UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Master in Informatics Engineering

Consuming data sources to generate actionable items

Master Thesis Report

Author:
Marc Vila Gómez

Advisor:
José Gorchs Morillas
(Worldsensing)

Tutor:
Maria Ribera Sancho Samso
(ESSI - UPC)

January 31th, 2019

Agradecimientos

Me gustaría agradecer a *Worldsensing* y al *inLab FIB* la oportunidad que me ofrecieron para la realización de este proyecto.

Agradecer a todo *Worldsensing*, en especial al equipo de *Innovación.* Y más concretamente al director del proyecto de fin de máster, *José Gorchs*, por su apoyo y supervisión durante este proyecto. Y, al compañero de equipo, *Toni Martinez* por su ayuda en la supervisión de esta memoria y días de trabajo para llevar hacia delante el proyecto.

También agradecer a la rama de *software* en Ingenieria y en concreto a *Sebastian Rajo*, por el apoyo recibido mientras trataba de entender el funcionamiento de su sistema visualizador de datos.

A *Maria Ribera Sancho*, profesora ponente del proyecto, agradecer que junto a su apoyo y su conocimiento, he obtenido unos sabios consejos para la realización de esta memoria.

Y por supuesto, quiero agradecer a *Ester Lorente* por su apoyo durante la realización del proyecto y en la redacción de esta memoria.

Me gustaría dar las gracias por el soporte recibido de mi familia, haciéndome muy sencilla la tarea de no estar en casa durante este último mes, para poder centrarme en este proyecto final de máster.

No quería despedirme sin dedicar este proyecto a mi abuela *Juana* y a mi abuelo *José*, que nos dejó mientras realizaba este máster.

Sin vosotros, esto no habría sido posible.

GRACIAS.

Abstract

Today it is known that the world is full of digital sensors. We currently live in a time when many types of sensors are becoming *Internet Of Things* sensors (*IoT*, sensors with Internet connection). Even information that is shared over the internet, for example on social networks, can be used as an *IoT* sensor.

It is also well-known that there are not many state-of-the-art alert systems. The alert systems that currently exist are based on traditional technologies and a few of them implement any solution that involves the Internet in between.

It was decided to create, in *Worldsensing*, a platform that consumes these IoT sensors to generate response actions related to alert systems. In addition, in order to demonstrate the possible use cases, functions required by European Projects will be incorporated into the project. Starting from the integration of two commercial sensors of *Worldsensing* and the implementation in the platform of generic standards, such as *Common Alerting Protocol (CAP)* for the management of alerts and *OGC SensorThings API* to be able to consume data generated from third parties in a standard way.

This platform will have two possible starting points. The first one will be the ingestion of raw data from IoT sensors, to be analyzed later, and in case of having anomalous values, to initiate the necessary way to launch an alert generated by that sensor, and, if required, to generate an actionable as a response plan. And the second will be the ingestion of data in the form of possible alerts from third party systems, and with them, from a filter, initiate the management of the response plans associated with that alert.

Abstract (Spanish)

Hoy en día se sabe que el mundo está lleno de sensores digitales, actualmente vivimos en un momento en el que muchos tipos de sensores están convirtiéndose en sensores *IoT* (*Internet Of Things*, sensores con conexión a internet). Incluso la información que se comparte por internet, por ejemplo en las redes sociales, puede ser utilizada como un sensor de *IoT*.

Además se observa que no hay muchos sistemas de alertas de última tecnología. Los sistemas de alertas que existen actualmente se basan en tecnologías tradicionales y pocos de ellos implementan alguna solución que implique internet de por medio.

Se decide crear, en *Worldsensing*, una plataforma que consuma estos sensores IoT para generar acciones de respuesta relacionadas con los sistemas de alerta. Y además para demostrar los casos de uso posibles, se incorporarán al proyecto funciones requeridas por Proyectos Europeos. Partiendo de la integración de dos sensores comerciales de *Worldsensing* y de la implementación en la plataforma de estándares genéricos, como por ejemplo el *Common Alerting Protocol (CAP)* para la gestión de alertas y de *OGC SensorThings API* para poder consumir datos generados de terceros de forma estándar.

Esta plataforma, tendrá dos posibles puntos de inicio. El primer punto será la ingestión de datos crudos de sensores *IoT*, para posteriormente ser analizados, y en caso de tener valores anomalos, iniciar el camino necesario para lanzar una alerta generada por ese sensor, y, si se requiere, generar un accionable a modo de plan de respuesta. Y el segundo punto será la ingestión de datos en forma de posibles alertas de sistemas terceros, y con ellos, a partir de un filtrado, iniciar la gestión de los planes de respuesta asociados a esa alerta.

Abstract (Catalan)

Avui dia es sap que el món està ple de sensors digitals, actualment vivim en un moment en el qual molts tipus de sensors estan convertint-se en sensors *IoT* (*Internet Of Things*, sensors amb connexió a internet). Fins i tot la informació que es comparteix per internet, per exemple en les xarxes socials, pot ser utilitzada com un sensor de *IoT*.

A més s'observa que no hi ha molts sistemes d'alertes d'última tecnologia. Els sistemes d'alertes que existeixen actualment es basen en tecnologies tradicionals i són pocs els que implementen alguna solució que impliqui internet pel mig.

Es decideix crear, a *Worldsensing*, una plataforma que consumeixi aquests sensors IoT per a generar accions de resposta relacionades amb els sistemes d'alerta. I a més per a demostrar els casos d'ús possibles, s'incorporaran al projecte funcions requerides per Projectes Europeus. Partint de la integració de dos sensors comercials de *Worldsensing* i de la implementació en la plataforma d'estàndards genèrics, com per exemple el *Common Alerting Protocol (CAP)* per a la gestió d'alertes i de *OGC SensorThings API* per a poder consumir dades generades de tercers de forma estàndard.

Aquesta plataforma, tindrà dos possibles punts d'inici. El primer punt serà la ingestió de dades crues de sensors *IoT*, per a posteriorment ser analitzats, i en cas de tenir valors anòmals, iniciar el camí necessari per a llançar una alerta generada per aquest sensor, i, si es requereix, generar un accionable com a pla de resposta. I el segon punt serà la ingestió de dades en forma de possibles alertes de sistemes tercers, i amb ells, a partir d'un filtrat, iniciar la gestió dels plans de resposta associats a aquesta alerta.

Glossary

Backend:

Data access layer in a software.

Business rule:

Defines or constrains some aspect of business and always resolves to either true or false.

Continous Delivery:

Is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, when releasing the software, doing so manually.

Deploy:

In software, software deployment is all of the activities that make a software system available for use.

Document Object Model:

Document Object Model (DOM), is a cross-platform and language-independent application programming interface that treats an HTML, XHTML, or XML document as a tree structure wherein each node is an object representing a part of the document.

Frontend:

Presentation layer in a software.

Gateway:

Node in a computer network that passes traffic from a local network to other networks or the Internet.

HTTP Requests:

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

- GET: Used to request data from a specified resource.
- POST: Used to send data to a server to create a resource.
- PUT: Used to send data to a server to overwrite a resource.
- PATCH: Used to send data to a server to update a resource.
- DELETE: Used to delete a resource.

IoT device:

Computing devices that connect wirelessly to a network and have the ability to transmit data.

JSON:

JavaScript Object Notation (JSON), is a lightweight data-interchange format.

KPI:

Key Performance Indicator (KPI). A KPI is a key performance indicator that allow us to measure quantitively the results.

LoRa:

Digital wireless data communication technology. LoRa uses license-free sub-gigahertz radio frequency bands like 169 MHz, 433 MHz, 868 MHz and 915 MHz. LoRa enables very-long-range transmissions with low power consumption.

Microservice:

Module or software development that involves de-composing application functionality into individual components that can be deployed separately from each other, and typically communicate via application programming interfaces or APIs.

OpenID:

Open standard and decentralized authentication protocol. Allows users to be authenticated by using a third-party service, eliminating the need for webmasters to provide their own ad hoc login systems, and allowing users to log into multiple unrelated websites without having to have a separate identity and password for each.

Time series database:

Is a database optimized for time-stamped data. Time series data are simply measurements or events that are tracked, monitored, downsampled and aggregated over time.

Contents

List of Figures

List of Tables

1 Introduction

This project is done as a Final Master Thesis (TFM) of the Master in Informatics Engineering at the *Barcelona School of Informatics*, from the *Universitat Politècnica de Catalunya - Barcelona Tech*. The project is being carried out at *Worldsensing*, inside the *Innovation* department of the company.

1.1 The company

Worldsensing is a global IoT pioneer. Founded in 2008, the Barcelona-based company delivers Operational Intelligence to traditional industries and cities. With over 90 employees and offices in Barcelona, London and Los Angeles, *Worldsensing* is globally active and has already conducted projects in +50 countries across 6 continents.

With the mission to transform operations, the company connects difficult to reach and distributed infrastructures such as on-street parking spaces, bridges or mines to the Industrial IoT. While *Worldsensing* is among the leading providers in the world at digitizing assets with wireless sensing technologies, the company also knows how to extract intelligence from collected data. By connecting sensors-based data, people and systems to its operational intelligence solutions, *Worldsensing* generates instant, geo-located insights enabling clients to transform their decision-making and long-term planning processes based on real-time intelligence.

As a leading IoT company which is constantly focused on creating disruptive technology, *Worldsensing* has been recognized as an *Innovative SME* (Small and Medium-sized Enterprises) by the Spanish Ministry of Economy and Competitiveness.

1.2 Contextualisation

In order to have a better understanding of the motivation of this project in *Worldsensing*, in the following sections, it will be introduced the company verticals that will leverage on this project. It will be also introduced the *European Commission-funded* projects in which it participates, and having with it the result of a system that will allow different inputs, even from outside the company, to generate actionable items. Thus, to build and develop a data-input agnostic platform.

1.2.1 Verticals and solutions in Worldsensing

Worldsensing has two main verticals: *Industrial IoT* and *Smartcities*, the next sections will explain how this project will impact the company products allocated for those markets.

1.2.1.1 City Mobility Management

Solutions for city mobility management enable cities, parking and transport operators to have greater control over the operation of traffic flow, parking, and city main KPIs. Smart solutions allow them to gain visibility over all operations in real time so they can easily identify the KPIs that, by means of Big Data technologies, will enable the transformation of insights into actionable items.

Worldsensing's product line includes a parking management and a traffic flow management system as well as a monitoring solution for critical infrastructures and construction sites (which is part of the *Industrial IoT* vertical), that is generically labelled as the *Mobility* or *OneMind* product within the company. [1]

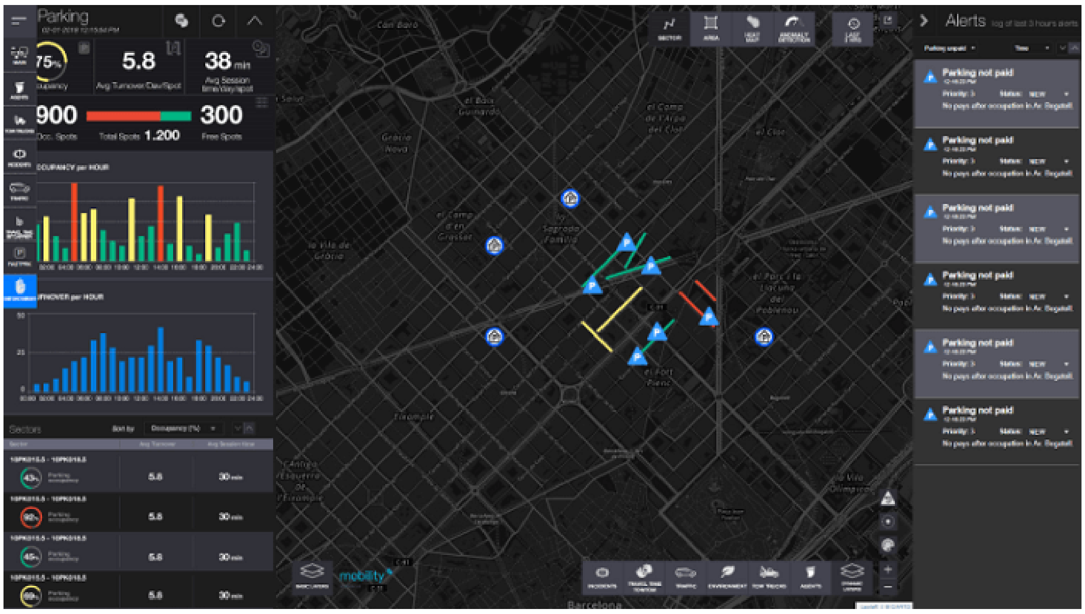


Figure 1: Mobility Platform by Worldsensing

1.2.1.1.1 Smart Parking

Smart parking concepts are constantly evolving to maximize the usage of available parking slots for cities, citizens, and operators. With smart parking solutions, cities, operators and private companies can monitor and manage parking spaces to enhance usage and traffic flow and consequently generate incremental revenues.

Wireless solutions gather information from sensors that provide data about an area, sector or specific spots, date range and time range, enabling providers within the public and private sectors to remotely generate occupation patterns reports and develop pricing strategies in line with real-world usage patterns, that can maximise the ROI (Return On Investment) and at the same time improve the drivers experience when looking for available parking spaces.

[1] City Mobility Management:
https://www.worldsensing.com/application-areas/city-mobility-management/

The Worldsensing parking management system is ideal for urban environments; parking sensors can be deployed across the streets, to enable operators to capture and analyze occupancy data in real-time. [2]

Fastpark

Fastpark is an intelligent Parking Management System that allows cities and operators to manage parking resources more efficiently and parking operators to generate additional revenue. The wireless system uses smart sensors installed in parking spaces and guides drivers to areas with vacancies via electronic panels and mobile apps.

