EcoTraffic

Smart urban mobility for a greener future

Lorenzo Esposito 10765981, Alessandro Frisone 10834360,
Francesco Grassi 10841139, Giorgio Venezia 10807335

April 23, 2025

Version 1.1.0

Contents

1 The Project and the project goals

1.1 The problem

Two urgent global concerns are environmental sustainability and climate change; because of air pollution and greenhouse gas emissions, transportation, especially urban commuting, contributes to worsening those issues.

1.2 The goal

We want to dynamically modify the duration of traffic lights on the main roads in the city depending on the directions from where we observe the main traffic movements. For instance, if, at a certain point in time, we observe that the traffic flow on a certain road A is significantly higher than in the crossing roads, then we may decide to extend, for instance, for one hour, the duration of green lights on A (and, consequently, extend the duration of red lights in the crossing roads).
We want to analyze the daily traffic patterns and identify possible optimizations in terms of one-way roads, traffic lights configuration, and public transport schedule.
We want to collect information about the planning of events attracting large crowds (e.g., important sport events, concerts, fairs) and define event-specific configurations for traffic lights, roads and public transport schedules.

1.3 Stakeholders

- Drivers: benefit from reduced waiting times and improved traffic flow.
- Citizens: benefit from lower air pollution and enhanced public transport.
- Urban Area Managers: responsible for optimizing city mobility and approving changes.
- Event Planners: coordinate with the system for efficient management of large events.

2 Requirement Analysis

2.1 Actors

2.1.1 Human Actors

- Drivers: use the traffic light system.
- Citizens: access the EcoTraffic public portal to view traffic reports and use public transport services.
- Urban Area Manager (UAM): responsible for approving or rejecting traffic light configurations and optimizations.

2.1.2 Non-Human Actors

- Traffic Lights Control System: receives traffic light configurations from the EcoTraffic system.
- Sensor Infrastructure: sends sensor information to the EcoTraffic system via data bus.
- Public Transport Microservice: sends public transport schedule to the EcoTraffic system via function calls.
- News Channel: transmits information about city events to the EcoTraffic system.

2.2 Use Cases

2.2.1 Scenarios

Traffic light duration adjustment

1. During peak traffic hours, vehicles take more than necessary to cross a particular intersection coming from a busy road, while the crossing road is less used.
2. The traffic system sensor measures crossing times and publishes them on the message bus.

3. EcoTraffic retrieves and stores the data in a database.
4. EcoTraffic analyzes the crossing times to detect imbalances in traffic flow.
5. If an imbalance is detected, EcoTraffic computes adjusted green light durations for the affected intersections.
6. EcoTraffic sends the updated timings to the traffic light control system.
7. EcoTraffic records the modifications performed in a log file.

Daily Analysis and optimization

1. At 00:10 AM, EcoTraffic retrieve daily data from the database.
2. EcoTraffic analyzes this data to detect daily traffic patterns and to find possible optimization in terms of one-way roads, traffic light configuration, and public transport schedules.
3. EcoTraffic retrieves the current transport schedules via the public transport microservice.
4. Suggested changes are stored in the database for review by the Urban Area Manager (UAM).
5. Once the UAM accesses the system, suggestions are displayed for approval or rejection.
6. EcoTraffic records the Urban Area Manager's decision in a log file for yearly reporting.
7. EcoTraffic sends the accepted suggestions to the traffic light control system.

Traffic Zone Optimization

1. The Urban Area Manager asks to EcoTraffic to verify if a certain zone is optimized in terms of one-way roads, traffic lights configuration, and public transport schedule.
2. EcoTraffic retrieves the traffic patterns of the zone from the database.
3. EcoTraffic analyzes the data suggest improvements to one-way roads and traffic light configuration.
4. EcoTraffic retrieves the current transport schedules via the public transport microservice to get all the bus lines that have a stop into the area to optimize.
5. Using this data, EcoTraffic proposes an optimized configuration. (i.e. if in the zone there's a school, then the system could anticipate the timetable of the stops near that to avoid the students arriving late or could suggest increasing the number of buses in the area).

6. EcoTraffic presents to the UAM the recommendations and waits till the UAM decides to accept or reject the recommendation and stores the decision.
7. EcoTraffic records the Urban Area Manager's decision in a log file for yearly reporting.
8. EcoTraffic sends the accepted suggestions to the traffic light control system.

Event-specific configurations

1. News channel publishes information about upcoming events with the expected attendance.
2. EcoTraffic receives event information from the news channel, categorizes it by expected attendance, location, and scheduled time.
3. EcoTraffic analyzes historical traffic patterns from similar events.
4. EcoTraffic retrieves public transport schedules.
5. EcoTraffic generates event-specific configuration recommendations for traffic lights and roads.
6. The UAM accesses to EcoTraffic and reviews the suggestions proposed and decides to accept or reject them.
7. EcoTraffic records the Urban Area Manager's decision in a log file for yearly reporting.
8. EcoTraffic sends the accepted suggestions to the traffic light control system.

