POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Breaking Red

SOFTWARE ENGINEERING FOR HPC

Author: **Ettore Cirillo, Mattia Gotti, Angelo Notarnicola**

Student ID: 10864944, 10766863, 10830690
Version: 1.0
Release Date: 2025-04-23

Contents

1 | Project Description and Goals

Because of air pollution and greenhouse gas emissions, transportation contributes to worsening those issues. Especially in an urban commuting environment.
To reduce the impact of urban commuting, the following types of action can be implemented.

- **Type 1**: we want to dynamically modify the duration of the traffic lights on the main streets of the city, depending on the directions from where we observe the main traffic movements.
- **Type 2**: we want to analyze daily traffic patterns and identify possible optimization in terms of one-way roads, traffic light configuration, and public transport schedules.

To achieve these goals, we can rely on the following elements:

- A preexisting sensor infrastructure that measures *the number of seconds the car needs to cross each intersection*. This infrastructure is developed according to an event-based style; all sensors periodically publish the data they have acquired on a specific message bus.
- A microservice offering information on public transport schedules. In particular, this microservice is offering the following operations:
  - `getScheduleByStreet`: Given the name of a street, it returns the timetable of all stops present on that street
  - `getScheduleByLine`: Given the number of a specific line, it returns the complete timetable for that line

The project goal is to create a comprehensive software system that integrates the sources of information listed above.
Actions of **Type1** are automatically applied by the software, actions of **Type2** are taken by the urban area managers, who are helped by the software through suggestions.
The software should also log all actions of **Type1** actuated. Moreover, it should publish for all the citizens the following reports:

- Daily reports on the average traffic flow on the main streets and on the **Type1** actions taken.
- Yearly reports on the actions of **Type2** suggested and not accepted by the urban area managers and those suggested and accepted.

2 | Requirement Analysis

2.1. Relevant Human and Non-Human Actors

This section describes the key actors involved in the Breaking Red system, divided into human and non-human entities. Each actor plays a role either by interacting directly with the system or by producing or consuming relevant data. This classification helps to clarify the responsibilities and interactions for the subsequent requirement and design phases.

2.1.1. Human Actors

- **Urban Area Manager**: accesses the system through a secure web portal to review and approve traffic optimization suggestions generated by the Main System.
- **System Administrator**: manages the Local System network through an administration interface. This includes monitoring the health of each node, adding or removing message-bus subscriptions, and updating configurations.
- **Citizen**: accesses the platform to view or download traffic and optimization reports. Although passive, this actor is important for transparency and public awareness.

2.1.2. Non-Human Actors

- **Sensor Infrastructure**: a pre-existing distributed system that publishes traffic measurements on the message bus using an event-driven approach. This was a crucial part to understand, and after discussion with the teacher, we assumed sensors send average crossing times per green light session.
- **Local System**: a system installed at each intersection, responsible for executing traffic light adjustments based on commands received from the Main System. It ensures consistency between the traffic lights and interfaces with the physical infrastructure.
- **Public Transport Service**: an external microservice that provides public transport schedules per line or street. Its data are queried by the Main System to support traffic optimization.
- **Message Bus**: a communication backbone that supports asynchronous, topic-based message exchange between Main and Local Systems.

- **Main System**: the central system responsible for processing sensor data, generating traffic optimization suggestions, logging actions, producing reports, and interacting with authenticated human users. It includes several internal components such as the FlowAnalyzer, TrafficOptimization module, AuthSystem, LogSystem, and ReportSystem, among others.

2.2. Use Cases

2.2.1. UC1 – Real-Time Traffic Light Adjustment

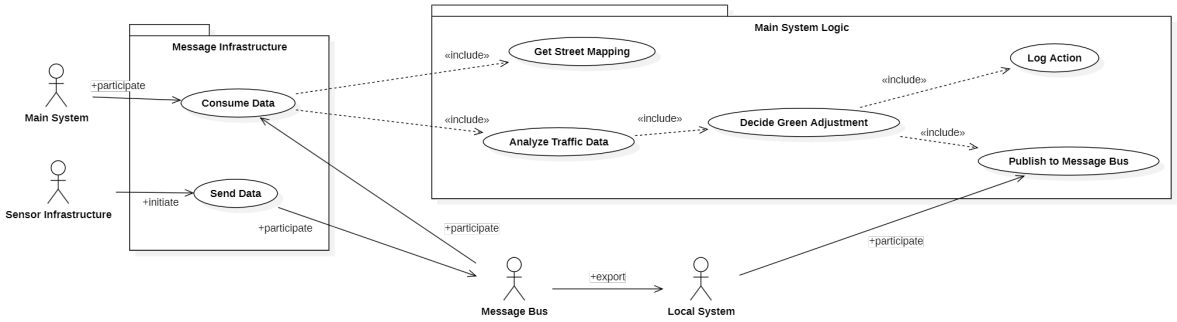

Figure 2.1: Use Case diagram – UC1 Real-Time Traffic Light Adjustment

Use Case ID	UC1
Title	Real-Time Traffic Light Adjustment
Actors	Sensor Infrastructure, Main System, Local System, Message Bus
Preconditions	Sensor is functioning and assigned to a monitored intersection. Message Bus is online. Main System is operational and subscribed to the appropriate topic.