Fastpark is the only system in the market featuring an integrated monitoring tool for sensors which provides real-time sensor updates on battery status, temperature, last message transmitted, among others. [3]

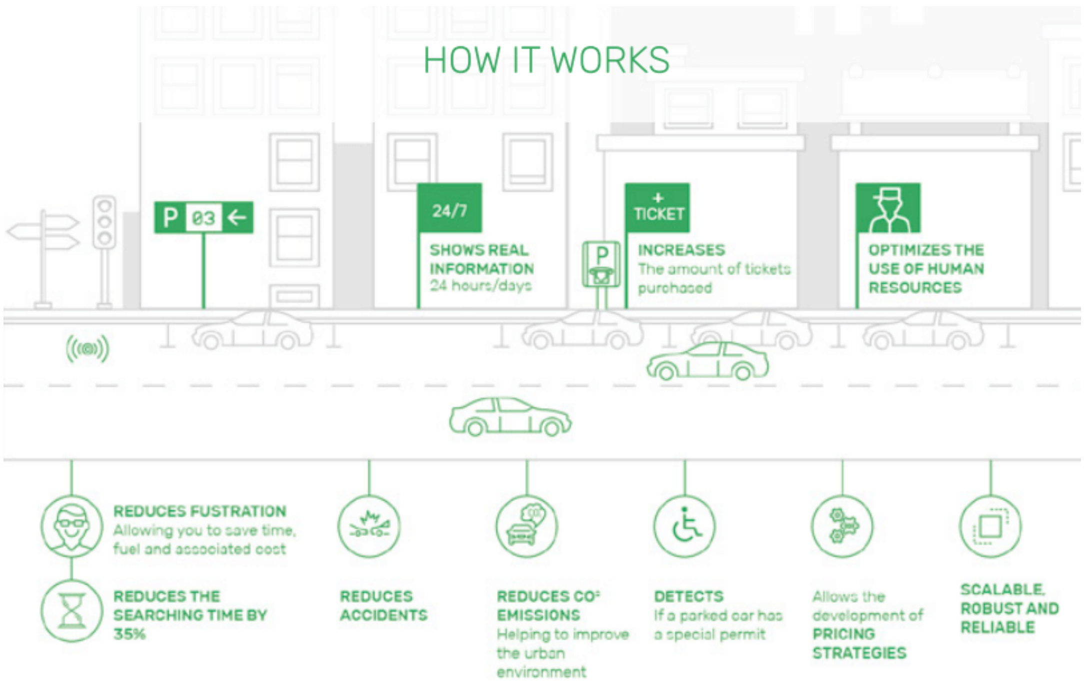

HOW IT WORKS
P 03
24/7
SHOWS REAL INFORMATION 24 hours/days
TICKET
INCREASES The amount of tickets purchased
OPTIMIZES THE USE OF HUMAN RESOURCES
REDUCES FUSTRATION Allowing you to save time, fuel and associated cost
REDUCES THE SEARCHING TIME BY 35%
REDUCES ACCIDENTS
REDUCES CO² EMISSIONS Helping to improve the urban environment
DETECTS If a parked car has a special permit
Allows the development of PRICING STRATEGIES
SCALABLE, ROBUST AND RELIABLE

Figure 2: How does Fastpark work

1.2.1.1.2 Traffic Flow Monitoring

With the collection of data relating to traffic divergence, continuity, congestion and dispersion, real-time traffic analysis, transport operators develop a greater understanding of traffic flow dynamics. Heat maps show origin and destination (O/D) vectors based on travel times and enable operators to manage fleets and perfect their mobility strategy. An incident alert system provides instant updates regarding accidents or roadblocks making choosing the right transportation route a breeze.

[2]Smart Parking:
https://www.worldsensing.com/application-areas/smart-parking/
[3]Worldsensing's Fastpark:
https://www.worldsensing.com/product/fastprk/

Worldsensing's product line includes a traffic flow monitoring system which enables operators to track how vehicles move in real-time. It wirelessly collects traffic information which is vital for the development of agile mobility concepts. [4]

Bitcarrier

Bitcarrier is a system for real-time traffic flow and road monitoring. Vehicle movements are captured with a network of smart sensors, which are placed at strategic locations in city streets, on roads and on highways.

City and road operators wirelessly collect traffic information and can visualize and analyze results to better manage traffic flow. [5]

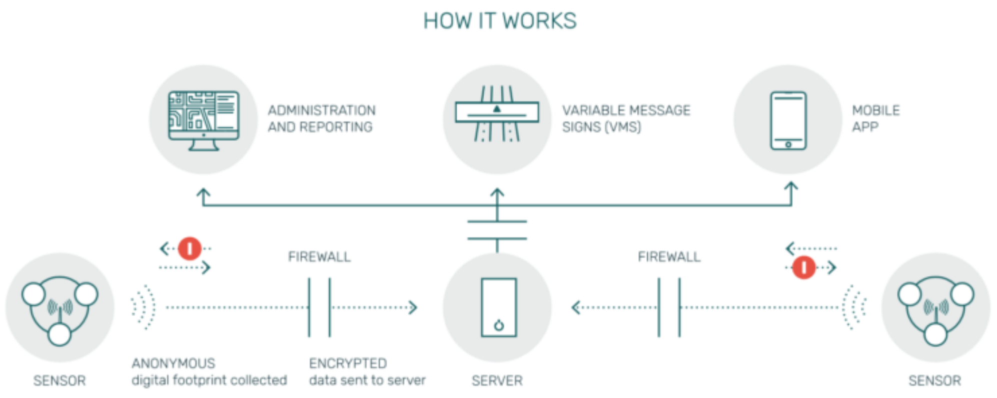

HOW IT WORKS
ADMINISTRATION AND REPORTING
VARIABLE MESSAGE SIGNS (VMS)
MOBILE APP
FIREWALL
FIREWALL
SENSOR
ANONYMOUS digital footprint collected
ENCRYPTED data sent to server
SERVER
SENSOR

Figure 3: How does Bitcarrier work

1.2.1.2 Construction Site Monitoring

Real-time monitoring of works enables operators to know a project's status and evolution at all times. Any problems can be detected before they become a danger. This helps to minimize the chances of costly changes or emergency operations, allowing operators to stick to terms and budget plans.

Worldsensing's product line includes a connected infrastructure solution which can be used to monitor how structures change over time. It's an ideal system to remotely monitor all critical stages of a construction project, from demolition and excavation through to high-risk and highly complex jobs. [6]

[4]Traffic Flow Monitoring:
https://www.worldsensing.com/application-areas/traffic-flow-monitoring/
[5]Worldsensing's Bitcarrier:
https://www.worldsensing.com/product/bitcarrier/
[6]Construction Site Monitoring:
https://www.worldsensing.com/application-areas/construction-site-monitoring/

Loadsensing

Loadsensing is the global leader for connecting and wirelessly monitoring infrastructures in remote locations. Construction and mining companies and operators of bridges, tunnels, dams, railways and many other inaccessible assets can now work with reliable data. Having access to this information and real-time insights enables operators to anticipate needs, manage their workforce, diminish risks, and even prevent disasters.

The wireless configuration also eliminates the need for manual monitoring and expensive cabling thus contributing to financial savings. [7]

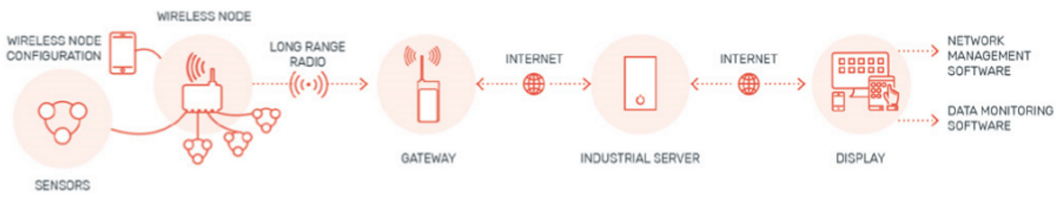


Figure 4: How does Loadsensing work

1.2.1.3 Critical Asset Monitoring

Critical infrastructure requires continuous monitoring to ensure early detection of failures and to enhance response times to prevent disasters. If critical assets such as dams and mines fail or function poorly this can have a severe impact on public health and safety.

From environmental changes to material fatigue and seismic activities like earthquakes, mines as well as dams, bridges, and tunnels, have a need for linear, structural health, and geotechnical monitoring to ensure safe environments. [8]

Spidernano

The Spidernano data acquisition system is a low-power Seismic Recording Unit suitable for a variety of geophysical applications within civil engineering, such as critical Structural Health Monitoring (SHM) or microseismic monitoring. Spidernano provides high-resolution data acquisition and it is easy to connect and operate due to its small form factor. Offering customizable data acquisition options and being compatible with most geophysical sensors, makes the recording unit extremely versatile.

The Spidernano monitoring unit can be used to measure and control induced vibrations generated during the construction of infrastructures, blasts, resource extractions and reservoir operations. Once deployed, it increases safety for contractors, insurances, and citizens.[9]

[7]Loadsensing:
https://www.worldsensing.com/product/loadsensing/
[8]Critical Asset Monitoring:
https://www.worldsensing.com/application-areas/critical-asset-monitoring/
[9]Worldsensing's Spidernano:
https://www.worldsensing.com/product/spidernano/

1.2.2 Innovation through European projects participation

Worldsensing's Innovation department participates in several projects funded by the *European Commission* which is a driver of the company. Its mission is to transform operations with data insights intelligent enough to enable real-time decision making. To achieve this goal and to stay in touch with the latest trends and technological developments, the company is part of several publicly funded research projects which are conducted in close collaboration with academia and European partners. Also participates in Catalan research and innovation projects.

Among the European research projects in which the company is part of, there are several of them that require investigating in sensoring, monitoring and data analysis, which will be introduced below, and more details will be shown in the upcoming sections.

- **European Projects**
  - *STOP-IT* (Strategic, Tactical, Operational Protection of water Infrastructure against cyber-physical Threats).
  - *SMESEC* (Cybersecurity for SMEs).
  - *mF2C* (Towards an open, secure, decentralized and coordinated Fog-to-Cloud Management ecosystem).
  - *BigIOT* (Bridging the Interoperability Gap of the Internet Of Things).
  - *<IMPACT>* (IMPACT connected car, impacting on the emerging connected car value chains)
- **Catalan Projects**
  - *RIS3CAT Utilities 4.0 Sènix* (Network Sensorisation and Inspection).
  - *RIS3CAT Activ4.0* (Operation and advanced asset management).

1.3 Problem formulation

1.3.1 What common needs arise?

Worldsensing focuses on creating, deploying, sensoring and monitoring IoT Solutions. The data workflow is relatively simple. It is based on, at most, storing data to be analyzed later. But there is no process implemented which once data is captured, it is integrated into a proper workflow that leads to items that can be actioned as the response to the data fluctuations.

Consequently, the company aims to create a system that could be capable of, automatically, read values from different kind of sensors: sensors from *Worldsensing*'s portfolio, as well as third-party external sources. Process and qualify these values according to some defined business rules and then, if these values reach a certain critical level, react accordingly to notify the predefined receivers such as third-party apps, government, etc. In other words, having an intelligent Warning System. This is the objective of the present work.

1.3.2 What is a Warning System?

A Warning System is any system of technical nature deployed by an individual or group to inform of a present or future danger. Its purpose is to enable the administrator(s) of the warning system to prepare for the danger and act accordingly to mitigate or avoid it. [25]

An integrated system of hazard monitoring, forecasting and prediction, disaster risk assessment, communication, preparedness activities systems and processes that enable individuals, communities, governments, businesses and others to take timely action to reduce disaster risks in advance of hazardous events. [21]

When the objective of a Warning System is to warn the population of a region, they are also called Public Warning System (PWS), so, we will refer as such to the platform that we will talk in this document from now on. These systems are needed to protect the lives of people in case of major emergency by warning the public of impending disasters.

Public Warning is the capability to bring to the immediate attention of all people who might be directly impacted following the onset, or predicted onset, of an emergency so that they can take action to mitigate the impact of this incident. For public warning there is no single solution that fits all requirements to reach all citizens in case of an emergency. Therefore, multiple technologies need to be considered. [5]

For the public authorities, warning the population on the occurrence of a tentative disaster is one of their responsibilities. They will use for this purpose all means of communication, in relation with the specific features of the disaster (e.g. the level of risks, if it can be forecasted or not, if the coverage is limited or broad, among others). [16]

These warnings can be generated by different causes, such as natural, meteorological, structural, ... But also can be human-induced like mobility or terrorism. The table below shows some examples of the principal elements or causes of the warnings.

Natural causes		**Human-caused**
Meteorological Geological	**Technology Structural**	
Flood, Drought	Hydroelectric plants in dams	Building or structure collapse
Tornadoes, Hurricanes	Chemical plants	Road incidents
Extreme temperatures	Nuclear facilities	Cyber-Attacks
Tsunamis	Critical infrastructures	Terrorists
Volcanoes		

Table 1: Different kind of warning triggers[10]

The time it takes to communicate critical information in an emergency can mean the difference between safety and catastrophe. The ability to accurately deliver the right information, to the right audience, at the right time is crucial to any emergency planning effort.

Early warning systems must be understood as working systems with interconnected components (see figure below). A weakness or failure in any component of the system (technical or human/organizational) can potentially undermine the whole platform.

[10]Federal Emergency Management Agency - Emergency Response Plan:
https://www.fema.gov/media-library-data/1388775706419-f977cdebbefcd545dfc7808c3e9385fc/Business_EmergencyResponsePlans_10pg_2014.pdf

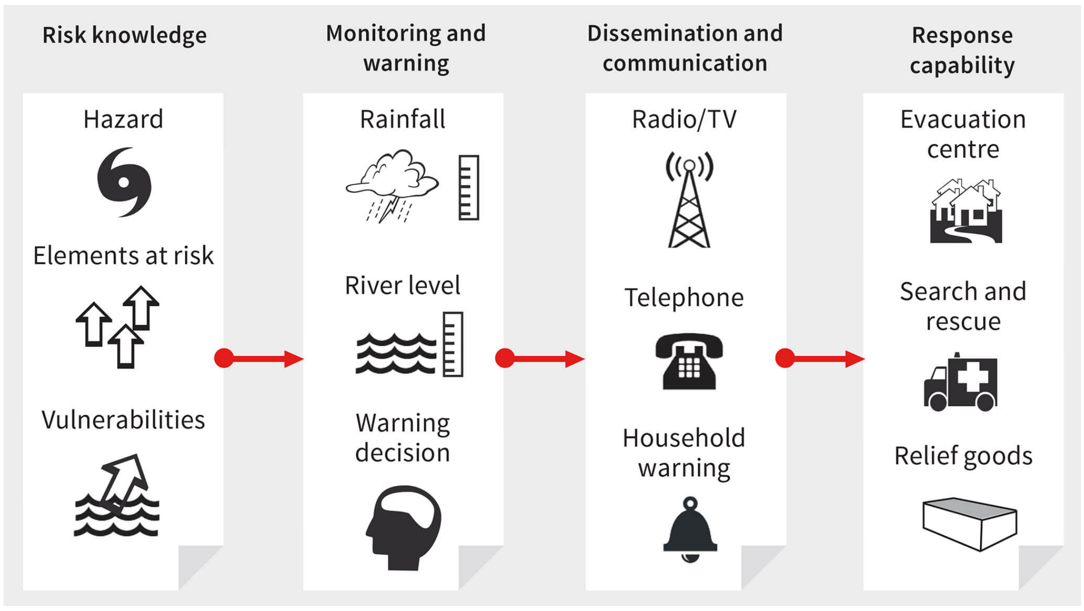


Figure 5: Actual idea of a Warning System[11]

Early warning is a major element of disaster risk reduction. Early action can often prevent a hazard from turning into a human disaster and preventing loss of life and reducing the economic and material impacts. To be effective and sustainable they must actively involve the communities at risk. [3]

1.4 Requirement Analysis

1.4.1 Product Areas

Section 1.2.1 explains the *Application Areas and Solutions in Worldsensing*. So, in alignment with that, the Public Warning System should ideally provide the following features:

1. As *raw data* input, integrate it from different sources.
2. As *possible alarm* input, also integrate different sources.
3. Have the ability to assess whether some data is an alert.
4. As *output*, have different ways to send/notify the alerts.
5. Almost all the data collected from the input should be treatable for a further analysis.