Citizen views public traffic reports

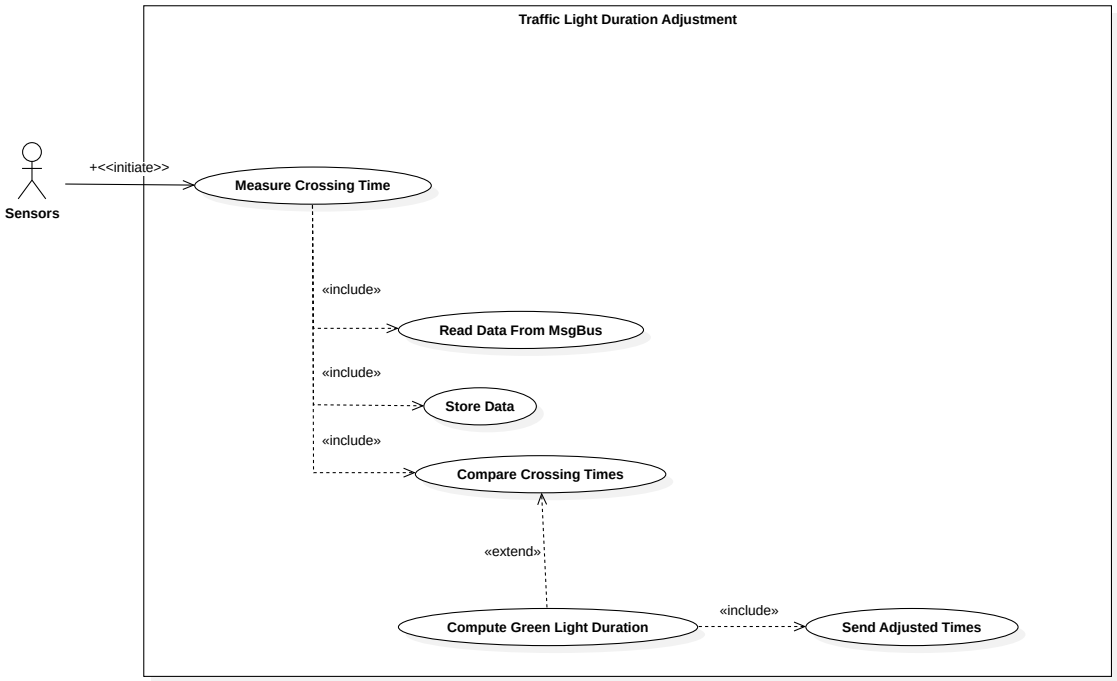
1. A citizen accesses the EcoTraffic public portal and selects either daily or yearly traffic reports.
   - 1.1. The citizen selects "Daily Traffic Reports" and chooses the date and time.
     - 1.1.1. EcoTraffic retrieves daily report data from the database service.
     - 1.1.2. EcoTraffic displays report showing:
       a. Average traffic flow on main roads.
       b. Visualization of peak congestion periods.
       c. List of actions taken automatically.
   - 1.2. The citizen selects the "Yearly Reports" option.
     - 1.2.1. EcoTraffic displays yearly report options.
     - 1.2.2. EcoTraffic retrieves yearly report data from the database service.
     - 1.2.3. EcoTraffic displays a comprehensive report showing:
       a. Suggested actions that were accepted.
       b. Suggested actions that were rejected.

UAM views optimization to approve or reject

1. UAM access to EcoTraffic public portal.
2. EcoTraffic presents the list of pending optimization suggestions awaiting UAM approval or rejection.
3. The UAM analyzes each suggestion and decide to approve or reject choosing an option displayed.
4. EcoTraffic records the UAM's decision in a log file for yearly reporting.

2.2.2 Use case diagrams

Traffic light duration adjustment

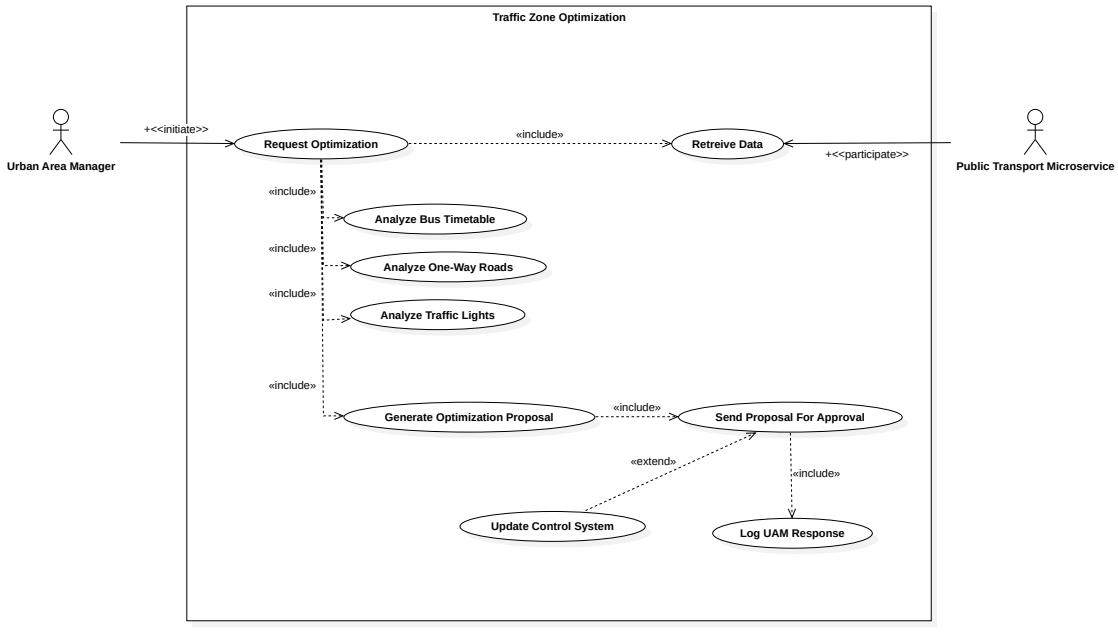


Aspect	Description
Actors	Sensors, Control System
Entry condition	After sensors measure new crossing times, new data arrives on the bus.

Continues on next page

Aspect	Description
Flow of events	1. EcoTraffic reads data from the message bus. 2. EcoTraffic stores the received data. 3. EcoTraffic compares the crossing times of the roads in the crossings. 4. If EcoTraffic detects traffic load imbalance, it computes the green light duration to optimize vehicle flow in the more congested direction.
Exit condition	The system sends the adjusted times to the traffic lights control system.