Flow of Events	1. The Sensor Infrastructure detects vehicles and sends the average crossing time to the Message Bus. 2. The Main System receives the message and identifies the associated street and intersection. 3. The system processes the traffic data and compares it with data from intersecting streets. 4. If a timing imbalance is detected, the system determines whether a green light adjustment is required. 5. The Main System publishes a command to the Message Bus to increase the green light duration on the specified street. 6. The Local System receives the command and updates the traffic lights accordingly. 7. The system logs the action for audit and traceability purposes.
Postconditions	Green light duration is adjusted where necessary. Action is recorded in the system log.
Exceptions	• Sensor data not received → the system skips the cycle. • Street identification fails → the system logs an error and performs no adjustment. • Message Bus unavailable → the command is delayed or lost, depending on retry configuration.

2.2.2. UC2 – Generate and Review Optimization Suggestions

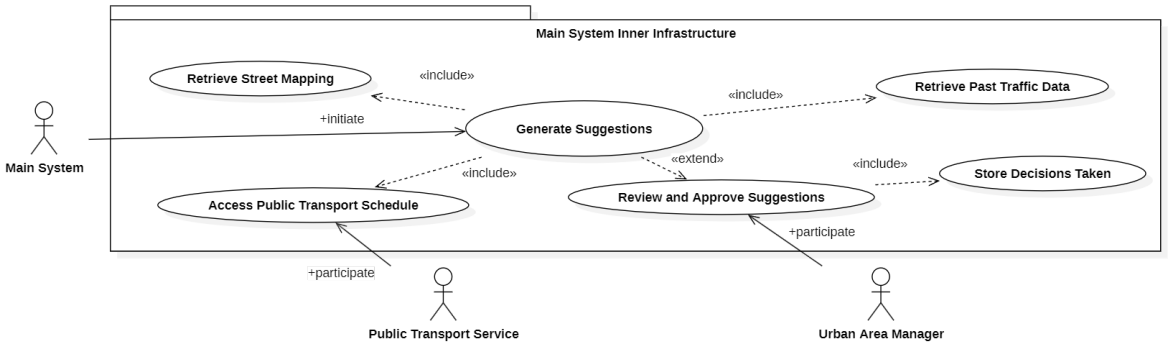

Figure 2.2: Use Case diagram – UC2 Generate and Review Optimization Suggestions

Use Case ID	UC2
Title	Generate and Review Optimization Suggestions
Actors	Main System, Public Transport Service, Urban Area Manager
Preconditions	The system has accumulated sufficient historical traffic data. The Public Transport Service is online and reachable. The Urban Area Manager has valid credentials to access the portal.
Flow of Events	1. At scheduled intervals, the Main System starts a background job to generate traffic optimization suggestions. 2. The system retrieves relevant past traffic data and current public transport schedules. 3. It also accesses mapping information related to streets and intersections. 4. Based on this information, the system computes optimal suggestions to improve traffic flow. 5. Each suggestion is structured and stored with its relevant metadata. 6. The Urban Area Manager logs into the system via a secure interface. 7. The Manager views the list of pending suggestions and reviews their details. 8. The Manager accepts or declines individual suggestions. 9. The system stores the decisions and updates the suggestions accordingly.
Postconditions	Suggestions are marked as approved or rejected. The system reflects the updated status for all reviewed suggestions.
Exceptions	• Failure to retrieve data from the Public Transport Service → partial suggestions only. • Optimization computation fails due to missing or inconsistent data → error logged, suggestions skipped. • Manager session expired → user is prompted to reauthenticate.

2.2.3. UC3 – Generate and Access Reports

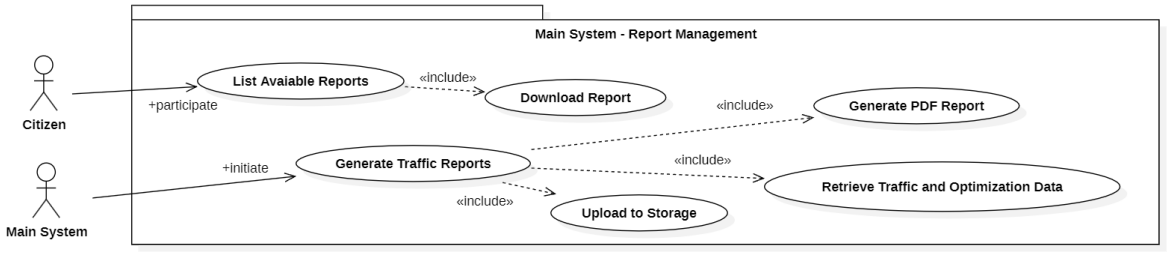

Figure 2.3: Use Case diagram – UC3 Generate and Access Reports

Use Case ID	UC3
Title	Generate and Access Reports
Actors	Main System, Citizen
Preconditions	The system has stored at least one day of traffic and optimization data. Citizen has access to the public report portal.
Flow of Events	1. At scheduled intervals, the Main System initiates the process for generating traffic reports. 2. The system gathers the necessary traffic data and suggestion outcomes. 3. A structured report is generated based on the collected information. 4. The report is published and made available through a persistent public portal. 5. The Citizen accesses the portal and views the list of available reports. 6. The Citizen selects and downloads a desired report.
Postconditions	The report is available for download. The Citizen obtains a local copy of the selected report.
Exceptions	• Required data not available → report skipped, event logged. • Publication failed → report not listed; retry scheduled. • Report unavailable or expired → Citizen receives error message.

