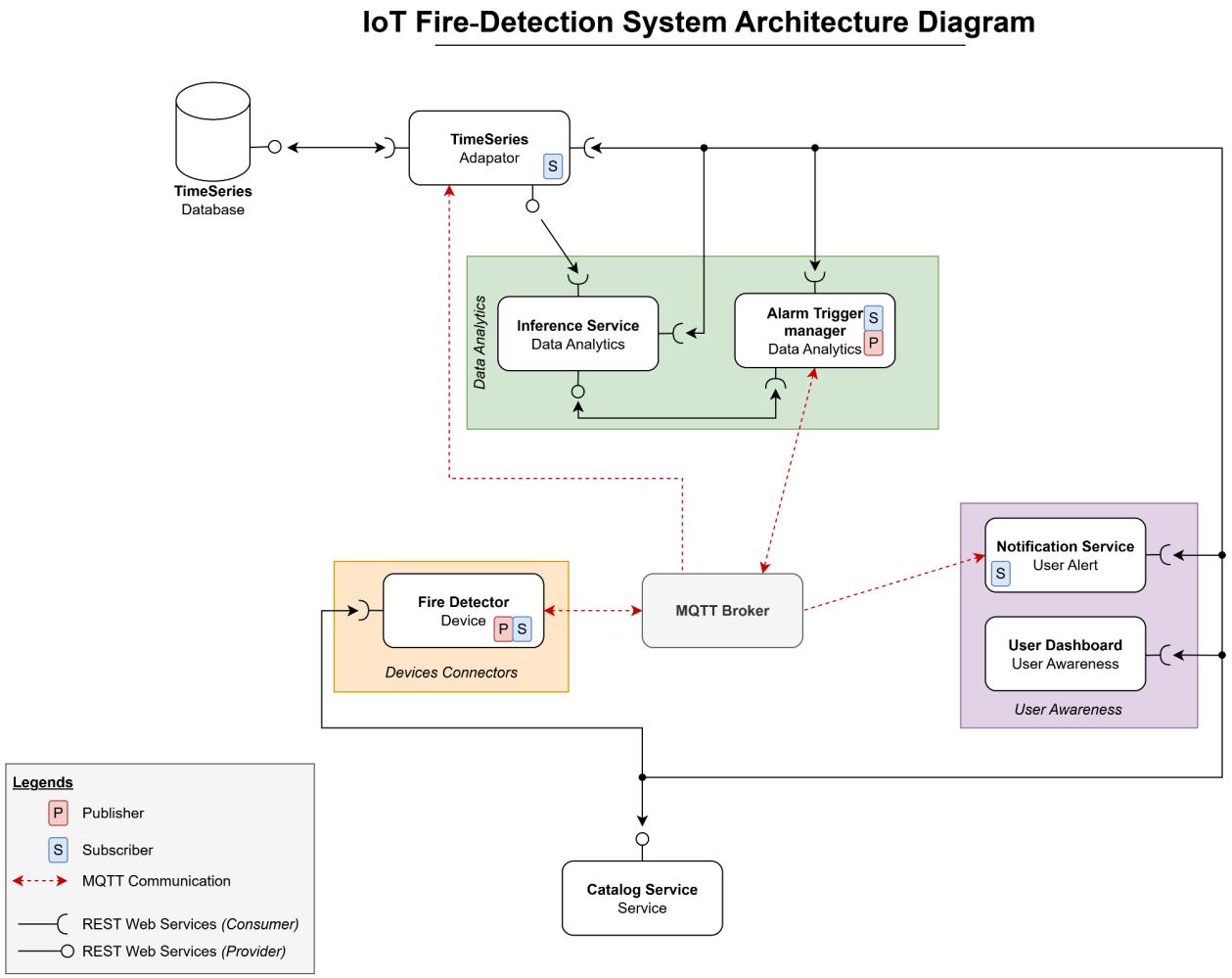
1 Name of Use Case

Name of the Use Case	IoT system for Fire detection city-scaled
Version No.	v0.2
Submission Date	08-12-2025
Team Members (with student ids)	Andrea Marietti (s355074) - Gabriele Efim (s361144) - Federico Galfione (s360864) - Sébastien Rivière (s350430)