Traffic zone optimization

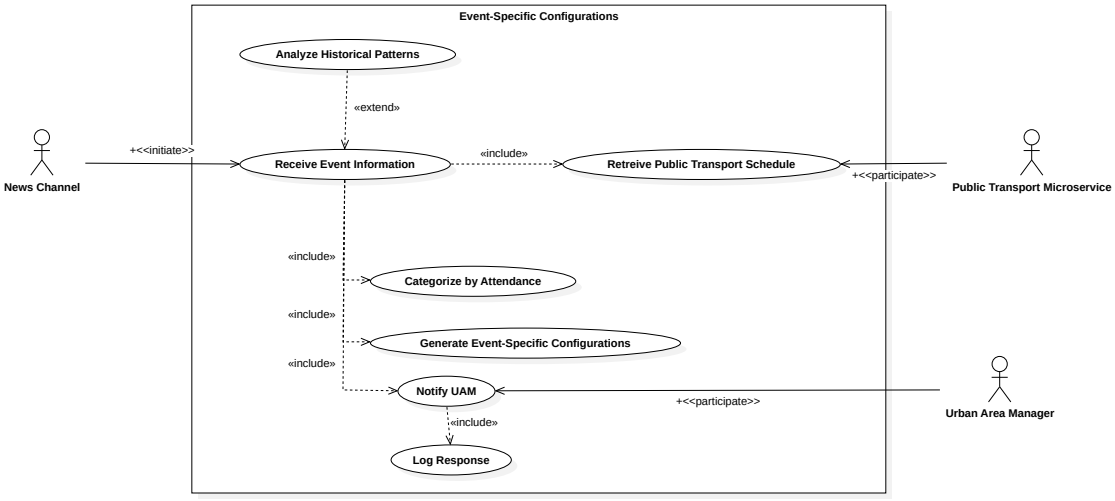


Aspect	Description
Actors	Urban Area Manager, Public Transport Microservice, Control System

Continues on next page

Aspect	**Description**
Entry condition	UAM accesses the EcoTraffic public portal and decides to request optimization of a certain area.
Flow of events	1. EcoTraffic analyzes the crossing times of the zone indicated by the UAM to identify potential optimizations regarding one-way roads and traffic light configurations. 2. EcoTraffic retrieves public transport schedules via microservice using getScheduleByStreet and getScheduleByLine operations. 3. EcoTraffic tries to find better schedules for the public transport line. 4. The system sends the new scheduling and configuration proposal to the UAM for approval. 5. When the decision is taken, the answer is written into a log file. 6. If the decision is to accept the proposal, EcoTraffic sends the new configuration to the traffic light control system.
Exit condition	Successful write of the log and successful communication with the traffic light control system.

Event-specific configurations



Aspect	Description
Actors	News channel, UAM, Public Transport Microservice
Entry condition	News channel publishes information about upcoming events.
Flow of events	1. EcoTraffic categorizes the scale of the event by attendance. 2. EcoTraffic analyzes historical traffic patterns and reviews past decisions for similar events. 3. EcoTraffic retrieves public transport schedules via microservice using getScheduleByStreet and getScheduleByLine operations. 4. EcoTraffic generates event-specific configurations that are optimized in terms of one-way roads, traffic light configurations, public transport schedules.
Exit condition	The system writes into the database the possible optimization found.

Daily Analysis and Optimization

Aspect	Description
Actors	Webservice, Database
Entry condition	It's ten past midnight.
Flow of events	1. EcoTraffic analyzes the crossing times measured the day before retrieved to obtain traffic patterns. 2. EcoTraffic analyzes the traffic patterns to identify potential optimizations regarding one-way roads and traffic light configurations. 3. EcoTraffic retrieves public transport schedules via microservice using getScheduleByStreet and getScheduleByLine operations. 4. EcoTraffic tries to find better schedules for the public transport line.
Exit condition	The logs the possible optimization found.

Citizen views public traffic reports

Aspect	Description
Actors	Citizen
Entry condition	Citizen accesses EcoTraffic public portal.

Continues on next page

Aspect	Description
Flow of events	1. EcoTraffic presents options for viewing reports. 2. The citizen selects the type of information that they want to be displayed. 3. EcoTraffic retrieves daily or yearly data from the database service. 4. EcoTraffic elaborates the data to facilitate interpretation. 5. EcoTraffic displays a report with the elaborated data. 6. The citizen chooses between seeing more information or exiting the portal.
Exit condition	The citizen exits the portal.

UAM views optimization to approve or reject

Aspect	Description
Actors	UAM
Entry condition	UAM accesses the EcoTraffic public portal and decides to view the optimization waiting for approval.

Continues on next page

Aspect	Description
Flow of events	1. EcoTraffic finds the optimization proposals waiting for approval. 2. EcoTraffic displays all the decisions to be taken. 3. EcoTraffic waits for the UAM to answer to each request. 4. Every time a decision is taken EcoTraffic stores it into a log file. 5. If the decision is to accept the proposal, EcoTraffic applies the modifications.
Exit condition	All the decisions have been taken.

2.3 Domain assumption

DA1. The sensor infrastructure works correctly and at low latency 24/7.

DA2. The traffic lights are not faulty and set their state correctly in time from the EcoTraffic system.

DA3. Drivers behave according to the traffic light state.

DA4. No car can obstruct the passage in the crossing no matter the reason.

DA5. Events planners always report to the news channel up-to-date events in the city.

DA6. The public transport microservice always returns the right timetable given a line or the name of a street.

DA7. The public transport microservice, the sensor infrastructure, the database and the traffic light control system are always available and responds in a timely manner.

DA8. The traffic light control system is able to schedule future adjustments in the traffic light state.

2.4 Requirements

2.4.1 Functional Requirements

FR1. In any circumstance, the system must not allow two orthogonal traffic lights to be green at the same time.

FR2. The system must modify the duration of the green lights to reduce traffic.

FR3. The system shall process and aggregate traffic data to identify traffic flow patterns.