2.3. Domain Assumptions

This section presents assumptions about the environment in which the Breaking Red system operates. These assumptions are not enforced by the system itself but are expected to hold true for the system to work reliably. The assumptions are grouped by thematic areas.

Sensor Infrastructure

D1 All relevant intersections are equipped with working sensors capable of detecting and measuring vehicle crossing times.

D2 Sensors compute and transmit the average crossing time at the end of each green-light session, rather than sending data for each individual vehicle. This is an important assumption because it means that this time is *independent* of the destination street of each car. Assume δ_G as the Green light time, then the sensors can perform the average car time by counting the cars entering the intersection. This can be done, for example, by recognizing car plates. Assume N_C the number of cars. Sensors will compute and share on the message bus:

$$ACT = \frac{\delta_G}{N_C} \tag{2.1}$$

where ACT is the Average Car Time for each street at each intersection, where the sensors are placed

Event Delivery and Synchronization

D3 The event-driven infrastructure guarantees reliable message delivery without loss or duplication.

D4 Each event message corresponds precisely to a completed green-light session on a specific street.

D5 All system components (e.g., sensors, Main System, Local Systems) are time-synchronized to ensure correct interpretation of sessions and batch operations.

Traffic Light Execution

D6 Each Local System correctly interprets and applies the timing changes received from the Main System without delay.

D7 Each intersection is managed by exactly one Local System, ensuring consistent and synchronized control of all traffic lights within the same intersection.

External Services and Human Factors

D8 The external Public Transport Service is available and responds within a reasonable delay to requests issued by the system.

D9 Urban Area Managers review and approve optimization suggestions in good faith,

without introducing arbitrary rejections.

D10 No manual interventions override traffic light configurations during active system operations, unless explicitly handled through authorized procedures.

2.4. Requirements

2.4.1. Functional Requirements

This section lists the functional requirements of the Breaking Red system. The requirements are grouped according to the system's core capabilities and are derived from the identified actors and architectural components described previously.

Real-Time Traffic Adjustment (Type 1)

- **FR1** The system shall consume aggregated crossing-time data from the Sensor Infrastructure at the end of each green light session.
- **FR2** The system shall compute and update the average crossing time for each street using the session data received.
- **FR3** The system shall compare the average crossing times among intersecting streets and determine whether an adjustment to green light duration is necessary.
- **FR4** If an adjustment is needed, the system shall publish an event on the corresponding message bus to increase the green light duration of the selected street.
- **FR5** Each executed action of Type 1 (real-time traffic light adjustment) shall be logged persistently.

Optimization Suggestions and Analysis (Type 2)

- **FR6** The system shall persist traffic and public transport data for later analysis by batch processes.
- **FR7** The system shall generate optimization suggestions (Type 2) based on historical traffic data and public transport schedules.
- **FR8** Urban Area Managers shall be able to view and approve or reject traffic optimization suggestions through a secure web interface.

User Access and Reporting

- **FR9** Citizens shall be able to view and download traffic and optimization reports from a public portal.

System Administration

- **FR10** System Administrators shall access a dashboard to monitor Local Systems, check their health status, and manage configuration settings.

2.4.2. Non-Functional Requirements

This section presents the non-functional requirements of the Breaking Red system, structured according to the ISO/IEC 25010:2024 Software Quality Model. Each subsection focuses on a specific quality attribute and includes requirements that ensure system correctness, performance, scalability, reliability, maintainability, portability, availability, and security.

Performance Efficiency

- **NFR1** The system shall evaluate and execute Type 1 green light adjustments within 5 seconds of receiving the corresponding sensor event.

Scalability

- **NFR2** The system shall support at least 100 intersections, each generating up to one event per minute, without degradation in performance.

Availability

- **NFR3** The system shall be available 24 hours a day, 7 days a week.

Reliability

- **NFR4** The system shall tolerate the failure of one or more sensors without halting the entire decision-making pipeline.

Security

- **NFR5** Access to administrative and managerial portals shall require authentication via username and password.

Usability

- **NFR6** The Urban Area Manager and System Admin dashboards shall present information in a clear, accessible, and intuitive format.

Maintainability

- **NFR7** System components shall be modular and independently deployable to support maintenance and upgrades with minimal downtime.

Portability

- **NFR8** The system shall be deployable on any cloud or on-premise infrastructure supporting containerization.

3 | Design

3.1. General Description of the Architecture

The solution consists of two systems: **Main System** and **Local System**.

3.1.1. Main System

Is the system that consumes the message from the street sensors, analyzes them in real-time, and, if necessary, sends the command to increase the green time to a specific street. It also saves them persistently to be analyzed later with a full history of traffic information and public transportation information, given from an external resource. It also allows, with a specific web portal, the `urban area manager` to obtain traffic optimization suggestions, and civilians to view and download reports.
An `system admin` is allowed to log into a dedicated portal where it can manage the Local System network, including health monitoring, the addition of new nodes, and configuration updates.