2 Scope and Objectives of Function

Scope and Objectives of Use Case	
Scope	The proposed IoT platform aims at providing services for smart fire detection management for firefighters.
Objective(s)	The project aims to create a reliable, smart city-scale service that simplifies fire detection and intervention for firefighters, thanks to the precise location of the sensors.
Domain(s)	Smart building - Smart City
Stakeholder(s)	Enterprise companies - Firefighters Department
Short description	The proposed IoT platform aims to create a fire detection system on an urban scale that can make the intervention of firefighters more efficient and precise.The overall platform is thought to unify and standardize the management of fire accidents in a city context. It uses smoke, temperature and air quality sensors connected to a controller in order to establish whether there is a fire or not, and call tempestively, if necessary, the firefighters. Summarizing, the main features it offers are: - Fire detection (precise localization) - Efficient notification system through Telegram Bot - Management of devices - Calibration of the data retrieved from the dedicated database - Test function (manually launching the alert system) - Broadcast alarm system

3 Diagram of Use Case



4 Complete description of the system

The proposed IoT platform for Smart Cities follows the microservices design pattern. It exploits two communication paradigms:

- PUBLISH/SUBSCRIBE with MQTT protocol
- REQUEST/RESPONSE with REST web services

In this context, several actors have been identified:

- The **MQTT Broker** provides an asynchronous communication based on the publish/subscribe approach.

- All the data regarding the system is collected through a **Catalog Service**. All the data is provided to the other components of the system through REST web services. We define 4 different categories:
  - The **Resource Catalog** stores information regarding the hardware devices registered in the system and how they communicate. The main data entries could be: ID, status, date_of_installation, data_frequency, location_id…

    - The **Service Catalog** contains all the information regarding the available services, including how to reach them: IP addresses,
    - ports, protocols, endpoints…
    - The **User Catalog** registers all the users of the system: user_id, role…
    - The **Localization Catalog** maps the position of each device. Since the aim of the project is to precisely detect the position of a fire accident, it is crucial to own all the information regarding: building name, floor, room, GPS coordinates…

- The **Fire Detector** is the core of the system: it is composed of a Raspberry Pi connected to temperature, smoke and air quality sensors through MQTT topics and it uses REST web services to retrieve data from the Catalog. It acts both as:
    - *Publisher* : continuously sending the data retrieved from the sensors to the MQTT broker.
    - *Subscriber* : it listens to possible alarm signals sent by the Alarm Trigger manager; if the message is received it automatically triggers the acoustic alarm.

  In case of temporary disconnection from the cloud, it can operate autonomously using local thresholds for emergency detection.

- The **Alarm Trigger Manager** processes real-time sensor data retrieved through MQTT by forwarding these values to the Inference Services (with REST) for obtaining a fire-event probability: in the case that it exceeds a threshold, the Alarm Trigger Manager will collect the information of the concerned device and users from the Catalog (RESTful connection) for raising the alarm. It will also update the status value regarding the sensors in the Catalog.

- The **Inference Service** is needed for the computation of mean and standard deviation of the historical data collected from the Time Series Adaptor through a REST API. It then uses these parameters to compute normalization, perform inference to predict the presence of a fire (for example a Machine Learning/Neural Network implementation) and finally it returns to the Alarm Trigger Manager the probability of a real fire-event (always through a REST API).

- The **Time Series Adaptor** acts as an intermediary providing the REST API for the system to retrieve the information from the Time Series Database. The service retrieves the information from the Catalog and the Time Series Database through REST API. The adaptor ensures that the sensor data, received through the MQTT Broker, is correctly formatted and stored, allowing to make the statistics available for calibrations, normalizations and further analysis.

- The **Time Series Database** stores all historical sensor measurements, provided by the Time Series Adaptor, in an ordered manner that is suitable for time-series analysis. It provides all the data needed for the inference process.

- The **Notification Service** handles warning signals, contacting firefighters through a Telegram Bot, which receives fire notifications from the Alarm Trigger Manager through the MQTT broker.

- The **User DashBoard** is implemented with Node-RED for example. There is a direct connection to the Catalog done by a REST API. It allows managing both devices and users registered in the system by adding, modifying, deleting them. It also retrieves the current state of all alarms and detectors deployed in a city.

5 Desired Hardware components (only among those we can provide)

Device Name	Quantity	Needed for…