FR4. The system should send adjustment commands to the traffic light control system.

FR5. The system should be able to write to a log file what is needed.

FR6. The system must generate optimization suggestions for one-way road and traffic light configurations.

FR7. The system must generate optimization suggestions for public transport schedules.

FR8. The system shall continuously receive data from both the message bus and the news channel.

FR9. The system must gather data from the microservice.

FR10. The system must assess the potential traffic impact of planned events.

FR11. The system must generate event-specific suggestions for public transport adjustments.

FR12. The system shall present suggestions to urban area managers for review.

FR13. The system shall record and apply the acceptance or rejection of the proposal by the urban area managers.

FR14. The system must generate daily reports on average traffic flow.

FR15. The system must generate yearly reports on suggestions proposed and their outcome.

FR16. The system shall publish reports for public access.

2.4.2 Non-Functional Requirements

NFR1. The system shall process sensor data in real-time.

NFR2. The system should be available 24/7.

NFR3. The system shall implement traffic light adjustments in 15 seconds.

NFR4. The system shall generate reports within 1 hour after midnight.

NFR5. The system should maintain data consistency during communication with external systems.

NFR6. The system should be scalable to support the addition of new sensors and bus lines without requiring significant changes to the architecture.

NFR7. The system dashboard should be web-based and accessible from any device.

2.4.3 Constraints

C1. The setLight function must be atomical.

C2. The call to the microservice must be done using an API REST.

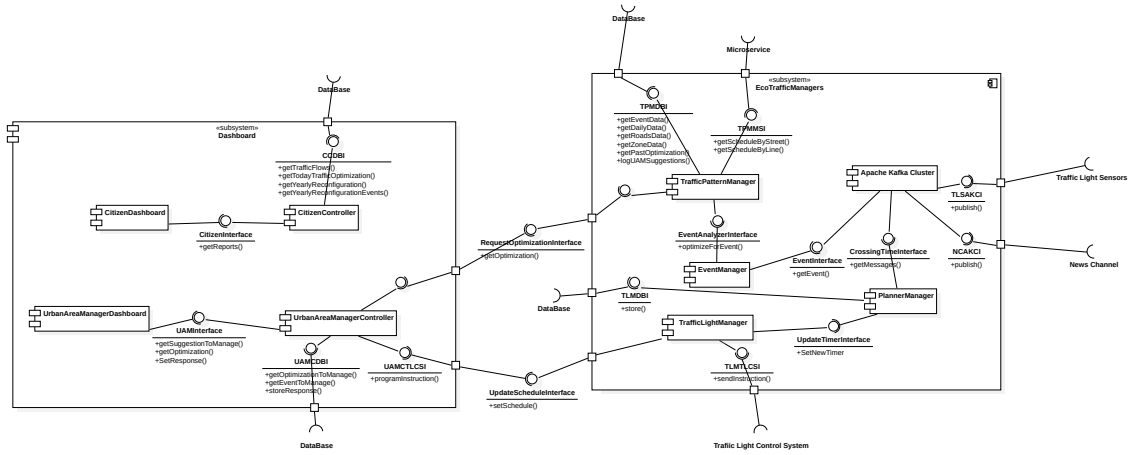
C3. All data must be stored in a PostgreSQL database.

C4. The frontend must use only open-source libraries.

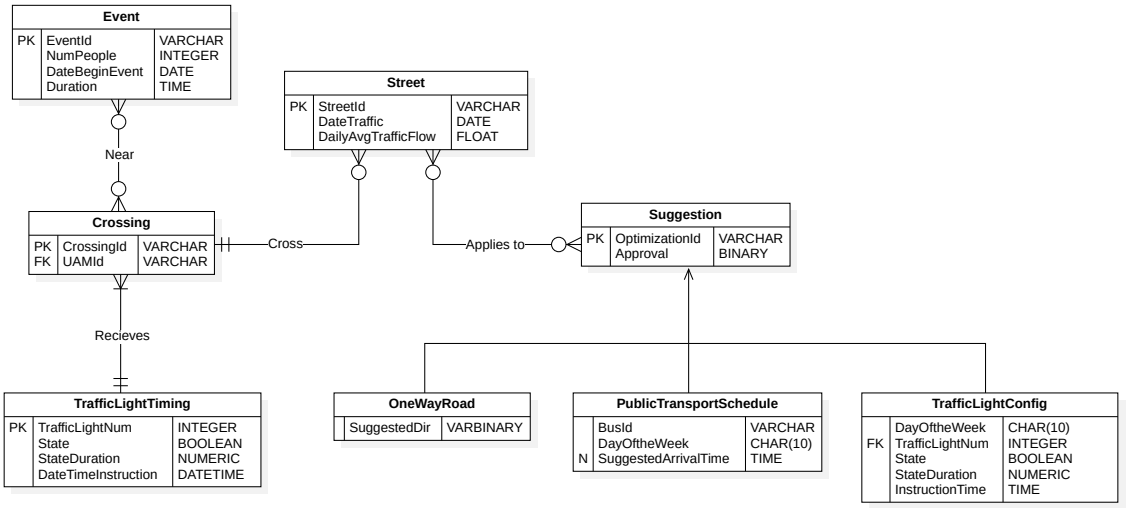
3 Design

3.1 Architecture

3.1.1 Components diagram



There have been inserted multiple interfaces named "Database" for readability. Actually, there is only one database that is used by all the components. Its structure is shown in the next diagram:



3.1.2 Subsystems and components

The EcoTrafficManagers subsystem includes the following components:

- **ApacheKafkaCluster:** the component is based on a framework for the event-driven paradigm and it includes primitives to create event producers and consumers and a runtime infrastructure to handle event transfer. This component it's the one responsible for receiving the data from the sensors system and from the news channel and to distribute them to the right components, which will then process and analyze them. More information can be found at `https://kafka.apache.org/`. In this case, the synchronization it's of the "at least once" type, so the component is able to resend the messages in case of failure.