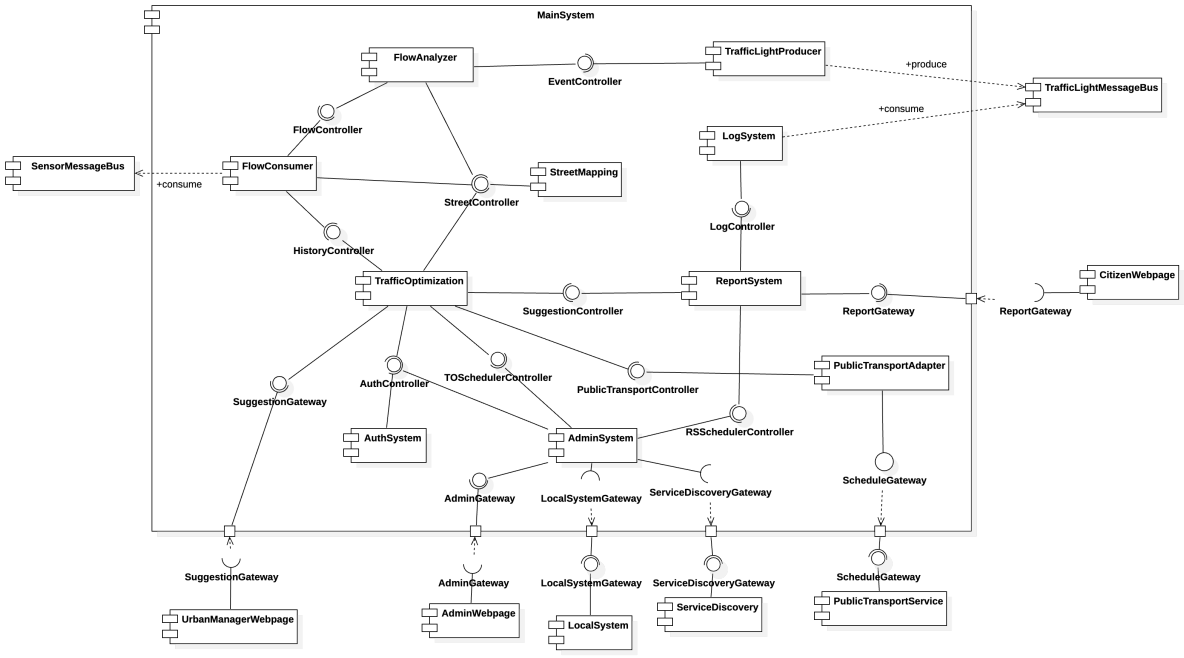

Figure 3.1: Component diagram of Main System

- **FlowConsumer**: It works as an adapter for the preexisting sensor infrastructure. Consumes events from the `SensorMessageBus` and forwards them to `FlowAnalyzer` and `TrafficOptimization`.
- **FlowAnalyzer**: keeps track of the latest data received from sensors and computes an average for the different intersections of the same street. Using the `StreetMapping` component, it compares the timings with other intersecting streets, and if the difference exceeds a programmable threshold, it sends a message to the `TrafficLightProducer` with all the necessary information, such as which street is involved and the amount of time the green light should be increased, which is limited to a specific range.
- **TrafficOptimization**: keeps traffic data persistently to be analyzed through a batch job (daily) that generates suggestions. These suggestions are stored and later reviewed by the Urban Area Manager. This service is also connected to the external `PublicTransportService` through the `PublicTransportAdapter`, which provides information about public transportation.
  The `urban area manager` can access an `authenticated web page` where they can view and approves the suggestions.

The separation between `FlowAnalyzer`, which processes data in real-time, and `TrafficOptimization`, which stores data and executes batch processes, forms what is called the λ `architecture`.

- **PublicTransportAdapter**: adapter layer to connect with the external `PublicTransportService` resource.
- **StreetMapping**: provides information about each managed street, including which other streets it intersects with and which sensors are assigned to monitor each street.
- **AuthSystem**: manage the authorization for the `urban area mangaer` and `system admin` webpages.
- **AdminSystem**: through a service discovery mechanism, it monitors all `Local Systems` connected to the network. It provides a web interface where the system administrator can configure each Local System, e.g. by adding or removing subscribed message buses or checking the status of each node.
- **TrafficLightProducer**: produces events on different message buses, with each managed street having its own dedicated bus. Based on information received from the `FlowAnalyzer`, it sends an event to increase the green light duration on the appropriate bus. For each event sent, it also notifies the `LogSystem`.
- **LogSystem**: persistently saves every executed action of Type 1.
- **ReportSystem**: executes scheduled jobs (daily and yearly). It retrieves data from `TrafficOptimization` and `LogSystem`, generates reports, and uploads them to a file storage system accessible to citizens through a web page.

3.1.2. Local System

Each intersection will have a centralized system that manages all the traffic lights at the crossroad. This is important, as each intersection requires a local system that ensures consistency between the traffic lights within that intersection.
It receives and executes messages sent by the Main System through the message bus.
Every Local System can also communicate with Main System through direct communication to be configured and analyzed.
It interfaces with the existing traffic light electronic system.