[11]Source: https://goodpracticereview.org/9/preparing-for-disasters-and-emergencies/forecasting-and-warning/

1.4.2 Innovation - European Projects

As it was introduced earlier in this document (in Section 1.2.2), *Worldsensing* participates in several European and Catalan projects. Each one of them has something in particular that is different among the others. But all of them, in broad strokes, share the core need of a central service capable of treating data, processing it and if its needed, reacting to it.

The following image shows the European projects involved in the project.

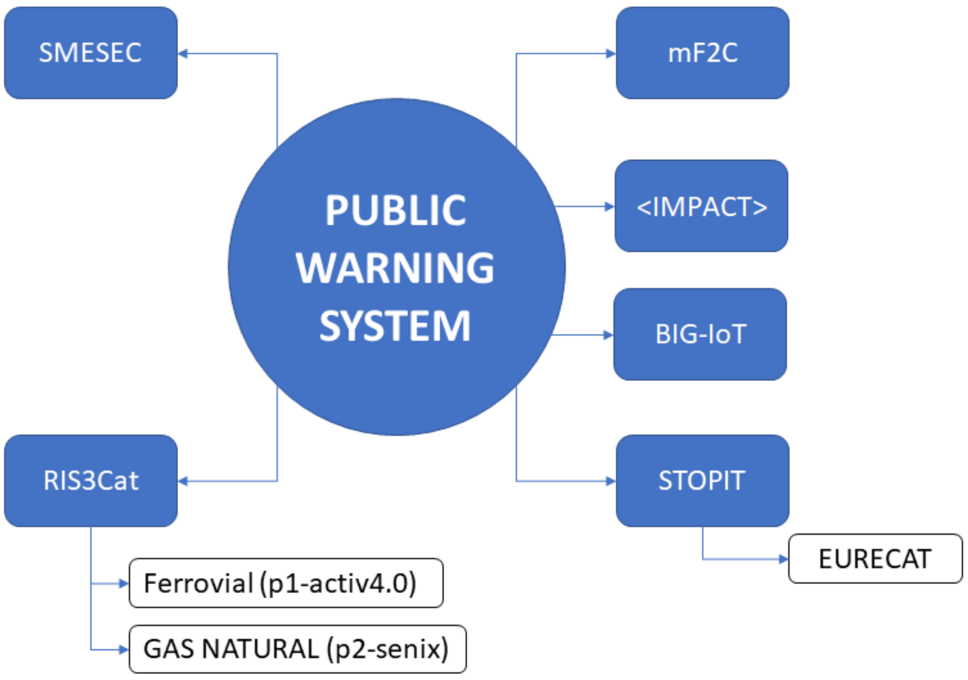


Figure 6: Projects in which this Public Warning System is involved

In the following section, it will be explained which are the requirements for every project in relation with the developments done in the frame of this Master Thesis.

1.4.2.1 STOP-IT

European Project Name: Strategic, Tactical, Operational Protection of water Infrastructure against cyber-physical Threats. [12]

Programmes: H2020-EU.3.7.2. - Protect and improve the resilience of critical infrastructures, supply chains and transport modes. H2020-EU.3.7.4. - Improve cyber security.

Water Critical Infrastructures (CIs) are essential for human society, life and health and they can be endangered by physical/cyber threats with severe societal consequences. STOP-IT assembles a team of major Water Utilities, industrial technology developers, high tech Small-Medium Sized Enterprises (SMEs) and top EU R&D providers. It organises communities of practice for water systems protection to identify current and future risk landscapes and to co-develop an all-hazards risk management framework for the physical and cyber protection of water CIs. **Prevention, Detection, Response and Mitigation of relevant risks** at strategic, tactical and operational levels of planning will be taken into account to generate modular solutions (technologies, tools and guidelines) and an integrated software platform.

1.4.2.1.1 Public Warning System - Expected Contribution

STOP-IT expects a PWS solution which is able to facilitate two way communication for incidents and warning messaging.

The first area is covering the input flow of external data for the STOP-IT solution. We are required to acquire data to inform about an incident from third parties and report it to the internal risk management system.

The second area is alerting users/citizens about an warning situation. The solution is required to contact user groups which are assumed to take some actions according the warning situation, inform actions due to situations, to a citizens so they can take protection actions, etc.

The solution should be able to differentiate technical operators, authorities, citizens, and alert them by the most appropriate channel for each of them and with the level of detail required.

1.4.2.1.2 Public Warning System - Use Case Validation

The PWS solutions delivered to STOP-IT will be validated in water infrastructure operators, mainly having the water treatment, distribution and commercialization businesses as its main operations.

1.4.2.1.3 Public Warning System - Standards applicable

The PWS solutions delivered to STOP-IT will comply with the most appropriated standards to the system. Following it is shown a list of the appropriated standards to comply with. For every standard, it is indicated the status of the implementation of it for the project.

OASIS Common Alerting Protocol v1.2 - Defined and starting with the implementation, specification of the standard can be found in Section 2.1.1.
http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.html

OGC Standards for sensors - SensorThings API - Implemented, specification of the standard can be found in Section 2.1.2.
http://www.opengeospatial.org/standards/sensorthings

Sensor Observation Service - To be implemented
http://www.opengeospatial.org/standards/sos

1.4.2.2 SMESEC

European Project Name: Protecting Small and Medium-sized Enterprises digital technology through an innovative cyber-SECurity framework. [11]

Program: H2020-EU.3.7.4. - Improve cyber security.

Small and Medium size Enterprises (SMEs) are an important driver for innovation and growth in the EU. SMEs also stand to gain the most from innovative technology, because it is complicated and costly for them to set-up and run Information and Communication Technologies (ICT) in the traditional way. Taking into account cyber-security, SMEs do not always understand all the risks and business consequences for the development of technologies without the adequate level of protection against cybercrime.

SMESEC consortium is proposing to develop a cost-effective framework composed of specific cyber-security tool-kit to support SMEs in managing network information security risks and threats, as well as in identifying opportunities for implementing secure innovative technology in the digital market.

1.4.2.2.1 Public Warning System - Expected Contribution

SMESEC is requesting the filtering of the detected cyber alerts (both from individual tools and integrated alerts), to decide whether the alert should be elevated to other systems or it is a false alarm. There is a requirement to define several user roles, and manage them, so the different roles can access and perform different actions. The solution requested for SMESEC is to receive information from the cybersecurity solution (using standard interfaces) and provide an environment to define (design) the business rules that are applicable to the different use cases.

- Data Ingestion from:
  - Antivirus - *Bitdefender*
  - SIEM - *ATOS*
  - Tiltmeter - *Worldsensing*
- Business Rule Management
- Alert Reporting

1.4.2.2.2 Public Warning System - Use Case Validation

Gateway seems to report abnormal data streams. The PWS should warn the upper level to be aware of abnormal data, and is the upper level who decides whether they keep ingesting the data streams or they stop them as it is believed that the gateway has been hacked or the data stream has been forged.

1.4.2.2.3 Public Warning System - Standards applicable

The PWS solutions delivered to SMESEC will try to adapt the following standards to the system. For every standard, it will be a status of the adaptation of it for the project.

MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing - To be implemented, specification of the standard can be found in Section 2.1.4.
https://www.misp-project.org/

STIX - A structured language for cyber threat intelligence - To be implemented
https://oasis-open.github.io/cti-documentation/

1.4.2.3 mF2C

European Project Name: Towards an Open, Secure, Decentralized and Coordinated Fog-to-Cloud Management Ecosystem. [9]

Program: H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT).

Fog computing brings cloud computing capabilities closer to the end-device and users, while enabling location-dependent resource allocation, low latency services, and extending significantly the IoT services portfolio as well as market and business opportunities in the cloud sector. With the number of devices exponentially growing globally, new cloud and fog models are expected to emerge, paving the way for shared, collaborative, extensible mobile, volatile and dynamic compute, storage and network infrastructure. When put together, cloud and fog computing create a new stack of resources, which we refer to as Fog-to-Cloud (F2C), creating the need for a new, open and coordinated management ecosystem.

The mF2C proposal sets the goal of designing an open, secure, decentralized, multi-stakeholder management framework, including novel programming models, privacy and security, data storage techniques, service creation, brokerage solutions, SLA policies, and resource orchestration methods.

The proposed framework is expected to set the foundations for a novel distributed system architecture, developing a proof-of-concept system and platform, to be tested and validated in real-world use cases, as envisioned by the industrial partners in the consortium with significant interest in rapid innovation in the cloud computing sector.

1.4.2.3.1 Public Warning System - Expected Contribution

The first proposed use case is an alarm manager for smart infrastructure monitoring service. The use case will include:

1. Decision-making service according to an inclination sensor that monitors the health status of selected infrastructures.
2. Tool for the Emergency Situation Management in a Smart City which processes information and triggers the intervention of the relevant emergency services

This proposed infrastructure monitoring use case aims to validate a novel and innovative hybrid architecture that serves:

1. Analyse flows of assets within Smart Cities (i.e., indoor or/and outdoor positioning) in order to provide useful information to citizens and authorities.
2. Detect a possible emergency in real-time.
3. Decrease the necessary resources in terms of energy, latency, etc. with the aim to respond to specific situations in accordance to the application's requirements.

The minimum functionalities are not fully defined in the use case. A *tiltmeter* is being monitored. When a threshold is reached, an alert is launched and the system evaluate which field operator is the closest so he can be asked to validate the status, and/or apply remedial actions.

1.4.2.3.2 Public Warning System - Use Case Validation

The list of actions of the storyline is the following:

1. Localizers send location of workers, displayed on *OneMind* dashboard and map.
2. Tiltmeter sends data to Gateway, that forwards to Monitoring Software.
3. Monitoring software shows data on *OneMind* dashboard and maps.
4. Monitoring software detects that the *Loadsensing Tiltmeter* threshold was exceeded.
5. Not confirmed alerts visualized on dashboard and maps in real time.
6. Closest worker identified and contacted.
7. Closest worker reports confirmation or cancellation.
8. Confirmed alerts visualized on dashboard and maps in real time.
9. Response plan activated.
10. Authorities and relevant actors alerted.

1.4.2.3.3 Public Warning System - Standards applicable

For now, there is no need of applying a particular standard.

1.4.2.4 <IMPACT>

European Project Name: <IMPACT> Connected Car. [8]

Programmes: H2020-EU.2.3.2.2. - Enhancing the innovation capacity of SMEs. H2020-EU.2.3.2.3. - Supporting market-driven innovation.

<IMPACT> Connected Car will fund 2 European Superstars and 64 Disruptive SMEs in the Connected Car OpenSpace by 5 European Cluster and 2 global ecosystems with the support of 10 Global Corporations and 4 pioneering startups.

An acceleration and smartization virtuous cycle methodology in the OpenSpace with vehicle, infrastructure and device & TelCo interactions and consumer & business services across 6 industrial value chains (Mobility, Automotive, Electronics, ICT, Services and Infrastructure) will lead to Value Link-Chains.

Beyond the 4 million euro support for the SMEs within the project, <IMPACT> Connected Car will seek private funding of 4 million euros and 1 million euros of mobilized ESIF. For this purpose, the "<IMPACT> Connected Car Label" will be instrumental.

<IMPACT> Connected Car will demonstrate that public clusterization and disruption discovery with key Innovation actors (SMEs) and proper corporate engagement develops Emerging Industries and builds the EU of ”entrepreneurial states”.

1.4.2.4.1 Public Warning System - Expected Contribution

In <IMPACT>, *Worldsensing* will develop and present a Public Warning System to alert mobility agents of potential parking occupation infringements. Such system is known as a *Parking Enforcement System.*

1.4.2.4.2 Public Warning System - Use Case Validation

In the case of use for this European project it will be about showing the current Fastpark system together with the OneMind dashboard. With the added improvement of including the enforcement system, and being able to generate alerts automatically when an irregular situation is detected.

1.4.2.4.3 Public Warning System - Standards applicable

For this *European Project* there is no need of developing or implementing an standard.

1.4.2.5 BIG IoT

European Project Name: BIG IoT - Bridging the Interoperability Gap of the Internet of Things. [7]

Program: H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT)

The objective of the BIG IoT project is to ignite really vibrant Internet of Things (IoT) ecosystems. We will achieve this by bridging the current interoperability gap between the vertically integrated IoT platforms and by creating marketplaces for IoT services and applications. Despite various research and innovation projects working on the Internet of Things, no broadly accepted professional IoT ecosystems exist. The reason for that are high market entry barriers for developers and service providers due to a fragmentation of IoT platforms.

The goal of this project is to overcome these hurdles by Bridging the Interoperability Gap of the IoT and by creating marketplaces for service and application providers as well as platform operators. We will address the interoperability gap by defining a generic, unified Web API for smart object platforms, called the BIG IoT API. The establishment of a marketplace where platform, application, and service providers can monetize their assets will introduce an incentive to grant access to formerly closed systems and lower market entry barriers for developers.

1.4.2.5.1 Public Warning System - Expected Contribution

For this *European Project*, the role of *Worldsensing* is to implement a similar use case, the *Parking Enforcement System* presented within the <IMPACT> *European Project*, but in this case the alerts data should be consumed from an integration of the *BIG IoT API* library and not from our own system. More information can be found in Section 2.1.3.