- **PlannerManager:** this component receives the data measured by the sensors system from the ApacheKafkaCluster and provides a method to analyze this data to find unbalanced traffic loads into a crossing, to compute the new time in which the green light is switched on to reduce traffic congestion, to store into the database of the system the data obtained from the ApacheKafkaCluster and to connect to the TrafficLightManager to apply the right modifications. Due to the big amount of data that the component has to process, it is designed to work in a multi-threaded way, so it can process multiple messages at the same time.

- **EventManager:** this component receives the event specification from the ApacheKafkaCluster after the news channel has published them. This component provides methods to retrieve information (such as the event type, the date, the place and the expected attendance) and a method to call the TrafficPatternManager component to optimize the event configuration to reduce traffic load.

- **TrafficPatternManager:** this component could be called by three events. The first is when the EventManager calls it to perform event-based optimization, in this case, the component queries the database to find similar events and the daily traffic patterns in the zone where the event takes place, then generate possible optimizations from this data and from the timetables obtained after the getScheduleByStreet() and the getScheduleByLine() calls to the microservice. The second is when the UrbanAreaManagerController asks for an optimization around a specific zone, in this case, the component queries the daily traffic patterns in the zone and the timetables and tries to optimize the traffic loads. The third takes place automatically ten minutes after midnight, in this case, the component queries the daily crossing time in all the crossings of the area and finds traffic patterns, after that, it tries to optimize in terms of one-way roads, traffic lights configuration, and public transport schedule.

- **TrafficLightManager:** this component is responsible for managing the traffic lights. It receives immediate adjustments from the PlannerManager and new schedules from the UrbanAreaManagerController.

The Dashboard subsystem includes the following components:

- **CitizenDashboard:** this component shows to the citizen an interface from where it can be decided to see the daily reports or the yearly ones. After

choosing, the component calls the CitizenController to get the reports. When the response arrives, the component shows to the user the reports.

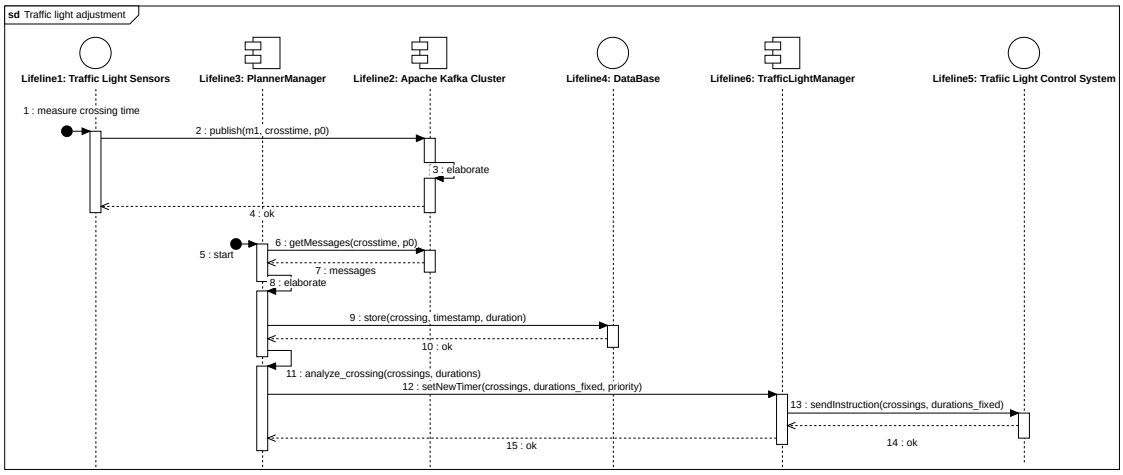
- **CitizenController:** this component receives the request from the citizen dashboard and then queries the database to satisfy the request. After, the data are elaborated and sent to the dashboard.

- **UrbanAreaManagerDashboard:** this component shows the UAM an interface from where it can decided if to see the pending suggestions or to ask the system to optimize all the configurations in a specific zone. If the first option is chosen, it calls the UrbanAreaManagerController to get the pending suggestions. When the response arrives, the component shows to the user the reports and waits till all the decisions are taken. When each decision is taken, the component sends the answer to the controller. If the second option is taken, the dashboard presents the zones in the area and waits for the UAM's decision, after it arrives the component sends to the controller the requests and waits for the answer, after this arrives it displays to the UAM the suggestions, when the decision is taken, the component sends the answer to the controller.

- **UrbanAreaManagerController:** this component is responsible for managing the requests arriving from the dashboard and redirecting them either to the database if the UAM wants to see the pending suggestions or to the TrafficPatternManager if the UAM wants an optimization for a specific zone or to the TrafficLightManager if the UAM wants to set a new schedule directly to the traffic lights. After the responses arrive, the component sends them to the dashboard. If the answer is an optimization the component waits for the approval or the rejection and stores the decision.

3.2 Sequence Diagrams

Traffic Lights Adjustment



This sequence diagram shows the communication and data flow involved in adjusting traffic light timings based on crossing data.

1. The traffic light sensors system measures a crossing time.

2. The traffic light sensors system publishes the crossing time on the message bus and it's received from the Apache Kafa Cluster.

3. The Apache Kafka Cluster performs internal operation, in particular, the message is added to a follower broker by the leader broker.

4. The Apache Kafka Cluster responds to the traffic light sensor system.

5. PlannerManager continuously operates in the background. Note that this diagram has to entry point for clarity, but it's actually the same PlannerManager component that polls for messages repeatedly throughout the system operation.