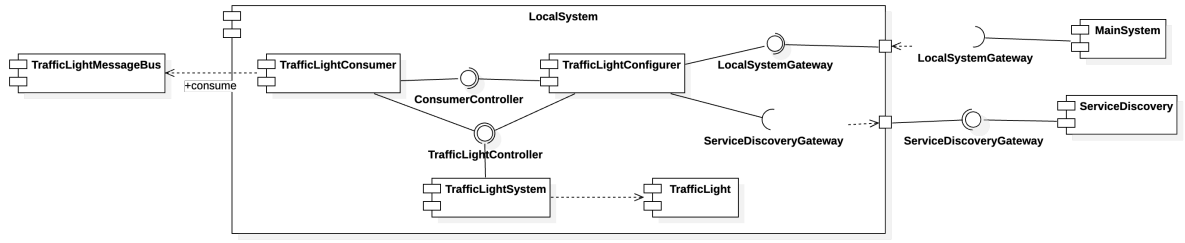

Figure 3.2: Component diagram of Local System

- **TrafficLightConsumer**: consumes events from the message bus related to the street managed by the Local System. It forwards them to the `TrafficLightSystem`.
- **TrafficLightSystem**: it's the system that interfaces with the existing electronic system. Manages all the traffic lights assigned to it. It stores the mapping between street names and their corresponding traffic lights. When it receives a command to increase the green time of a specific street, it consequently increases the red time for all the other streets it manages. It's responsible for the consistency.
- **TrafficLightConfigurer**: is responsible for interfacing with the service discovery mechanism to allow the Main System to identify new Local Systems. It acts as the entry point through which the Main System can configure and retrieve the current state of the Local System.
- **TrafficLight**: the actual traffic light.

3.2. Sequence Diagrams

3.2.1. Sensors publish and traffic lights consumes

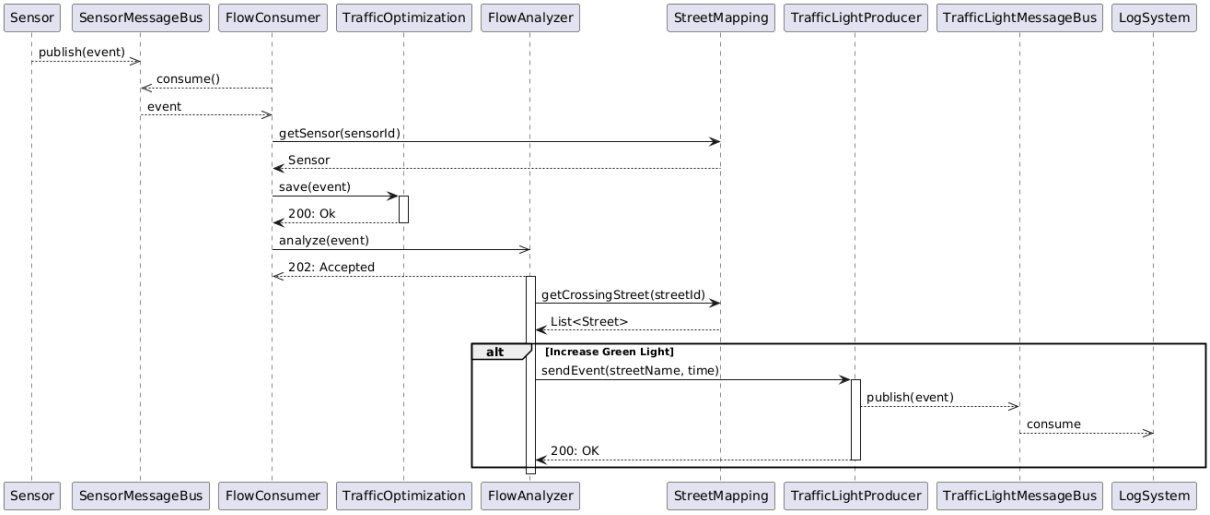

Figure 3.3: Sequence diagram of sensor publish on message bus

When a sensor publishes an event on the message bus, the Main System consumes it through the `FlowConsumer` component. It retrieves the sensor's information from the `StreetMapping` service, including which street the sensor is associated with.

The event is then split and forwarded to two components:

- `TrafficOptimization`, which stores it persistently for future batch analysis.
- `FlowAnalyzer`, which processes the event in real-time, retrieving crossing street data from `StreetMapping`.

If the `FlowAnalyzer` determines that intervention is necessary, it contacts the `TrafficLightProducer`, which publishes the appropriate command to the message bus of the traffic lights.

Every event published is also consumed by the `LogSystem`, which persistently stores each Type 1 action performed.