1.4.2.5.2 Public Warning System - Use Case Validation

In the case of use for this European project it will be about showing the current Fastpark system together with the OneMind dashboard. With the added improvement of including the enforcement system, and being able to generate alerts automatically when an irregular situation is detected.

In addition, this system must collect data from the BIG-IoT library, rather than using the corporate system itself.

1.4.2.5.3 Public Warning System - Standards applicable

BIG IoT API - Implemented, specification of the standard can be found in Section 2.1.3.
https://big-iot.github.io/

1.4.2.6 RIS3CAT

Catalan Project Name: Research and Innovation Strategy for the Smart Specialisation of Catalonia (RIS3CAT) [6] [10]

The Research and Innovation Strategy for the Smart Specialisation of Catalonia (RIS3CAT) is the response of Catalonia to the European Commission's demand that the states and regions of the European Union develop research and innovation strategies for to smart specialisation (RIS3) that adjust to their innovation potential.

With the challenge of strengthening the links and collaboration between the R+D+I system and the business fabric, RIS3CAT establishes the priorities of public policies for research, development and innovation (R+D+I) and the actions that will be supported by the FEDER Catalunya 2014-2020 Operational Program, in accordance with four strategic objectives, which correspond to four axes of action.

1. To modernise the business fabric by improving the efficiency of production processes, internationalisation and the reorientation of consolidated sectors towards activities with greater added value.
2. To promote new emerging economic activities through research and innovation to create and develop new market niches.
3. To consolidate Catalonia as a European knowledge hub and link technological and creative capacities to existing and emerging sectors in the territory.
4. To improve the overall Catalan innovation system, increasing the competitiveness of companies and steering public policies towards promotion of innovation, internationalisation and entrepreneurship.

The four pillars of action are: Leading sectors; Emerging activities; Cross-cutting enabling technologies; Innovation environment.

1.4.2.7 P1-Activ.4.0

The following section is in Spanish because it is the only official language available.

El objetivo general que se aborda en este proyecto es la transformación de las redes e infraestructuras proveedoras de sevicios (que denominamos "Utilities") a través de la implantación y adaptación del nuevo concepto de Industria 4.0, (Utilities 4.0 en el caso de los servicios) y en el análisis de su impacto. Este objetivo general, que denominaremos "Utilities 4.0" y que será la proyección del concepto "Industria 4.0", pretende la investigación en nuevas tecnologías habilitadoras hacia la digitalización, sensorización y monitorización de las redes proveedoras de servicios de acuerdo con este nuevo concepto.

Para alcanzar este objetivo general, el proyecto afrontará la investigación de nuevas tecnologías habilitadoras hacia la digitalización, sensorización y monitorización de los activos y procesos asociados a las "Utilities". Los conocimientos adquiridos serán el vehículo para el desarrollo de soluciones avanzadas que permitan la optimización de la operación de las plantas y redes en términos de eficiencia y costes operativos, así como la mejora de los procesos de mantenimiento con el fin de minimizar tanto las incidencias como los costes de inversión y renovación de los activos.

En resumen, se trata de investigar y desarrollar las tecnologías que permitan 1) la interoperabilidad entre máquinas, sensores y personas a través del Internet de las cosas (IoT) o de las personas (IoP) y la capacidad de transformar datos en información útil y fiable, 2) la agregación y visualización de información actual o predecible para tomar decisiones informadas y resolver problemas urgentes con poca antelación o prevenir su aparición, 3) la capacidad autónoma para la toma de decisiones de los sistemas físicos cibernéticos o la delegación a un nivel superior mediante optimización multicriterio en caso de conflicto, 4) la aplicación de la tecnología BIM (Building Information Modelling) como herramienta integradora de información relevante para el diseño, operación y mantenimiento de las instalaciones a partir de modelos avanzados de representación 3D.

1.4.2.7.1 Public Warning System - Expected Contribution

A solution that allows the identification of anomalies produced in the environment of a power production plant (from natural gas).

Management of anomalies in an integrated way to build a robust incident reporting system, allowing actuation from the situations that were detected. A system that will integrate sensor and operational data from the plant in order to detect anomalies, categorise risks and allow the agents involved to act and report as quickly as possible.

1.4.2.7.2 Public Warning System - Use Case Validation

Use of new noise and image sensors for predictive detection of operational incidents in combined cycle plants and distribution assets.

The availability of new high-resolution, low-cost sensors for capturing noise and images opens up the possibility of integrating local intelligence for the detection of multiple incidents in equipment and installations in parallel with existing scada instrumentation and systems. The objective is to improve predictive maintenance and reduce unscheduled downtime in assets and plants, improving service quality and reducing costs.

Artificial intelligence and self-learning algorithms will be developed from new generation commercial sensors that will allow these sensors to recognise and foresee the most common incidences and anomalies. These smart sensors could be massively implemented and act as sensor networks with wireless communication and correlation of their data centrally in a second phase.

Introduce predictive incident detection systems that can be used on a massive scale (low cost and easy installation) in the three areas of application proposed by the project for asset management.

1.4.2.7.3 Public Warning System - Standards applicable

The PWS solutions delivered to P1-Activ.4.0 will comply with the most appropriated standards to the system. Following it is shown a list of the appropriated standards to comply with. For every standard, it is indicated the status of the implementation of it for the project.

OGC Standards for sensors - SensorThings API - Implemented, specification of the standard can be found in Section 2.1.2.
http://www.opengeospatial.org/standards/sensorthings

1.4.2.8 P2-SENIX

The following section is in Spanish because it is the only official language available.

Generación y el desarrollo de soluciones avanzadas basadas en las tecnologías investigadas, que permitan la optimización de la operación de las plantas y redes en términos de eficiencia, costes operativos y de mantenimiento y costes de inversión y renovación y el desarrollo de pilotos sobre casos de uso concretos para aplicaciones en entornos relacionados con las Utilities, en sinergia con el proyecto P1 y el estudio de su impacto económico y social.

En este proyecto en particular, se pretende transformar las Utilities con el objetivo final de conseguir una gestión óptima de sus redes e infraestructuras, lo que revertirá en última instancia en un mejor servicio al cliente/ciudadano.

1.4.2.8.1 Public Warning System - Expected Contribution

A solution for monitoring water and gas networks (distributed environment), with LP-WAN sensors to improve the service and detect possible fraud.

Detection and management of anomalies in an integrated way to build a robust incident reporting system, allowing action from the situations detected.

Obtaining data that allows the management of the network according to the KPIs indicated by the operators. Detection of fraud in a reasonable time (based on business rules defined by the operators).

1.4.2.8.2 Public Warning System - Use Case Validation

Monitoring of gas and water networks with flow and pressure sensors. Availability of solid state sensors for measuring flow and pressure in gas and water distribution networks that can be massively implemented and fed by harvesting. The objective is to have network operation data at multiple points, downstream of the remotely managed ERMs, which will allow both network management and fraud detection to be improved.

1.4.2.8.3 Public Warning System - Standards applicable

The PWS solutions delivered to P2-SENIX will comply with the most appropriated standards to the system. Following it is shown a list of the appropriated standards to comply with. For every standard, it is indicated the status of the implementation of it for the project.

OGC Standards for sensors - SensorThings API - Implemented, specification of the standard can be found in Section 2.1.2.
http://www.opengeospatial.org/standards/sensorthings

1.5 Scope of the project

The objective of the project is to have the basis, or the first iteration, of a Public Warning System for *Worldsensing* (note below)*. The following image shows the first iteration overview of a PWS. But keep in mind that we have iterated from this image, however, this image shows a clear vision of what this project wants to achieve.

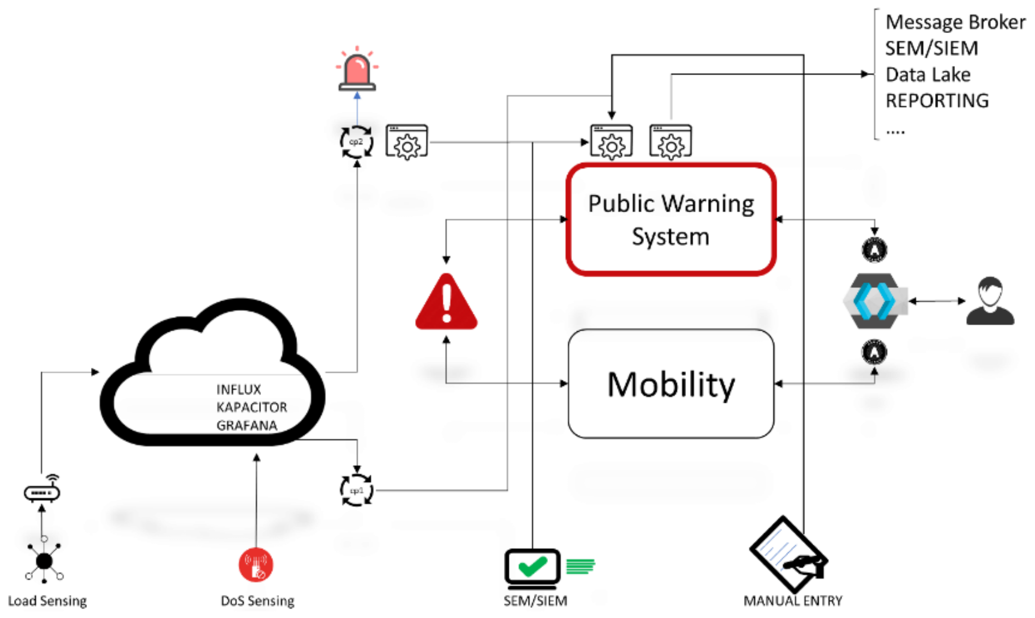


Figure 7: High level overview of the PWS

This Public Warning System, will be a platform to manage the lifecycle of the alerts and incidents coming from different sources, as well as the raw data received from the sources itself. Processing and filtering them in order to generate actionable items for those alerts.

Most of the input channels will come from IoT devices, either directly from a *Worldsensing* device or a third-party sensor or either indirectly from a cloud-based backend that pre-processes the data before it will be consumed by the Public Warning System. Also, this system will be capable of consuming alerts directly from different sources.

Then, this data will be processed in the system in order to generate *tentative alerts*, which after filtering, will be validated to generate actionable items such as email or SMS sending, or even reporting to a third-party collaborator.

***Note**: This Public Warning System will be developed using Agile methodologies (SCRUM, to be more precise, Section 3.1) and consequently will go through several iterations that will be grouped in larger time-cycles called releases. It is not intended that a single engineer will be able to deal with all the required workload or know all the technologies involved in such a development. That is why in this master thesis what will be done is the basis to continue with the work in the following months.

In the same way, several European and Catalan projects involved in the project have been explained. But some of them will only be specified in this master's thesis for further development. Therefore, the list of the status of these projects when finished the master's thesis will be shown:

- **STOP-IT**: Specified, implemented and delivered a first release.
- **SMESEC**: Specified a first release.
- **mF2C**: Specified a first release.
- **IMPACT**: Specified, implemented and delivered totally.
- **BIGIOT**: Specified, implemented and delivered totally.
- **RIS3CAT**: Preparing the specification for a first release.

2 State of Art

Early warning technologies appear to be mature in certain fields but not yet in others. Considerable progress has been made, thanks to advances in scientific research and in communication and information technologies. Nevertheless, a significant amount of work remains to fill existing technological, communication, and geographical coverage gaps. Early warning technologies are now available for almost all types of hazards, although for some of them (such as droughts and landslides) these technologies are still less developed.

Most countries have or seem to have early warning systems for disaster risk reduction. However, there is still a technological and national capacity gap between developed and developing countries. From an operational point of view, some elements of the early warning process are not yet mature. In particular, it is essential to strengthen the links between sectors involved in early warning, including organizations responsible for issuing warnings and the authorities in charge of responding to these warnings, as well as promoting good governance and appropriate action plans. [2]

Current generation PWS are extremely basic, consisting mostly of sirens and alerting over radio and TV. Modern sirens have the capability to generate sounds with different tones, but it is not used everywhere, since citizens may not remember, or even know, the meaning of the different tones.

The use of radio and TV are components in the mix of PWS technologies, but radio and TV are only capable of reaching citizens when they are listening or viewing. [5]

2.1 Standards

The following sections explain existing standards that could be applicable to a Public Warning System.

2.1.1 OASIS - Common Alert Protocol

The Common Alerting Protocol (CAP) standard is an internationally recognized standard for information exchange published by the Organization for the Advancement of Structured Information Standards (OASIS). OASIS is a global nonprofit consortium that works on the development, convergence, and adoption of open standards for security, Internet of Things, energy, content technologies, emergency management, and other areas.

The CAP standard is a simple but general format for exchanging all-hazard emergency alerts and public warnings over all kinds of networks. CAP allows a consistent warning message to be disseminated simultaneously over many different warning systems, thus increasing warning effectiveness while simplifying the warning task. CAP also facilitates the detection of emerging patterns in local warnings of various kinds, such as might indicate an undetected hazard or hostile act. And CAP provides a template for effective warning messages based on best practices identified in academic research and real-world experience. [1]

A generalised *Document Object Model* for a *CAP* message is illustrated below.

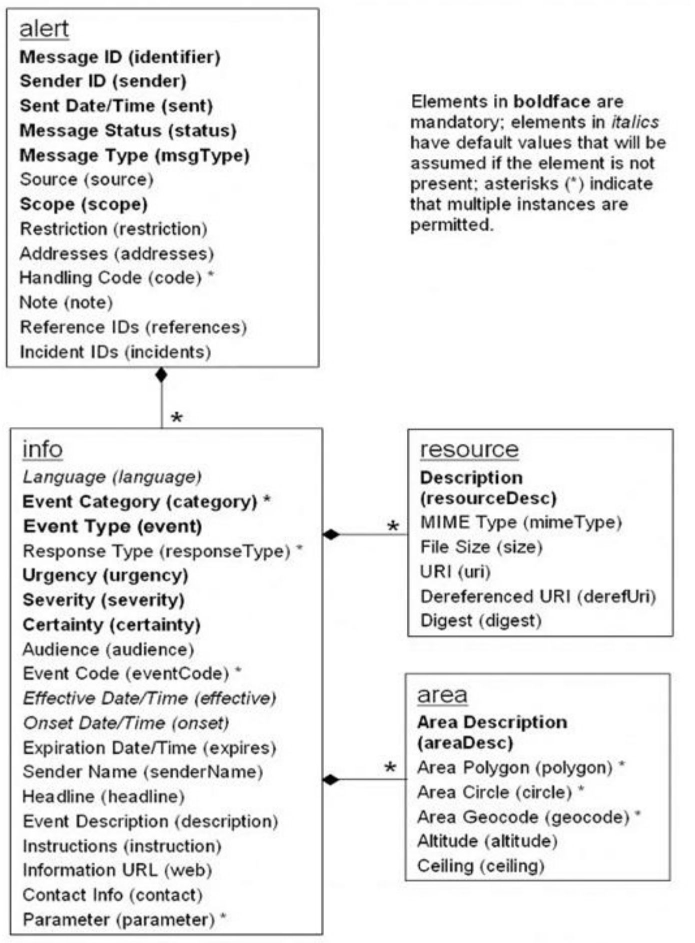


Figure 8: CAP Document Object Model[12]

CAP is a set of common published eXtensible Markup Language (XML) tags that allow the formatting of messages in a common and open format that can then be used by various systems. CAP allows the user to format messages and add links to other information such as audio, video, and pictures.