6. PlannerManager tries to get a message.

7. The message, if present, is sent to PlannerManager.

8. PlannerManager elaborates the message received extracting the important information.

9. PlannerManager sends a message to the database to make it store the data received.

10. Successful store.

11. PlannerManager analyzes the data received to find eventually unbalanced traffic loads. If it finds them, then it computes also the new green light time for a particular traffic light.

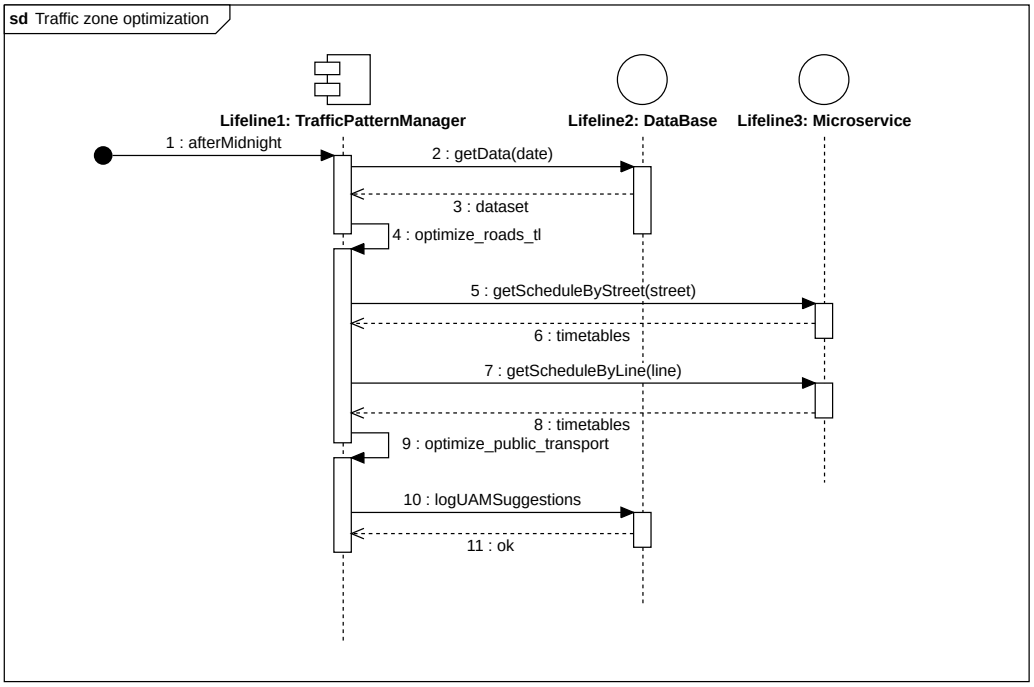
12. PlannerManager sends a message to the TrafficLightManager with the modifications to perform.

13. TrafficLightManager sends a the instructions to the traffic light control system.

14. TrafficLightManager receives the ACK message from the traffic light control system.

15. TrafficLightManager sends a message to the PlannerManager to confirm that the modifications have been performed.

Traffic Zone Optimization

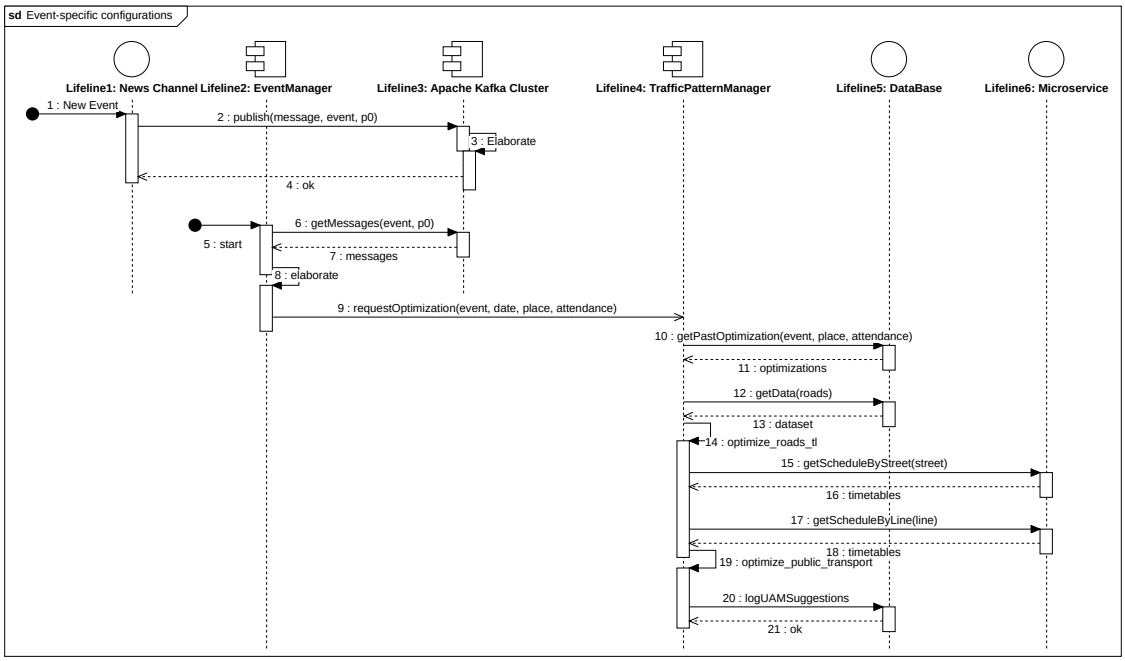


This sequence diagram illustrates the data flow between TrafficPatternManager, database, and Microservice components for optimizing traffic zones, including both road traffic and public transportation optimization.

1. The process starts when the TrafficPatternManager receives an "afterMidnight" trigger.
2. TrafficPatternManager requests data from the database collected on the date passed as an argument.
3. The database responds by returning the data to the TrafficPatternManager.
4. TrafficPatternManager analyzes the data to identify traffic patterns.
5. TrafficPatternManager tries to optimize the road traffic lights to reduce the traffic loads.
6. TrafficPatternManager asks for the timetables of the streets to the microsystem provided by the state.
7. The microsystem returns the street timetables.