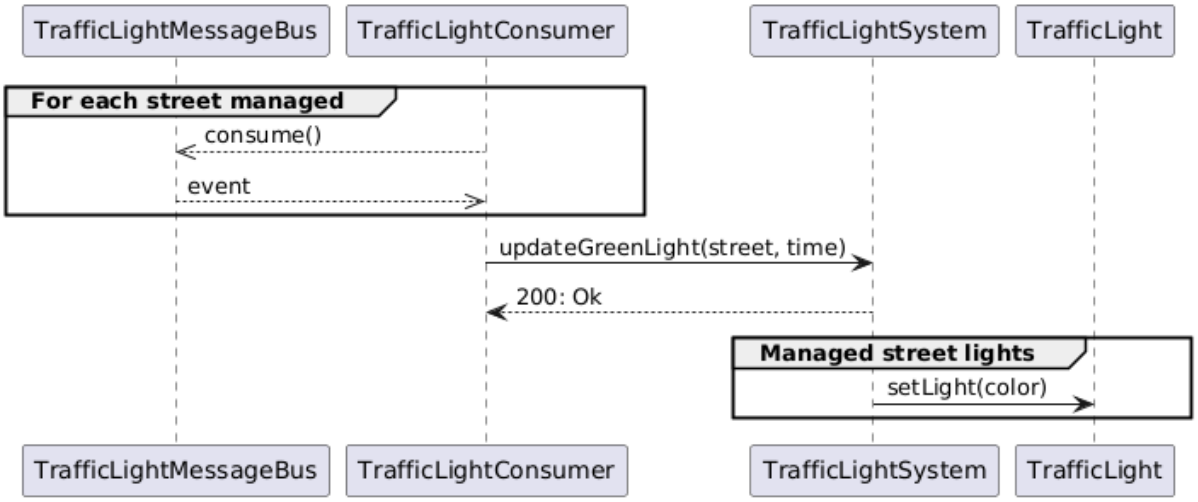

Figure 3.4: Sequence diagram of traffic light consume from message bus

A Local System subscribes to one topic for each street it manages. Upon receiving a command, it forwards it to the `TrafficLightSystem`, the component responsible for coordinating the traffic lights. When an update command is received, the `TrafficLightSystem` adjusts the traffic light schedule according to the new instructions.

3.2.2. Generate suggestion for the Urban Area Manager

Below is the detailed component diagram of the macro-component `TrafficOptimization`.

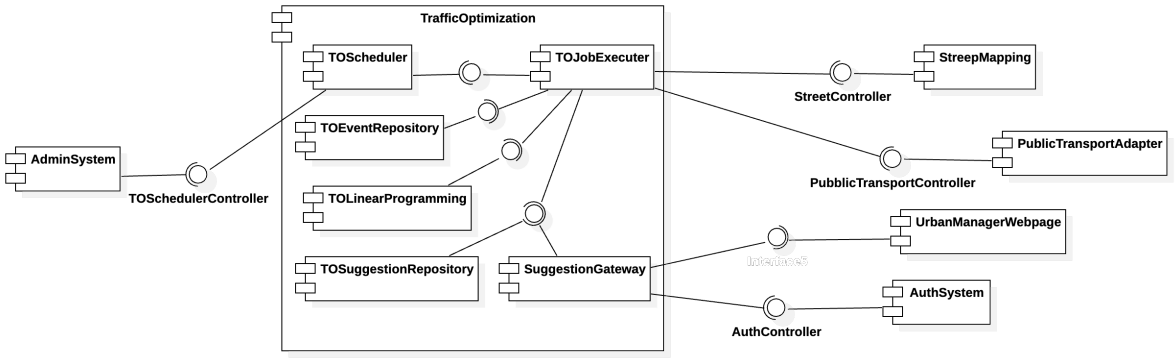

Figure 3.5: Component diagram of TrafficOptimization

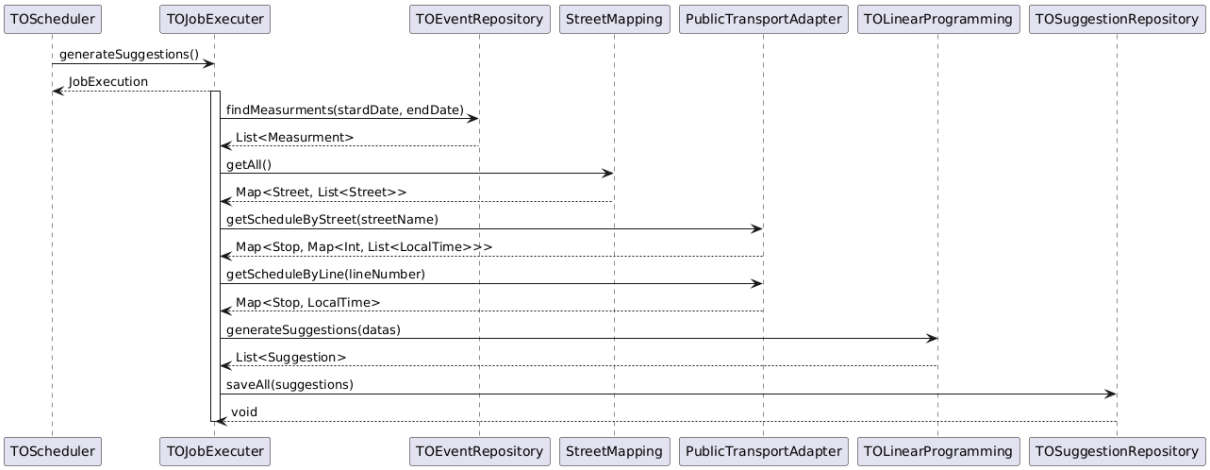

Figure 3.6: Sequence diagram of the generation of suggestion for the Urban Area Manager

The execution of the job is managed by the `TOScheduler`, which triggers the job either at a specific time or after a fixed delay, as configured by the system administrator. The `TOScheduler` sends a command to the `TOJobExecutor`, which gathers all the necessary information, packages it, and forwards it to the `TOLinearProgramming` component. This component is responsible for performing the actual computation and generating the traffic optimization suggestions.

Once generated, each suggestion is saved by the `TOSuggestionRepository`. Suggestions are structured in a way that prevents duplication in the database and avoids potential conflicts between suggestions.

