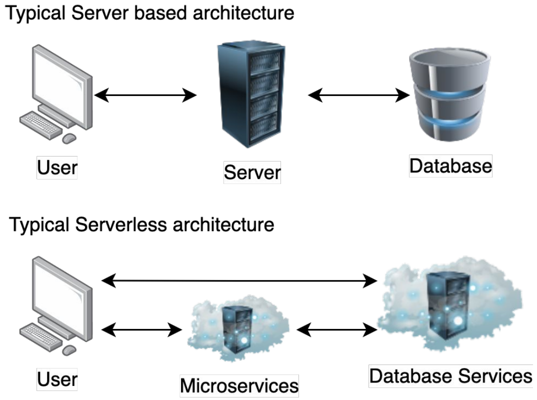


Architectural patterns

- **Model–View–Controller:**



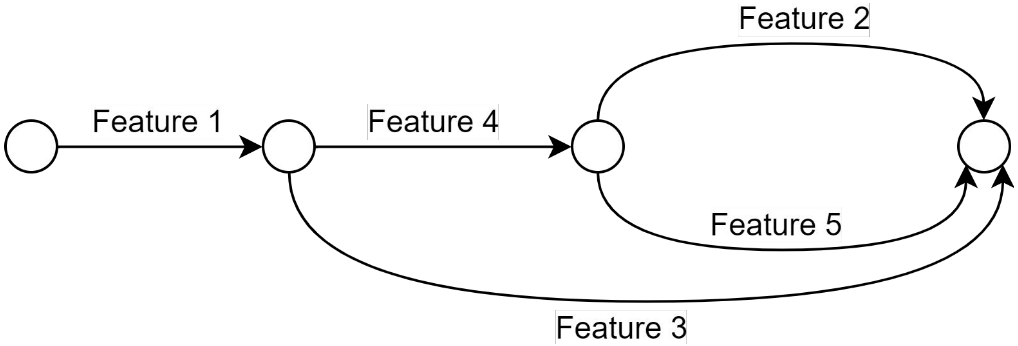
- **Serverless computing:**


Typical Server based architecture
User
Server
Database
Typical Serverless architecture
User
Microservices
Database Services

Implementation and testing plan

Features:

1. Report violations (User)
2. Visualize data (User & Municipality)
3. See own violations (User)
4. Retrieve and review violations (Municipality)
5. Visualize statistics (Municipality)



ITD

Platform and APIs

- **Firebase:**
  - **Authentication**
  - **Hosting**
  - **Storage**
  - **DBMS**
  - **Microservices**
- **Google Maps API**
- **Google Cloud Vision API**
- **Google Geocoding API**

Why Firebase?

Reliability

Availability

Focus on the system’s logic

Portability

Scalability

Elasticity

Code structure for Web & App

Web:

- Index.html
- AcceptViolations.html
- DetailedViolationView.html
- DisplayData.html

App:

- Model
- View
- Controller
- Interfaces
- Util

Code structure for Cloud

Cloud:

- Microservices:
  - ApprovingMS.js
  - GroupingMS.js
  - ClusteringMS.js
  - OnReportStatusChangeMS.js
  - ...

- DBMS: security rules

- Storage: security rules

Testing

- **Website testing:** end-to-end tests
- **Application testing:** end-to-end tests
- **Cloud testing:** unit tests

Website testing

- Correctly retrieving reports
- Accepting or rejecting reports
- Visualizing reports on the map

Application testing

- Constructs a valid report
- Correctly uploads a violation report
- Retrieves own reports
- Visualizes the violations on the map

Testing frameworks

- **ESLint:** static code analysis tool
- **Mocha:** test framework for JavaScript
- **Chai:** assertion library
- **Sinon:** mocking library for JavaScript
- **Nyc:** test coverage analysis

Microservices testing

- **ApprovingMS:**
  - **Number of tests:** 2 (with 4 total assertions)
  - **Line coverage:** 100% (of a total of 115 raw lines)
- **GroupingMS**
  - **Number of tests:** 6 (with 30 total assertions)
  - **Line coverage:** 100% (of a total of 171 raw lines)
- **ClusteringMS**
  - **Number of tests:** 3 (with 15 total assertions)
  - **Line coverage:** 97% (of a total of 140 raw lines)
- **OnReportStatusChangeMS**
  - **Number of tests:** 2 (with 6 total assertions)
  - **Line coverage:** 100% (of a total of 61 raw lines)

Use cases

Use Case 1	Sign up	✓
Use Case 2	Log in of a user	✓
Use Case 3	Report violation	✓
Use Case 4	Filter violations on the map (User)	✓
Use Case 5	Log in for a police officer	✓
Use Case 6	Verify reports	✓
Use Case 7	Receive suggestions	✓
Use Case 8	Filter violations on the map (Municipality)	✓
Use Case 9	Visualize statistics	✗

Live Demo

FIAT
panda
FK 896AE

Thanks for your attention