8. TrafficPatternManager asks for the timetables of the lines to the microsystem.
9. The microsystem returns the line timetables.
10. TrafficPatternManager tries to optimize public transportation schedules.
11. TrafficPatternManager logs the suggestions and stores them in the database.
12. The Microservice responds with an "ok" confirmation message, completing the sequence.

Event-specific Configuration



This sequence diagram illustrates the data flow between the News Channel, EventManager, Apache Kafka Cluster, TrafficPatternManager, Database, and Microservice components for event-specific traffic optimization.

1. The process starts when the News Channel triggers a "New Event".
2. The News Channel publishes the event message to Apache Kafka with relevant parameters. In particular, the message is added to a follower broker by the leader broker.
3. Apache Kafka acknowledges and elaborates the message.
4. News Channel confirms receipt with an "ok".
5. EventManager it's working.
6. EventManager queries Apache Kafka to retrieve messages related to the event.
7. Apache Kafka returns the corresponding messages.

8. EventManager elaborates on the event data.

9. It sends a request to the TrafficPatternManager to optimize traffic based on event details (event, date, place, and expected attendance).

10. The TrafficPatternManager fetches from the database any previous optimizations for similar conditions.

11. The database returns past optimization results.

12. TrafficPatternManager requests road data from the Database.

13. The Database returns the road dataset.

14. TrafficPatternManager attempts to optimize road traffic light configurations and one-way road configurations.

15. TrafficPatternManager performs the getScheduleByStreet operation from the Microservice.

16. The Microservice provides street timetables.

17. TrafficPatternManager performs the getScheduleByLine operation from the Microservice.

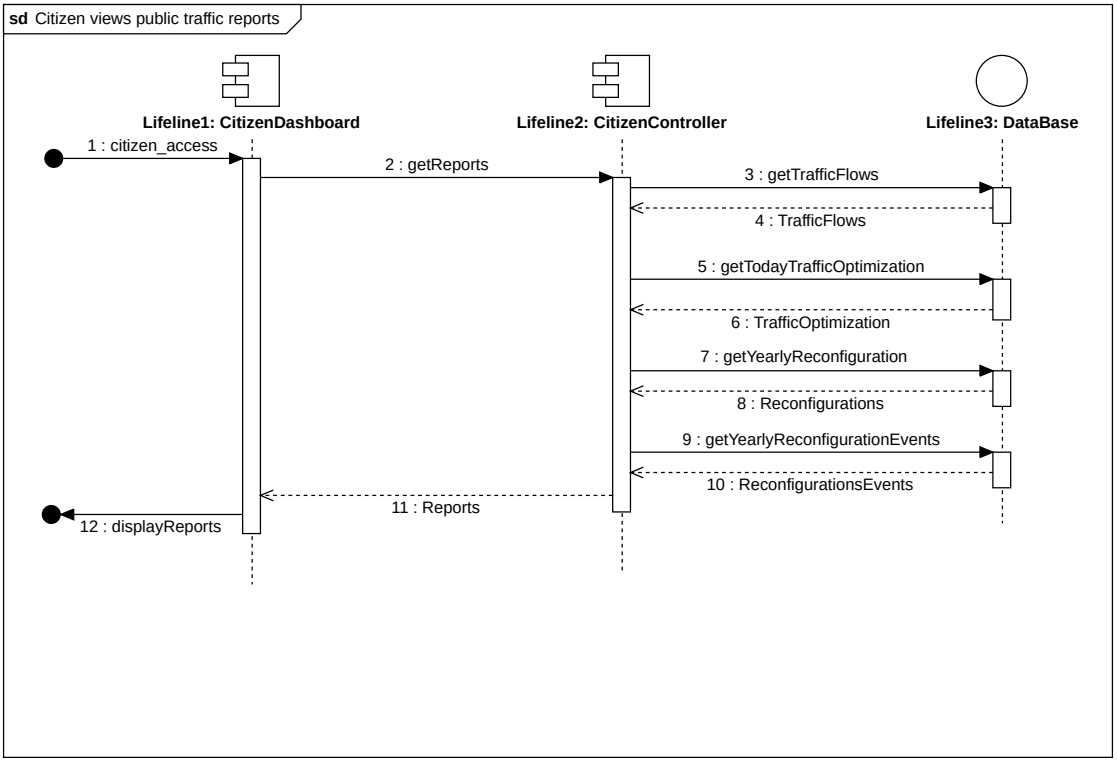
18. The Microservice responds with line timetables.

19. TrafficPatternManager optimizes public transportation schedules based on the retrieved data.

20. It logs the Urban Area Manager suggestions into the Database.

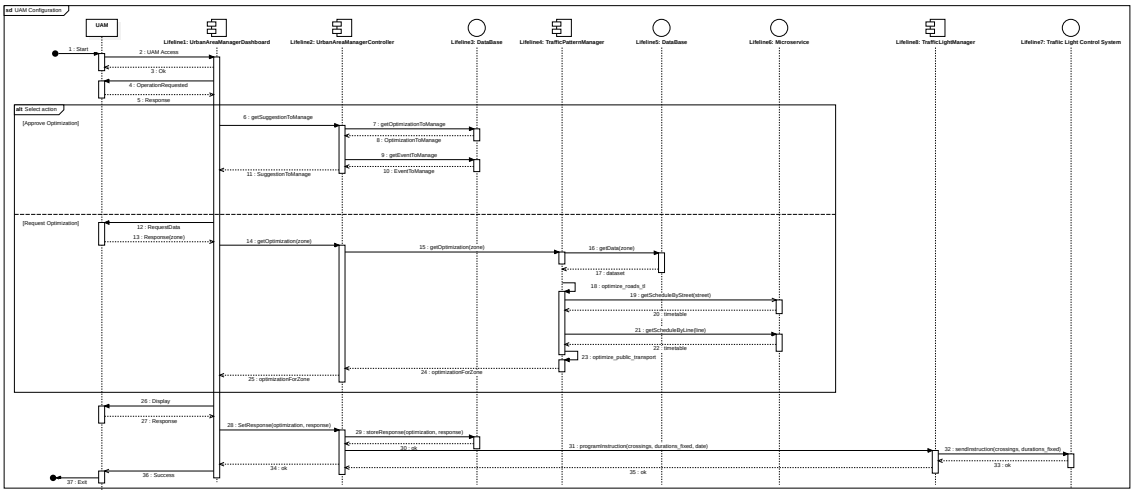
21. The Microservice responds with an "ok", completing the sequence.