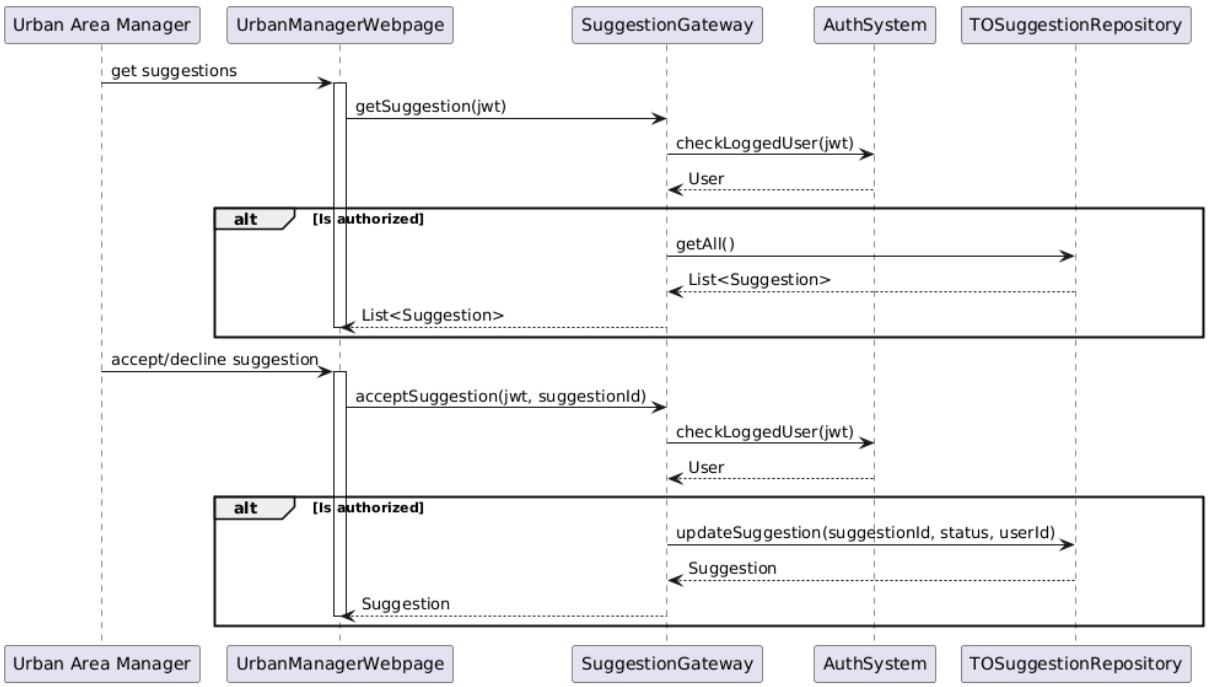

Figure 3.7: Sequence diagram of the interaction of the Urban Area Manager

Every action performed by the Urban Area Manager is protected through authentication. The Urban Area Manager can access all the suggestions stored in the database and has the ability to accept or reject individual suggestions, which are then updated accordingly in the database.

3.2.3. Generate reports for the citizens

Below is the detailed component diagram of the macro-component `ReportSystem`.