2.1.2 OGC - SensorThings

The Open Geospatial Consortium (OGC) is an international nonprofit organization committed to making quality open standards for the global geospatial community. These standards are established through a consensus process and are freely available for anyone to use to improve sharing of the world's geospatial data. OGC standards are technical documents that detail interfaces or encodings. Software developers use these documents to build open interfaces and encodings into their products and services.

[12]OASIS CAP:
http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.html

OGC has more than 50 standards, and implementing a solution per standard is not possible since it could take a lot of time. Therefore, a small research was conducted by our team to see which standard could be the most suitable one, and decided to implement the SensorThings standard in our project.

The OGC SensorThings API provides an open, geospatial-enabled and unified way to interconnect Internet of Things (IoT) devices, data, and applications over the Web. Provides a standard way to manage and retrieve observations and metadata from heterogeneous IoT sensor systems. [15]

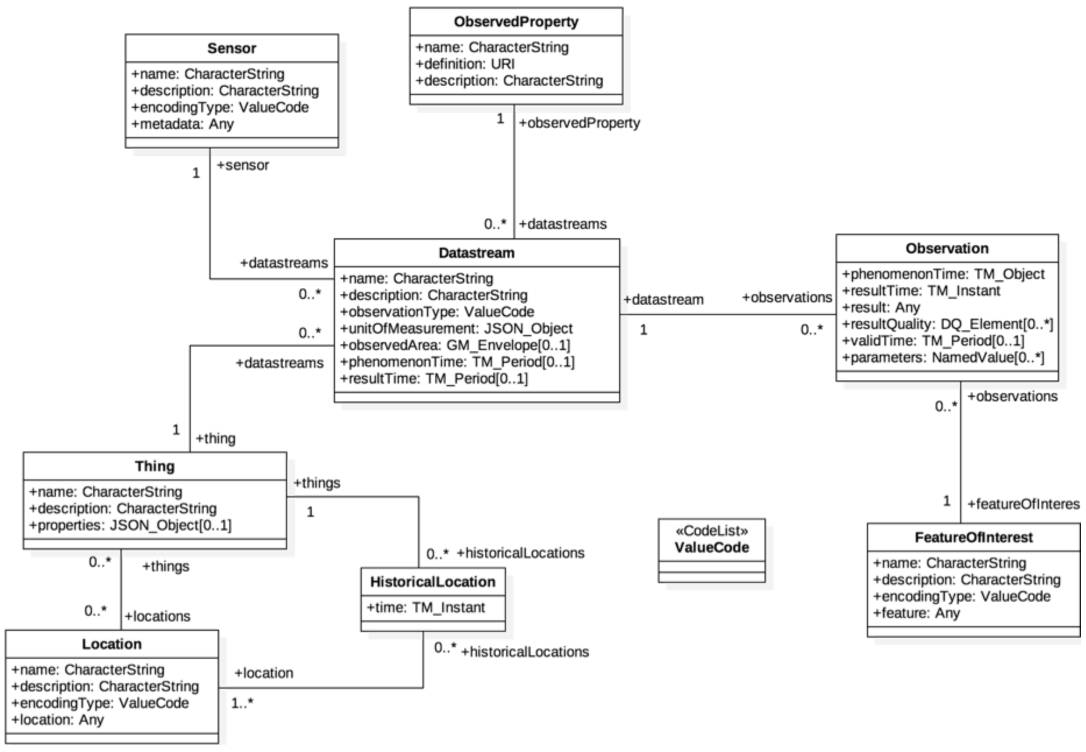


Figure 9: Sensing Entities in OGC SensorThings API[13]

2.1.3 BIG IoT API - Bridging the Interoperability Gap of the Internet of Things

Using the *BIG IoT API*, currently as a library, IoT platforms and services can offer their resources in a generic way, so anyone can use them. Using the same API lib, IoT applications (but also services) can consume those resources very easily. In this way, the *BIG IoT API* lib solves today's interoperability issues between IoT providers and consumers.

Once a platform or service is using the lib, their resources can be registered as offerings on the *BIG IoT Marketplace*. For example, an IoT platform could offer data on the status of parking spots (occupied or available) within a city. And an IoT service could provide the functionality to reserve resources. An application could access both with the *BIG IoT API* and combine them to enable reservation of parking spots.

[13]Open Geospatial Consortium SensorThings API: http://docs.opengeospatial.org/is/15-078r6/15-078r6.html

The figure below illustrates the *BIG IoT* approach. All registered platforms and services are discoverable on the Marketplace. The API gives the functionality (discovery, data access, security, etc) to easily build IoT applications on top of this ecosystem. [17]

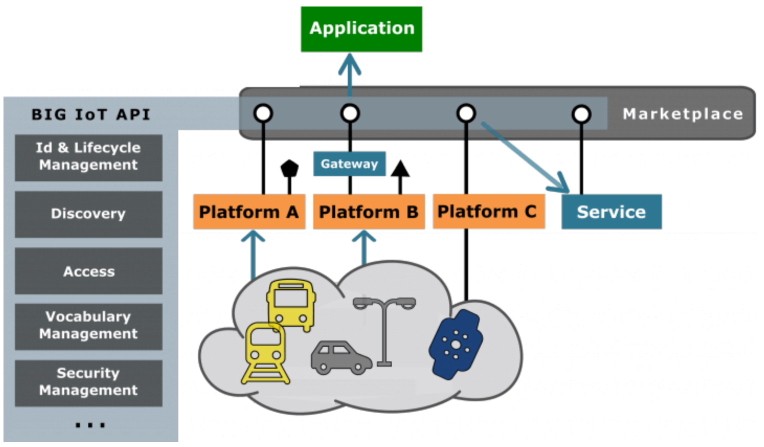


Figure 10: BIG IoT API library overview[14]

2.1.4 CIRCL - Malware Information Sharing Platform

Malware Information Sharing Platform (MISP) is an open source software solution for collecting, storing, distributing and sharing cyber security indicators and threat about cyber security incidents analysis and malware analysis. It is developed as free software by a group of developers from CIRCL but also the Belgian Defence and NATO (North Atlantic Treaty Organization).

MISP is designed by and for incident analysts, security and ICT professionals or malware reverser to support their day-to-day operations to share structured informations efficiently. The objective of MISP is to foster the sharing of structured information within the security community and abroad. [19]

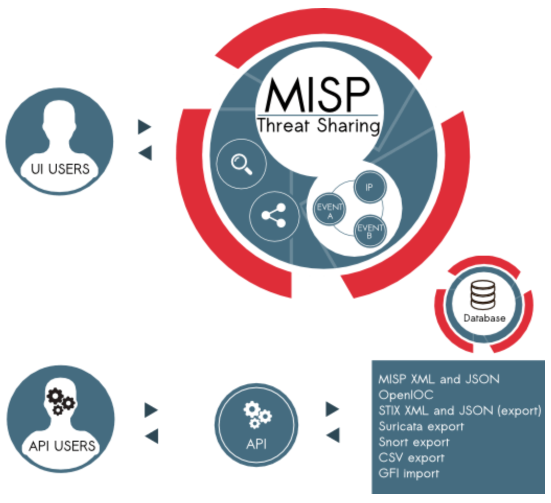


Figure 11: How does MISP work?[15]

[14]BIG IoT API Library:
https://big-iot.github.io/

2.2 Technological context analysis

Before starting to specify and build our system, it is convenient to search the results of previous methods to obtain relevant information for our project. With this information, we can verify if our project will really contribute something that stands out from the existing proposals and if it is possible to take advantage of some existing ideas or to see the shortcomings of the applications and to avoid them.

Below we will show some methods/systems/solutions that we have analysed, grouped by mode of work and selected by similarity to our project.

2.2.1 By transmission mode

First, we will analyze from a generic perspective the traditional warning system options, as well as how new technologies are slowly being adopted in this field.

2.2.1.1 Based on Telephony

Cell Broadcast

Cell Broadcast (CB) is a technology that has a similar user experience to SMS, text messages are displayed on the screen of the mobile device. However, the technology that is used to send the message to the mobile phone differs between the two services. Whereas SMS is a point-to-point service, CB is a point-to-multipoint service, a broadcast service.

SMS based

One of the most obvious advantages of using SMS is that it works on any handset that can receive traditional SMS. No handset changes are required.

On the other hand, traditional SMS is neither location based nor, due to network issues, suitable for alert purposes and cannot be prioritised. There are solutions today that enable the location-based capability besides using the network in a more efficient way, while still delivering the message to the handset as traditional SMS.

APP based

Location-based SMS and Cell Broadcast solutions require the mobile operator to deliver components of the PWS solution. Mobile Application based solutions bypass this involvement; the mobile operator merely acts as the ordinary bit-pipe.

Mass alerting via apps is often a feature of a 112-app. When citizens install a 112-app on their mobile phone, they implicitly agree to provide information about their location to the app service provider. This allows the app provider to send location-based alert messages to the app.

It should be considered, that potential cybersecurity issues (denial of service attacks) might impact system responsiveness under some circumstances since the availability of the system depends on public resources.

[15] Malware Information Sharing Platform:
https://www.circl.lu/services/misp-malware-information-sharing-platform/

2.2.1.2 Based on TV / Radio

Insertion in broadcast signal: In this case, the emergency notification platform is connected to a gateway located after the signal output from the relevant TV station adding a "super title" slide to the existing TV signal.

Insertion into set-top boxes: With digital video broadcast (more precisely multi-cast) the information is sent by the alert platform to the network service provider and from there to all the set-top-boxes in the specific area of the target area.

2.2.1.3 Based on Sirens

Sirens are an effective warning system for outdoor use specially in areas with special warning needs such as dams, chemical plants, harbours, etc.

2.2.1.4 Based on Internet - Social Media

Increasingly, emergency services have incorporated social media in their communication plans and actively disseminate alerts and warnings via existing accounts on major social networking platforms like *Twitter* or *Facebook*. In this regard warning systems shall consider social media in their concepts as additional means to reach-out to specific citizen groups.

2.2.2 Governments with deployed Public Warning Systems

Next, we will analyse solutions that are being applied and in use by government organisations. First, indicating those that follow a traditional design of warning systems, and then those that use modern methods.

2.2.2.1 Traditional solutions

In this section, the Systems that are considered as traditional are the ones which use radio, cell broadcasting, or television to send warning messages. Hereinafter they will be sorted by geographical area in alphabetical order. [24]

Australia - Standard Emergency Warning Signal - SEWS

Warning alarm used to alert the public of danger. The siren is played over television and radio.
Source: https://www.dfes.wa.gov.au/safetyinformation/warningsystems/Pages/sews.aspx?info

Belgium - BE-Alert

Multi-channel public warning mechanism to alert the population in case of a crisis. The warnings are sent through SMS, automated voice call or e-mail.
Source: http://www.be-alert.be/en

Canada - National Public Alerting System - Alert Ready

Warning alarm used to alert the public of danger about weather or public safety emergencies. The alarm is played over television, radio and phone communication.
Source: https://www.alertready.ca/

Chile - Emergency Alert System - EAS

Warning system used to alert the public of danger. The alert is played over sirens, but also through radio, TV and SMS.
Source: http://www.onemi.cl/tipos-de-alertas/

Czech Republic - Unified System of Warning and Information

Consists of information centres, a data network, radio networks and warning, information and measuring terminal devices. The USWI terminal devices are rotary sirens, electronic sirens and local radio systems.
Source: https://eena.org/wp-content/uploads/2018/11/Public-Warning-updated-version.pdf

European Union - EU-Alert

European Public Warning Service using the Cell Broadcast Service as a means of delivering public warning messages to the general public.
Source: https://en.wikipedia.org/wiki/EU-Alert

France - Air Raid Siren

Warning siren used to alert the public of danger. The siren is played over electromechanical sirens.
Source: https://en.wikipedia.org/wiki/Civil_defense_siren

Germany - Modular Warning System - MoWas

Utilises the government-owned satellite-based warning system (SatWaS). The warnings are sent to television, radio, internet and phone providers.
Source: https://www.kritis.bund.de/EN/Topics/Crisis_management/Warning/MoWaS/MoWaS_node.html

Japan - J-Alert

Warning alarm used to alert the public of danger. It aims to cover earthquake, tsunami, volcano, and military emergencies. The alarm is played over loudspeakers, television, radio and phone communication.
Source: https://www.centreforpublicimpact.org/case-study/disaster-technology-japan/

Lithuania - LT-Alert

Multilingual geographically targeted Public Warning System based on cell broadcast technology to complement the siren system.
Source: https://eena.org/wp-content/uploads/2018/11/Public-Warning-updated-version.pdf

Nederlands - NL-Alert

Based on Cell Broadcast technology, has been used mostly for fire related emergencies.
Source: https://crisis.nl/nl-alert

New Zealand - Emergency Mobile Alert

Using the Cell Broadcast Service as a means of delivering public warning messages to the general public.
Source: https://www.civildefence.govt.nz/get-ready/civil-defence-emergency-management-alerts-and-warnings/emergency-mobile-alert/

South America - Cadena Nacional

Joint broadcast, directed at the general population of a state. The message is played over television and radio.
Source: https://en.wikipedia.org/wiki/Cadena_nacional

Sri Lanka - Disaster and Emergency Warning Network - DEWN

Emergency message system in Sri Lanka that enables the disaster centre to use Sri Lanka's largest mobile service provider to send mass alerts as well as more specific warnings customized by region
Source: https://ieeexplore.ieee.org/document/6103681/

Sweden - Important Public Announcement

Warnings and information via traditional radio, radio data system, and television are complemented by the siren-system for outdoor warnings.
Source: https://www.msb.se/Upload/English/About_MSB_fact/Public%20warning%20systems%20in%20Sweden.pdf

Taiwan - Public Warning System

Using the Cell Broadcast Service as a means of delivering public warning messages to the general public.
Source: https://www.aspanet.org/ASPADocs/Annual%20Conference/2018/Papers/ChangSsuMing.pdf

United States - Emergency Alert System

Warning alarm used to alert the public of danger. The message is played in many states through air raid sirens. People living near certain nuclear facilities have special radios at their home. Some emergencies are also sent out via e-mail, cellphone text message, and highway signs.
Source: https://en.wikipedia.org/wiki/Emergency_Alert_System

2.2.2.2 Non-Traditional Warning Systems

These alarm systems may include the Traditional alarm systems, but the ones listed here are because they also use the Internet, like Social Media or even Mobile Phone applications.