Citizen views public reports



This sequence diagram shows the communication and data flow involved in citizens viewing public traffic reports.

UAM configuration



This sequence diagram shows the communication and data flow involved in UAM configuration and how the UAM can approve or reject the suggestions proposed by the system.

1. The sequence starts.

2. The UAM's access the system's dashboard.

3. It confirms with an ACK message.
4. The UrbanAreaManagerController shows to the UAM all possible action that can be taken
5. The UAM choose what to do.
6. If the UAM chooses to see the pending suggestions, the dashboard asks to the controller to retrieve them.
7. The controller queries the system for pending optimizations to manage.
8. The database returns the pending decision.
9. The controller then requests related event data to manage.
10. Event data is provided in response.
11. Suggestions to manage are returned to the controller.
12. If The UAM chooses to optimize a zone, the UrbanAreaManagerDashboard asks the UAM to indicate the zone to optimize.
13. The UAM indicates the zone to optimize.
14. The UrbanAreaManagerDashboard sends the request to the UrbanAreaManagerController.
15. The UrbanAreaManagerController asks to the TrafficPatternManager to optimize the zone.
16. The TrafficPatternManager queries the database for the zone data.
17. The database returns the zone data.
18. The TrafficPatternManager tries to optimize the zone for traffic light and one-way road configurations.
19. TrafficPatternManager performs the getScheduleByStreet operation from the Microservice.
20. The Microservice provides street timetables.
21. TrafficPatternManager performs the getScheduleByLine operation from the Microservice.
22. The Microservice responds with line timetables.
23. TrafficPatternManager optimizes public transportation schedules based on the retrieved data.
24. TrafficPatternManager sends optimization for the zone required to the UrbanAreaManagerController.

25. The UrbanAreaManagerController sends to the UrbanAreaManagerDashboard the optimization to display them.
26. The UrbanAreaManagerDashboard displays the proposed optimization.
27. The UAM provides a response to the optimization.
28. The response is sent to the UrbanAreaManagerController.
29. The response is stored in the database.
30. ACK is returned.
31. TrafficLightManager receive the adjustment to perform from the UrbanAreaManagerController.
32. Instructions are sent to the Traffic Light Control System.
33. The control system acknowledges with an "ok".
34. TrafficLightManager confirms success.
35. UrbanAreaManagerController confirms success.
36. UrbanAreaManagerDashboard confirms success.
37. The sequence ends.

3.3 Critical points and design decisions

Critical Point	**Design Decision**
Analyzing traffic patterns requires complex processing of historical and real-time data.	Our architecture is designed to separate the concerns of data collection, processing, and presentation. The `TrafficPatternManager` is specialized in creating algorithms to identify common traffic patterns and propose optimizations, while the `TrafficLightManager` handles the actual traffic light adjustments.
A scalable and efficient communication system is required to handle high-frequency data from sensors and from the news channel.	We decided to use the `ApacheKafkaCluster` to handle event-driven communication, allowing real-time and asynchronous data processing of a large volume requests.

Continues on next page

Critical Point	**Design Decision**
The system must be able to provide adaptive responses to city events with a minimum latency.	The `EventManager` component processes the event data and classifies the event based on the expected attendance. It then triggers the `TrafficPatternManager`, which will provide an appropriate optimization routine.
The system must be able to provide a user-friendly interface for both citizens and urban area managers.	The `CitizenDashboard` and `UrbanAreaManagerDashboard` components are designed to provide intuitive interfaces for users to access reports and make decisions. The dashboards are separated from the core logic of the system to ensure modularity and maintainability.
Human stakeholders must be involved in taking decisions for critical changes.	On the `UrbanAreaManagerDashboard` pop up suggestions for optimizations retrieved directly from the unified database by the `UrbanAreaManagerController`. The human manager then decides whether to accept or reject the suggestions. When a response is received, the system logs the decisions in the database for future reference.
Supporting future scalability and integration of additional sensors or data sources.	We designed our architecture to be modular, allowing an extension of the sensor net or for new data sources. The `ApacheKafkaCluster` can easily integrate new actors, ensuring that the system can adapt to future needs without significant modifications.
The system must be able to handle data consistency and coherence during communication with external systems.	We implemented a robust error handling and data validation mechanism in the `PlannerManager` and `TrafficPatternManager` components to ensure that only valid data is processed and stored. This helps maintain data integrity and coherence across the system.

Continues on next page

Critical Point	**Design Decision**
The system must be able to report real-time updates to the citizens.	The `CitizenDashboard` is designed to provide real-time updates on traffic optimizations and possible changes on viability. The system uses a push mechanism to notify citizens of significant changes, ensuring they are always informed.
Two differents users can acces the system: the UAM and the citizen. They can performs different operations	The system has two different types of distribution: one for the citizen and one for the UAM. The first presents the `CitizenDashboard`, while the latter the `UrbanAreaManagerDashboard`. The first one is designed for citizens to view reports, while the second one is tailored for urban area managers to manage traffic optimizations and configurations.