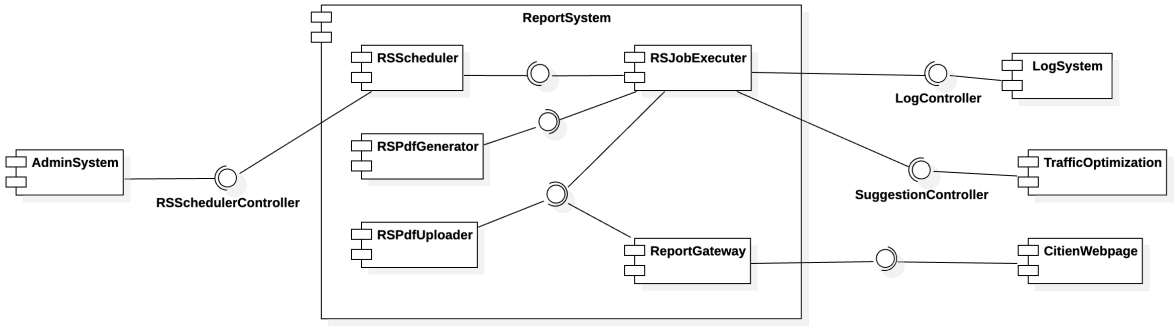

Figure 3.8: Component diagram of ReportSystem

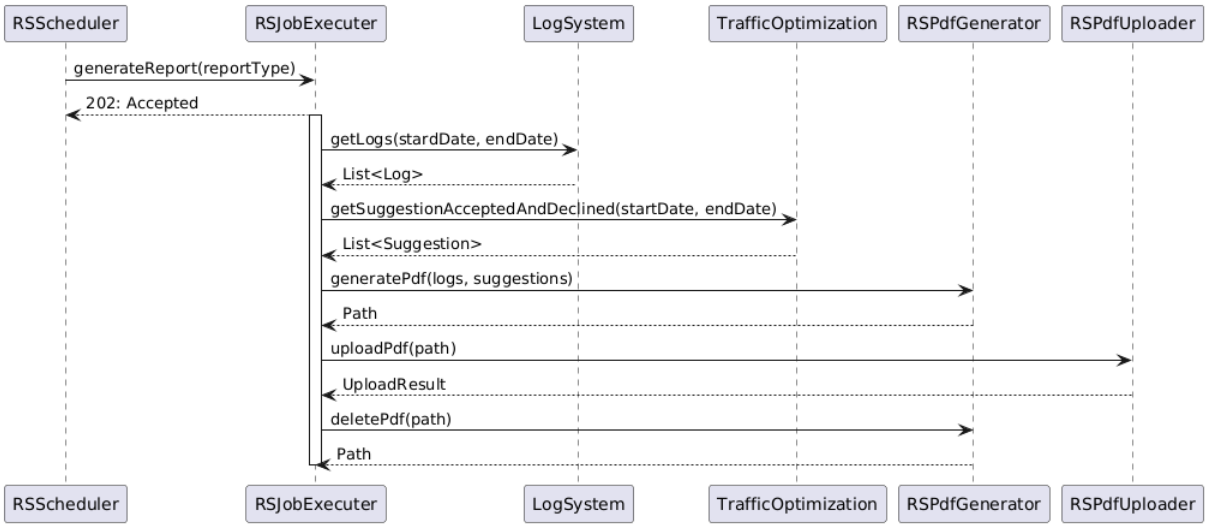

Figure 3.9: Sequence diagram of the generation of report for the citizen

Similar to `TrafficOptimization`, the `ReportSystem` triggers report generation through a scheduler component, the `RSScheduler`, which is configured by the system administrator. The scheduler triggers the `RSJobExecutor`, which, at the start of the job, gathers all the required data — including accepted and rejected suggestions from `TrafficOptimization`, and all Type 1 actions retrieved from the `LogSystem`.

The collected data is then passed to the `RSPdfGenerator`, which uses predefined templates to generate and save a PDF locally. Finally, the `RSPdfUploader` is responsible for

uploading the generated PDF to a cloud storage service. After a successful upload, the local copy of the file can be safely deleted.

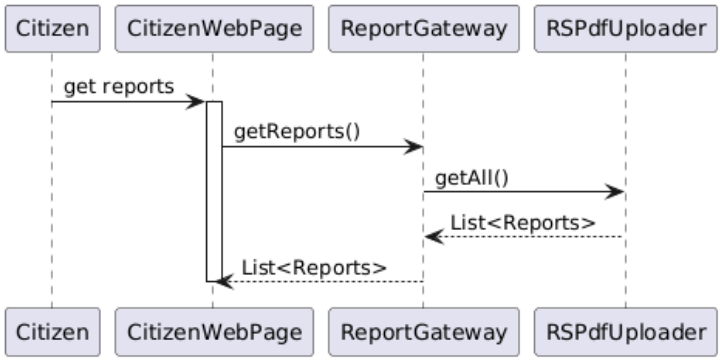

Figure 3.10: Sequence diagram of the interaction of the citizen

Citizens can access all generated reports through the `CitizenWebpage`, which retrieves them from the `RSPdfUploader`. The uploader returns a list of all uploaded reports, each associated with a URL. By clicking on the link, the citizen can directly download the corresponding PDF to their device.

3.3. Critical Points and Design Decisions

3.3.1. Microservice Architecture

The `MainSystem` is implemented with a **Microservice architecture**.
It is an architectural style in which an application is decomposed into a set of different components. These components are independent, each running in its own process and, when needed, communicating through a common mechanism.
This allows:

- **Technology heterogeneity**: Each component to use a different technology stack, depending on which one the team is more comfortable with and which best suits the specific use case.
- **Scaling**: Each component can be scaled and deployed independently (e.g., in case the number of sensors increases).

Each component, if needed, has its own database.

- **TrafficOptimization**: `MySQL`, relational database
- **LogSystem**: `MySQL`
- **AuthSystem**: `MySQL`
- **FlowAnalyzer**: `Redis`, key-value in-memory database
- **StreetMapping**: `Neo4j`, graph database

- **ReportSystem**: `AWS S3`, not a database, but an object storage service

This architecture is implemented with `Docker` and `Kubernetes`.

- **Docker**: allows software to be deployed as a container, which is a fully functional and isolated environment that runs a specific application. Multiple containers on the same machine share the host operating system's kernel.
- **Kubernetes**: is a `container orchestrator` that automates the deployment, management, and scaling of containers across a cluster of machines running containerization software, such as Docker.

3.3.2. Event-drive Architecture

The `LocalSystem` is based on **Event-driven architecture**. It is an architecture in which communication between decoupled services is handled through events sent to an event dispatcher. Each component can publish events to a specific message bus, and all components subscribed to that bus will receive them. This allows for easy addition or removal of components without reconfiguration and enables flexible communication patterns, such as one-to-many or many-to-one.
Every street will have its own dedicated message bus, and all Local Systems that manage that specific street will be subscribed to it.
This is implemented using `Apache Kafka`, which provides a distributed, scalable, fault-tolerant, and secure framework suitable for event-driven architectures.
The service discovery mechanism is provided by `ZooKeeper`, which acts as a coordinator for distributed systems.

3.3.3. Authentication

The `Urban Area Manager` and the `System Administrator` must be authenticated to access their respective web pages. Authentication is handled using the JSON Web Token (JWT) standard. JWT is a proposed internet standard that encodes a JSON payload in a token containing all the necessary information for the user's session, such as user role, expiration time, and access rights.

3.3.4. Network

All systems must be connected to the Kafka and ZooKeeper clusters to enable message exchange. This requires that each `LocalSystem` has network access, which must be defined and managed by the urban infrastructure administrator.

The network should be properly isolated to prevent potential malicious intrusions.
Additionally, it is mandatory to correctly configure Access Control Lists (ACLs) and to enable Kafka authentication mechanisms in order to provide an extra layer of security.