Australia - Emergency AUS

The Emergency AUS is a public warning system used as a phone application to alert the users of danger in Australia.
Source: https://twitter.com/emergencyaus

Canada - Alberta - Emergency Public Warning System

The Emergency Public Warning System (EPWS) is a public warning system used to alert the public of danger in the Alberta or surrounding areas. The warning is disseminated over television, radio and phone communication, but also in Social Media and through an Android and iOS application.
Source: https://www.emergencyalert.alberta.ca/content/about/mobileapp.html

Canada - Saskatchewan - Emergency Alert System

The Emergency Alert System is a public warning system used as a phone application to alert the users of danger in Saskatchewan, Canada.
Source: https://www.saskatchewan.ca/residents/emergency/saskalert

Indonesia - MAGMA

The MAGMA is a public warning system used as a phone application to alert the users of danger in Indonesia.
Source: https://magma.vsi.esdm.go.id/

Israel - Home Front Command

The Home Front Command uses sirens as alert mechanism.
Source: https://www.oref.org.il/894-en/Pakar.aspx

Mauritius - Emergency Alert System

The Emergency Alert System is a public warning system used as a phone application to alert the users of danger in Mauritius.
Source: https://play.google.com/store/apps/details?id=org.govmu.emergencyalert

Pacific Disaster Center - Disaster Alert

The Disaster Alert is a public warning system used as a phone application to alert the users of danger in the Pacific Area.
Source: https://disasteralert.pdc.org/disasteralert/

Spain - My112

My112 can contact the Emergency Center 112, sending the current position to the operator and receive real-time alerts emergency near the user position.
Source: http://112.gencat.cat/en/que-fem/apps-per-dispositius-mobils/

2.2.3 Commercial Public Warning System

Next, we will analyze solutions that are being applied and in use by private organisations.

2.2.3.1 Cell Broadcast

Cell Broadcast (CB) is a technology that has a similar user experience to SMS, text messages are displayed on the screen of the mobile device. The following companies have a product which uses the *Cell Broadcasting* technology.

- One2Many - https://www.one2many.eu/en/cell-broadcast/
- Celltick - https://www.celltick.com/cell-broadcast-center/

2.2.3.2 Cell Aggregators

Aggregator means that is based on Cell Broadcast together with SMS alerting among other technologies. And the following companies have a product that uses these technologies.

- Celltick - https://www.celltick.com/products/mass-alert/
  - Cell Broadcast
  - SMS Alerting
- MASSALERT - http://www.ntservice.lt/go.php/Sprendimai_LIT89793396780956777
  - Cell Broadcast
  - SMS Alerting
  - Television DVB Alerting
  - IP web Alerting

2.2.3.3 Alert Clients

Like the *Non Traditional Public Warning Systems* in Section 2.2.2.2 but below there is the list of companies that use these systems.

- CODERed - https://play.google.com/store/apps/details?id=com.ecnetwork.crma

2.2.3.4 Incident reporting and management

Looking at a software perspective, there are also companies implementing solutions about incidents (that could be alerts in some way):

- 1st Incident Reporting - https://1stincidentreporting.com/en/
- Safety Culture - https://safetyculture.com/spotlight/
- Cyphon - https://www.cyphon.io/
- Worldsensing incidents - https://incidents.worldsensing.com/

2.3 Conclusions

After enumerating and analyzing the points indicated above, several conclusions can be drawn of the warning systems:

First of all is that due to the previous ages (World War II), almost all the countries have adopted a traditional alert system, based on sirens and/or radio.

Another remarkable aspect is that not all countries have the same idea about an alert system. Since in reality the need for a system of alerts is born from the very need to "monitor" the surrounding natural environment (by all means, we must also take into account that artificial hazards also exist), but the fact is that every country has some kind of self-tailored warning system.

It has also been observed that there are few countries that choose to have a system of alerts of the latest technology, that is based on mobile applications. Not only to indicate the impending hazard but also to report more information, like "What caused this hazard", "When will be no hazard", "Visualise maps within the affected zone", etc.

Then, among the uses of a Public Warning System in a commercial way, what you see is that there are companies which do a small part of every needed piece of a PWS, but none of them do it end-to-end and in a simple way of usage, which is the main idea of this project.

3 Methodology

3.1 Work methodology

Almost all of the projects that are inside *WorldSensing's Innovation* area follow the *Agile* methodologies. Specifically the *Scrum* methodology. So, this project is proposed to be done using this methodology.

3.1.1 Agile

Agile is a time-boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end. [20]

Agile Manifesto

Individuals and interactions	over	Process and Tools
Working Product	over	Comprehensive Documentation
Customer Collaboration	over	Contract Negotiation
Responding to change	over	Following a plan

That is, while there is value in the items on the right,
we value the items on the left more.

Table 2: The Agile Manifesto [18]

Overview of Agile [23]:

- **Iterative, incremental and evolutionary**

  Most agile development methods break product development work into small increments that minimise the amount of up-front planning and design. Iterations, or sprints, are short time frames (time-boxes) that typically last from one to four weeks. Each iteration involves a cross-functional team working in all functions: planning, analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is demonstrated to stakeholders.

- **Efficient and face-to-face communication**

  No matter which development method is followed, every team should include a customer representative. This person is agreed by stakeholders to act on their behalf and makes a personal commitment to being available for developers to answer questions throughout the iteration. At the end of each iteration, stakeholders and the customer representative review progress and re-evaluate priorities with a view to optimising the return on investment (ROI) and ensuring alignment with customer needs and company goals.

- **Very short feedback loop and adaptation cycle**

  A common characteristic in agile software development is the daily stand-up. In a brief session, team members report to each other what they did the previous day toward their team's iteration goal, what they intend to do today toward the goal, and any roadblocks or impediments they can see to the goal.

- **Quality focus**

  Specific tools and techniques, such as continuous integration, automated unit testing, pair programming, test-driven development, design patterns, behaviour-driven development, domain-driven design, code refactoring and other techniques are often used to improve quality and enhance product development agility.

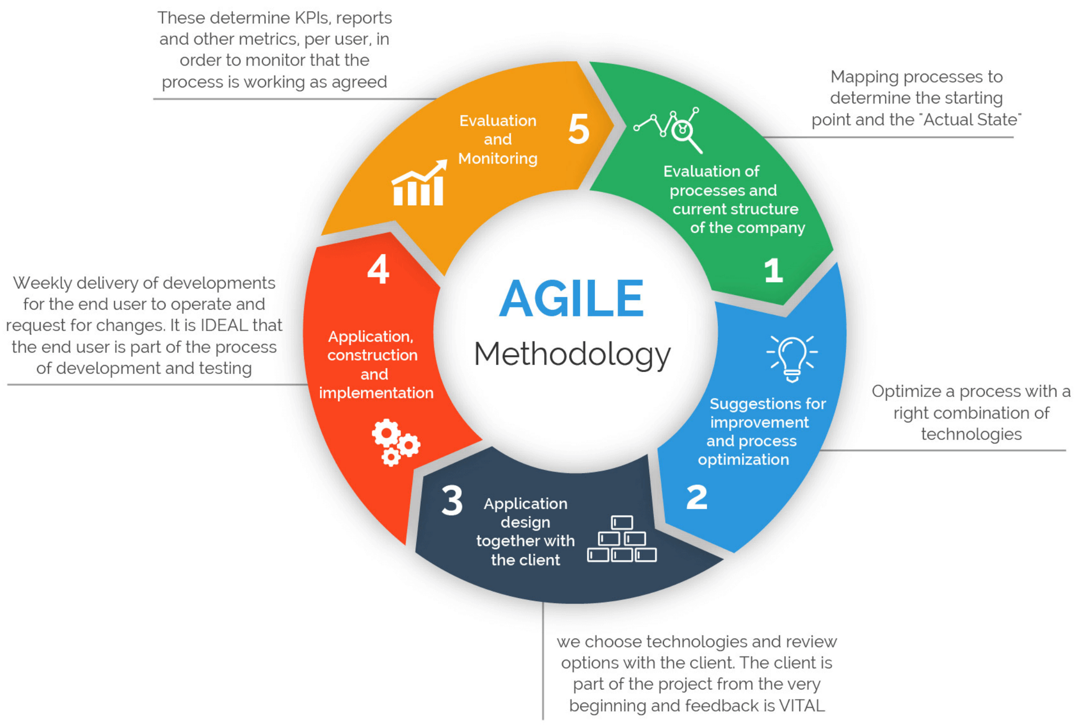

These determine KPIs, reports and other metrics, per user, in order to monitor that the process is working as agreed
Evaluation and Monitoring
5
Mapping processes to determine the starting point and the "Actual State"
Evaluation of processes and current structure of the company
1
AGILE
Methodology
Weekly delivery of developments for the end user to operate and request for changes. It is IDEAL that the end user is part of the process of development and testing
4
Application, construction and implementation
Suggestions for improvement and process optimization
2
Optimize a process with a right combination of technologies
3
Application design together with the client
we choose technologies and review options with the client. The client is part of the project from the very beginning and feedback is VITAL

Figure 12: Overview on *Agile*[16]

3.1.2 Scrum

Scrum is an Agile framework for managing work with an emphasis on software development. [4]

This framework is based in iterations, which are called sprints [17]. Tasks are defined for each sprint, which must be finished before the sprint ends. The most common duration of a sprint is two weeks.

Before starting each sprint there is a *Sprint Grooming* in which tasks are defined or taken from the *Backlog*[18], and later a *Sprint Planification* activity in which team members plan about what will be their work in the next sprint.

[16]GitScrum:
https://site.gitscrum.com/what-is-agile-methodology/
[17]Iteration:
Timebox during which development takes place.
[18]Backlog:
Ordered list of items representing everything that may be needed to deliver a specific outcome.

Finally, when the sprint comes to an end there is another activity which is called *Sprint Retrospective*, in which all the team members put in common their feelings or opinions on the work done in that sprint, which are the positive areas/ideas to maintain, which to improve/create, or which to stop doing.

Also, each day (typically at the beginning or end of the workday) there is a short ceremony called *Daily Stand-up*, in which every scrum-team member tells it's work for the last day and its next work.

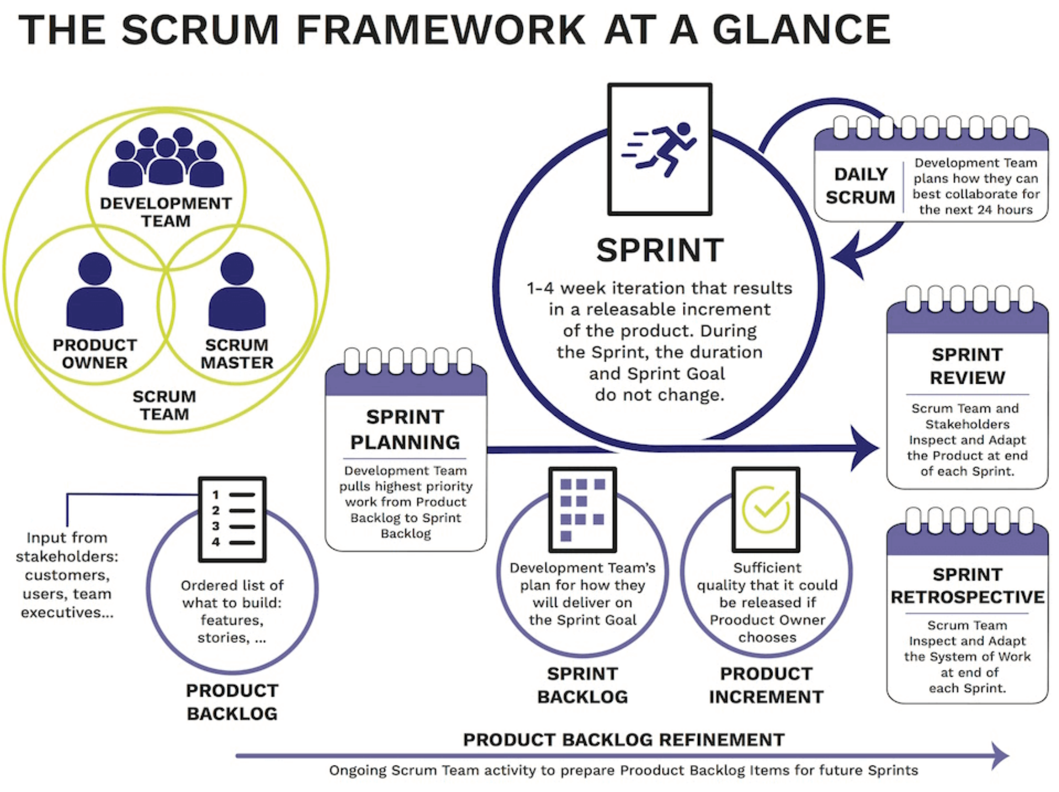

THE SCRUM FRAMEWORK AT A GLANCE
DEVELOPMENT TEAM
PRODUCT OWNER
SCRUM MASTER
SCRUM TEAM
SPRINT
1-4 week iteration that results in a releasable increment of the product. During the Sprint, the duration and Sprint Goal do not change.
DAILY SCRUM
Development Team plans how they can best collaborate for the next 24 hours
SPRINT REVIEW
Scrum Team and Stakeholders Inspect and Adapt the Product at end of each Sprint.
SPRINT PLANNING
Development Team pulls highest priority work from Product Backlog to Sprint Backlog
SPRINT RETROSPECTIVE
Scrum Team Inspect and Adapt the System of Work at end of each Sprint.
Input from stakeholders: customers, users, team executives...
Ordered list of what to build: features, stories, ...
PRODUCT BACKLOG
Development Team's plan for how they will deliver on the Sprint Goal
SPRINT BACKLOG
Sufficient quality that it could be released if Prooduct Owner chooses
PRODUCT INCREMENT
PRODUCT BACKLOG REFINEMENT
Ongoing Scrum Team activity to prepare Prooduct Backlog Items for future Sprints

Figure 13: Overview on *Scrum*.[19]

3.1.2.1 Actions and ceremonies

Sprint

During a Sprint (an iteration), a working product increment is developed. It has usually a duration of two weeks, and this duration remains constant for all the sprints in the project. A new Sprint starts immediately after the conclusion of the previous Sprint.

Sprint Grooming

Is when the product owner and some, or all, of the rest of the team review items on the backlog to ensure the backlog contains the appropriate items, that they are prioritised.

[19] AgileForAll:
https://agileforall.com/resources/introduction-to-agile/

Sprint Planning

Is an event that occurs at the beginning of a sprint where the team determines the product backlog items they will work on during that sprint.

Sprint Retrospective

When the sprint ends, there is an opportunity for the Scrum Team to evaluate itself and create a plan for improvements to be enacted during the next Sprint. *What went well in the Sprint*, *What could be improved*, *What will we commit to improve in the next Sprint*, ...

Sprint Review

Held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed.

Show and Tell

Same as *Sprint Review* but for all the Scrum Teams, as a summary of the sprint work. This ceremony is not in *Scrum*, but it is a way to keep all *Scrum Teams* up-to-date.

3.1.3 Project's Scrum

So, for this project this is our team, the ceremonies we perform and their approximate duration.

- Number of members: 1 Product Owner + 2 Developers.
- Duration of the sprint: 2 weeks.
- Daily Stand-up: 2 min per person.
- Sprint Grooming: 2h per sprint.
- Sprint Planning: 1h per sprint.
- Sprint Retrospective: 30m per sprint.

Also, we increase the number of ceremonies with the *Sprint Review*, a ceremony that lasts 1h and its objective is to show internally in the scrum team the work that has been done in that sprint. And also the *Show and Tell.* In this ceremony, all the scrum teams in Worldsensing (nowadays five teams) have 5 - 10 minutes to present their work during the sprint.

3.2 Tracking Tools

In order to accomplish the *Scrum* objectives, in Worldsensing we use the following concepts/solutions/software.

Google Suite

In *Worldsensing* we use the *Google Suite*[20] on a regular basis. So we have all, internally, documented and well placed in a master *Google Drive* folder, with sub-folders and the documents inside them.

TargetProcess

From *TargetProcess Inc.*[21] It is a visual project management software that aims to provide teams with a powerful Agile tool. Used for task control in *Scrum*. In this tool, the task management is carried out. The tasks that should be done by the end of the sprint, the definition of them... Also it is used to report *bugs*, etc.

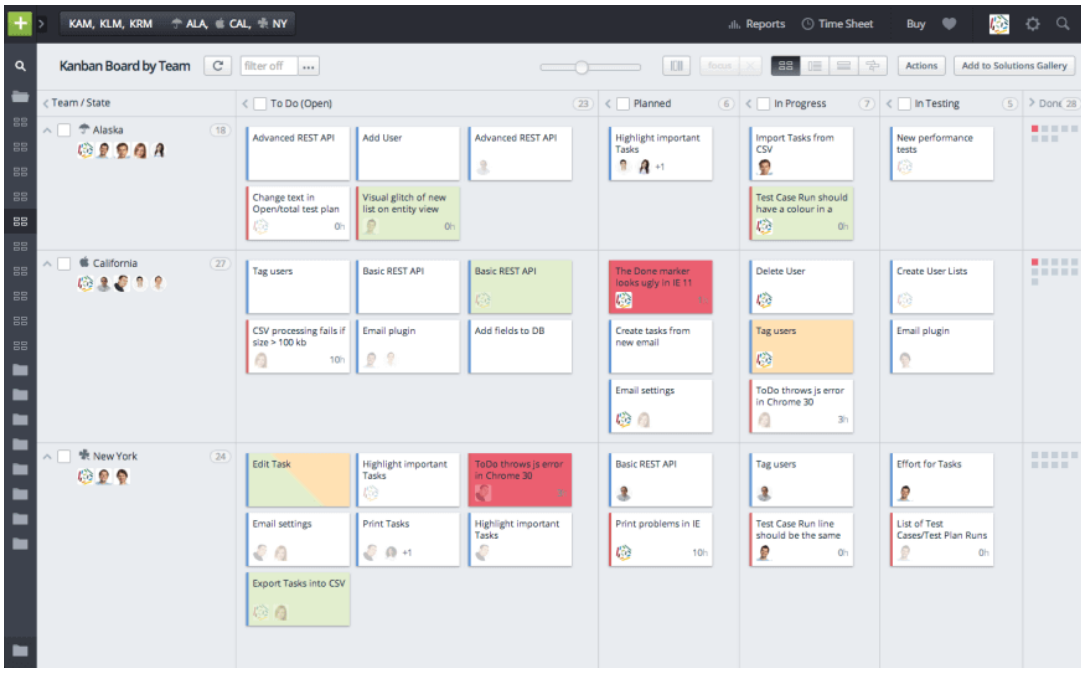


Figure 14: *TargetProcess* sample board.[22]

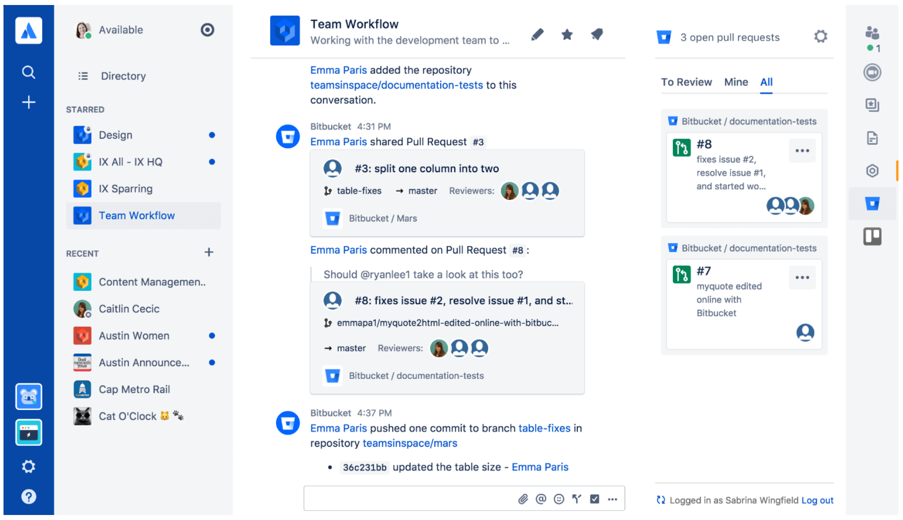
[20] Google Suite:
https://gsuite.google.com/

[21] TargetProcess:
https://www.targetprocess.com/

[22] TargetProcess board:
https://www.capterra.com/p/147228/Targetprocess/

Bitbucket

From *Atlassian*[23]. It is a web-based version control repository hosting service, for source code and development projects that use either Mercurial or Git revision control systems. Used to be able to develop all the code in parallel and also to have the historical revisions of it.



Emma Paris added the repository teamsinspace/documentation-tests to this conversation.

Bitbucket 4:31 PM
Emma Paris shared Pull Request #3

#3: split one column into two
table-fixes → master Reviewers:
Bitbucket / Mars

Emma Paris commented on Pull Request #8 :

Should @ryanlee1 take a look at this too?

#8: fixes issue #2, resolve issue #1, and st...
emmapa1/myquote2html-edited-online-with-bitbuc...
→ master Reviewers:
Bitbucket / documentation-tests

Bitbucket 4:37 PM
Emma Paris pushed one commit to branch table-fixes in repository teamsinspace/mars

- 36c231bb updated the table size - Emma Paris

Bitbucket / documentation-tests
#8
fixes issue #2, resolve issue #1, and started wo...

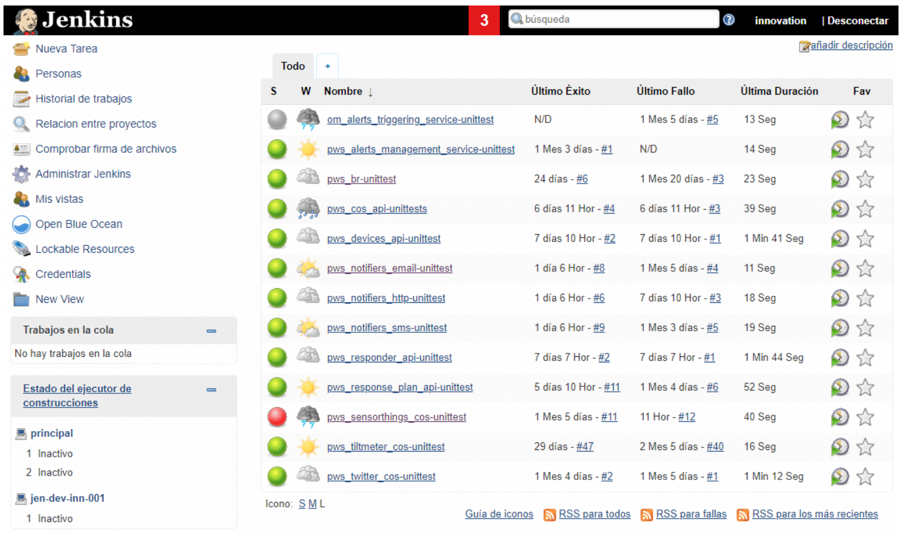
Bitbucket / documentation-tests
#7
myquote edited online with Bitbucket

Logged in as Sabrina Wingfield Log out

Figure 15: *Bitbucket* sample home screen.

Jenkins

Is an open source automation server written in Java. Jenkins[24] helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery. You can also configure it so that when a change is made in the version control, it automatically runs the tests for that repository.



Trabajos en la cola

No hay trabajos en la cola

Estado del ejecutor de construcciones

principal
1 Inactivo
2 Inactivo

jen-dev-inn-001
1 Inactivo

S	W	Nombre ↓	Último Éxito	Último Fallo	Última Duración	Fav
●		om_alerts_triggering_service-unittest	N/D	1 Mes 5 días - #5	13 Seg	☆
🟢		pws_alerts_management_service-unittest	1 Mes 3 días - #1	N/D	14 Seg	☆
🟢		pws_br-unittest	24 días - #6	1 Mes 20 días - #3	23 Seg	☆
🟢		pws_cos_api-unittests	6 días 11 Hor - #4	6 días 11 Hor - #3	39 Seg	☆
🟢		pws_devices_api-unittest	7 días 10 Hor - #2	7 días 10 Hor - #1	1 Min 41 Seg	☆
🟢		pws_notifiers_email-unittest	1 día 6 Hor - #8	1 Mes 5 días - #4	11 Seg	☆
🟢		pws_notifiers_http-unittest	1 día 6 Hor - #6	7 días 10 Hor - #3	18 Seg	☆
🟢		pws_notifiers_sms-unittest	1 día 6 Hor - #9	1 Mes 3 días - #5	19 Seg	☆
🟢		pws_responder_api-unittest	7 días 7 Hor - #2	7 días 7 Hor - #1	1 Min 44 Seg	☆
🟢		pws_response_plan_api-unittest	5 días 10 Hor - #11	1 Mes 4 días - #6	52 Seg	☆
🔴		pws_sensorthings_cos-unittest	1 Mes 5 días - #11	11 Hor - #12	40 Seg	☆
🟢		pws_tiltmeter_cos-unittest	29 días - #47	2 Mes 5 días - #40	16 Seg	☆
🟢		pws_twitter_cos-unittest	1 Mes 4 días - #2	1 Mes 5 días - #1	1 Min 12 Seg	☆

Icono: S M L

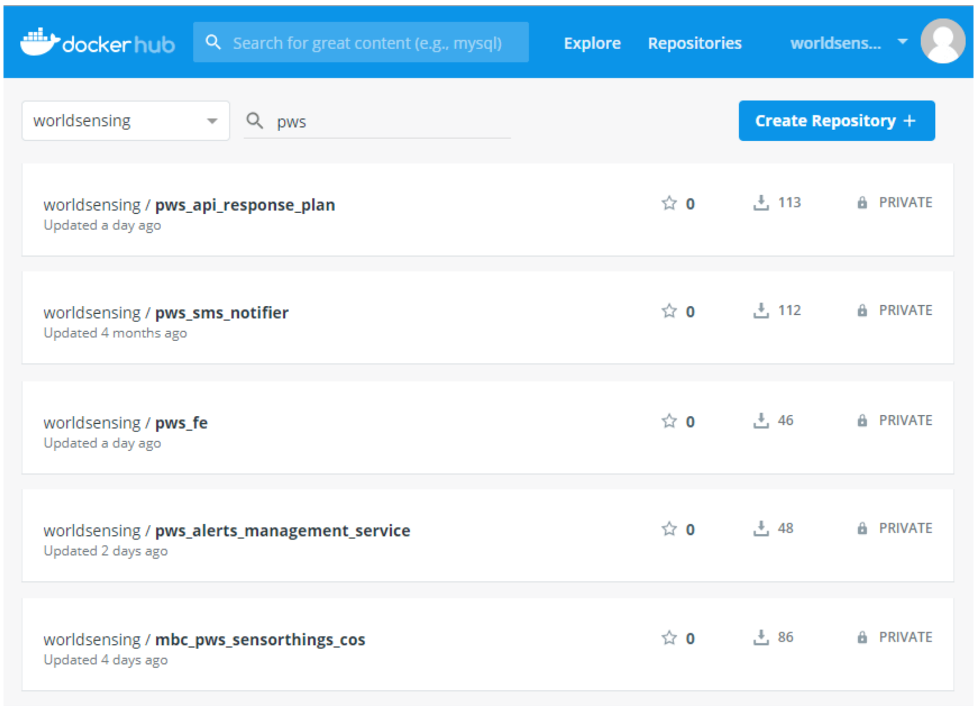
Guía de iconos RSS para todos RSS para fallas RSS para los más recientes

Figure 16: *Jenkins* for this project.

[23] Bitbucket:
https://bitbucket.org/product
[24] Jenkins:
https://jenkins.io/

DockerHub

Docker Hub is a service provided by Docker[25] for finding and sharing container images with a team. It provides repositories, tags, among other features.



worldsensing / pws_api_response_plan
Updated a day ago
0 113 PRIVATE

worldsensing / pws_sms_notifier
Updated 4 months ago
0 112 PRIVATE

worldsensing / pws_fe
Updated a day ago
0 46 PRIVATE

worldsensing / pws_alerts_management_service
Updated 2 days ago
0 48 PRIVATE

worldsensing / mbc_pws_sensorthings_cos
Updated 4 days ago
0 86 PRIVATE

Figure 17: Worldsensing's *DockerHub* with "pws" search.

3.3 Actors

3.3.1 Product Owner

José Gorchs as Product Owner in this project for *Innovation*. He will keep in mind the roadmap and calendar timing that the team should follow while developing and making sure the team delivers the desired outcome. He is also the advisor of this Master Thesis.

3.3.2 Project Manager

There will be Project Managers for the different European projects. They will be the stakeholders in their respective projects and will ensure that the solution obtained in the project is adapted to the respective needs.

[25] Docker:
https://www.docker.com/

3.3.3 Project Architect

As for defining the architecture of this project, *Marc Vila* (the author of this Master Thesis) will be the lead software architect. Making high-level design choices and dictating technical standards, including software coding standards, tools and platforms used in the project. But also having in mind the guidelines and coding standards of the other software team in *Worldsensing*.

In the annex can be found the Coding Guidelines, Section 10.5.

3.3.4 Developers

For the development of this project, *Marc Vila* will be the principal. Also with the definition and documentation of the project. He will also have the help of another developer, *Toni Martínez*.

With the development of both, the project will be carried out. In the next table, the development of every functional module in this project will be shown. The *X* means that this person led the development of the module. The *x* means that this person worked in the module. Also, it should be noted that below it is only shown the functional modules, but behind them there is also software to develop as software dependencies (databases, connections, among others).

Module	**Marc V**	**Toni M**
Sensor Connectors	x	X
Sensor Management	X	
Business Rules	X	
Alert Connectors	x	X
Alert Management	X	X
Alert Validation		X
Alert Forwarder		X
Actionables	X	x
Response Plan	X	
Frontend	X	
Deployment and integration	X	x

Table 3: Table with the modules of the project per person

3.3.5 UX and design

The project will be principally supported by *OneMind*, which its development relapse in another software team. But also another frontend will be developed fully by the developer team.

3.3.6 Testers

In every module of this project, there will be a test supporting it. These tests will be done by the developers. Also, the Product Owner and the stakeholders will review every Task or User Story that the developers deliver.

3.3.7 Users

The main users will be the administrators of the other project partners. Although, this software will also be tested internally in Worldsensing for potential productization.

3.4 Validation method

3.4.1 Internal

Each sprint, the product increase is reviewed by the Product Owner, the stakeholder. And pivoting the deployment if needed.

Also, all the *backend* code is tested with *unit-testing* and *integration tests.* It is also documented so a non-developer can try the code if needed. See Section 7.2.

It is in our internal coding guidelines that all code should be added to the repository by doing *pull requests.* In these, at least one developer (different than the one that pushes the code) should be reviewing the code. After the revision, the code is added to the *pre-production* site. When this development has been revised by the Product Owner, then it is deployed to the *production-ready* site.

3.4.2 External

The occlusion of the different European projects define a set of deadlines that have to be met. These are defined in Section 6.

4 Specification

In this section it will be explained which is the decided approach to implement this *warning system*.

The figure below illustrates the *Public Warning System* specification approach. In horizontal, it is represented the *state* of the information. Data can be added into our system by two ways. If the *data* comes to our system in the functional module of *Sensor Connectors*, it is called *Raw data*, each sensor has its own microservice. Once this data is processed in the *Business Rules*, it is feasible considered a *Tentative Alert*. After the *Alert Filtering* and the *Alert Validation* by an operator, it is called *Confirmed Alert*. And then an *Actionable* is executed, which has *Response Plans* to alert the relevant agents or organizations.

However, all this information will be explained in detail in the sections below.

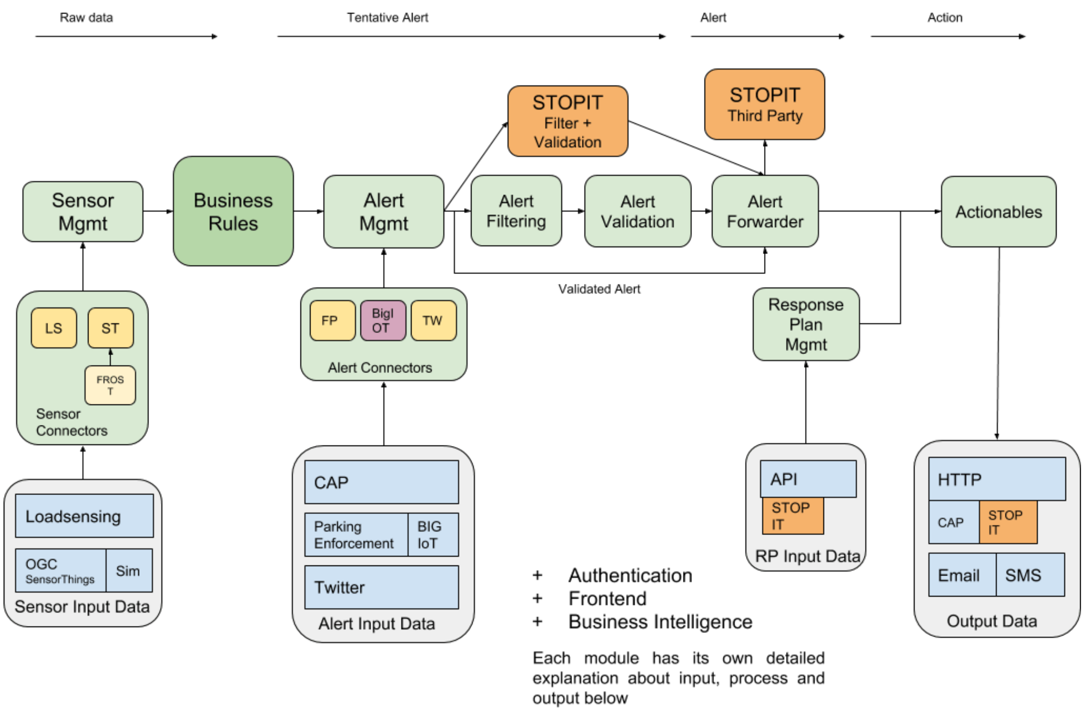


Figure 18: Overview of the *Public Warning System*.

All functional modules contain at least one microservice. The whole project will be a group of microservices, in order to remove the highest level of dependence between components. And to guarantee the interoperability of the components. This is, for example, the functional module that is called *Alert Validation* could be removed from the system if needed and the system could continue to work with no code changes (microservices, see Section 5.1).

The communication between the different modules is done in the shape of APIs. All functional modules are APIs. In which the way of communication is through an HTTP call. In addition, at places where bottlenecks can be generated due to the high number of calls per second, it is decided to implement queuing systems. In this way we always accept as many transactions as possible and at peak times are queued for further processing.

In the following sections, it will be explained each functional module with its inputs and outputs.

4.1 Sensor Connectors

The first step provided by a Public Warning System is the ability to collect data from different sources.

The system has multiple sensor inputs, called *Sensor Connectors* in Figure 18. This is because our system works with more than one kind of input from sensors. Thus, our system will have as many input APIs as types of devices supported.

4.1.1 Sensor Input Data

Two types of sensors have been considered in this *Master Thesis*. In the next section will be explained in detail each of them:

- Particular implementation from the *Worldsensing*'s commercial IoT node, which is the *Loadsensing Tiltmeter*, in the configuration of a two axis tiltmeter.
- Generic implementation from *OGC SensorThings* to support more inputs from sensors.

4.1.1.1 Loadsensing Tiltmeter

Loadsensing has been incorporated in our base of *Input Data from Sensors*. As it is currently the most important sensor in *Worldsensing*. A connector to our system has been developed. The sensor will send to the *LoRa* gateway the information, and this gateway will do an *HTTP POST Request* to our system, containing the sensor's data in *JSON* format.

Here is the schema of a *Loadsensing Tiltmeter*. Although the only values that will be currently read / saved are the *"nodeId:type"*, *"readings:axisOne"*, *"readings:axisTwo"*, *"readings:temperature"* and *"readings:timestamp"*.

```
{
  "nodeModel": {
    "type": "string"
  },
  "commMetaData": {
    "type": "dict",
    "schema": {
      "networkId": {"type": "string"},
      "macAddress": {"type": "integer"},
      "receivedTimestamp": {"type": "string"},
      "frequencyHertz": {"type": "float"},
      "snr": {"type": "integer"},
      "sequenceCounter": {"type": "list"},
      "gatewayId": {"type": "integer"},
      "rssi": {"type": "integer"},
      "type": {"type": "string"},
      "sf": {"type": "integer"},
      "macType": {"type": "string"},
    },
  },
  "nodeId": {
    "type": "integer",
    "required": true
  },
  "readings": {
    "type": "list",
    "Items": [
      {
        "type": "dict",
        "schema": {
          "axisOne": {"type": "integer", "required": true},
          "axisTwo": {"type": "integer", "required": true},
          "temperature": {"type": "float", "required": true}
        }
      }
    ]
  },
  "readTimestamp": {"type": "string", "required": true},
  "type": {"type": "string"}
}
```

Here is the full example of a *Loadsensing Tiltmeter* that will be sent to our system.

```
{
  "commMetaData": {
    "frequencyHertz": 868.85,
    "gatewayId": 12273815315514655000,
    "macAddress": 76553340,
    "macType": "ETSIV1",
    "networkId": "13797",
    "receivedTimestamp": "2019-01-14T09:35:05Z",
    "rssi": -73,
    "sequenceCounter": [
       1
    ],
    "sf": 9,
    "snr": 11,
    "type": "longRangeRadioMetaDataV1"
  },
  "nodeId": 7292,
  "nodeModel": "LS-G6-INC15",
  "readTimestamp": "2019-01-14T09:35:04Z",
  "readings": [
    {
      "axisOne": 49,
      "axisTwo": -103347,
      "temperature": 27.042537942785813
    }
  ],
  "type": "tiltReadingsV1"
}
```

4.1.1.2 OGC SensorThings API

A connector for *SensorThings* that injects the measurements into our database has been developed. This connector comes with a *FROST* server [26], which is a *SensorThings* server implementation by the *Fraunhofer Institute*[27]. This is needed because of the *OGC SensorThings*, which is a complex standard that has a lot of fields to fill, as seen in Figure 9. We could do another development but *FROST* is open source and has all the features that are needed. Thus, *FROST* was integrated in our system.

This *FROST* server accepts requests using the *OGC's SensorThings protocol*, which is a standard for IoT, as it was explained before.

Our connector queries periodically the *FROST* server for new data and sends it to *COS* microservice. This period is configurable and by default it is set on 5 seconds. Supported data types are numbers (integers, doubles, ...) and lists of numbers. More complex data can't be handled at the moment.

An example of a request can be found in the Annex 10.1.

4.1.1.2.1 Simulator

To validate the *OGC SensorThings Connector* we have developed a *SensorThings* simulator, which generates simulated data for a *Loadsensing Tiltmeter* sensor. The output of this simulator is a fully compliant *OGC SensorThings* data which is sent to the *PWS Sensor API*.

An example of a request generated by this simulator can be found in the Annex 10.2.

4.2 Sensor Management

When data arrives to our system connectors, it is still not standardized. That means, every input data sensor is yet inside our system but with different fields between them. Maintaining a system with unlimited kinds of data is not easy.

The data that is being ingested from the different sensors is not properly structured for the component to be analysed. This stage adapts the data to treatable formats for each of the different sensor which can be connected as a data source, as it has to structure the specific data variables to the format that will be used for all the rest of the processing.

So, this functional module reads the inputs from our system and adapts its data, so we can then treat it, which implies that we work at the data level. We can have *N* instances of sensors, but this *N* instances will follow the same structure. There is a list below explaining the structure per each sensor type.

[26]FROST explanation:
https://github.com/FraunhoferIOSB/FROST-Server/

[27]Fraunhofer Society is a German research organization with 72 institutes spread throughout Germany, each focusing on different fields of applied science. https://www.fraunhofer.de/en.html

4.2.1 Sensor COS

Worldsensing's *Custom Object Service*(COS) is an object with the necessary fields to position it in the world map showing its instances data. This nomenclature comes from the heritage of a part of *Mobility*, which is the frontend of the project.

A specific connector for every sensor type has been developed (so far: Loadsensing Tiltmeter and OGC). This connector reads data from their respective sensors and post it to COS following a standard format. COS is an API that works for us as an abstraction layer. After data has been posted to this service, we can guarantee that it will follow a format the rest of the PWS will be able to handle.

4.2.1.1 Loadsensing Tiltmeter

The following *JSON* shows the schema from the *Loadsensing Tiltmeter* that our system will read to proceed.

```json
{
  "type": "object",
  "properties": {
    "loadsensing_tiltmeter_id": {
      "$ref": "#/external_id"
    },
    "axis_one": {
      "$ref": "#/timeseries"
    },
    "axis_two": {
      "$ref": "#/timeseries"
    },
    "temperature": {
      "$ref": "#/timeseries"
    },
    "date": {
      "$ref": "#/date"
    },
    "coordinates": {
      "$ref": "#/geometry"
    }
  }
}
```

And following, the field definition of each value of the schema:

```json
{
  "loadsensing_tiltmeter_id": {
    "value": {"type": "string"}
  },
  "axis_one": {
    "value": {"type": "float"},
    "time": {"type": "string"}
  },
  "axis_two": {
    "value": {"type": "float"},
    "time": {"type": "string"}
  },
  "temperature": {
    "value": {"type": "float"},
    "time": {"type": "string"}
  },
  "date": {
    "value": {"type": "string"}
  },
  "coordinates": {
    "type": {"type": "string"},
    "coordinates": {"type": "list float"}
  }
}
```

4.2.1.2 OGC SensorThings

The following *JSON* shows the schema from the *OGC SensorThings* that our system will read to proceed with the flow of it.

```
{
    "type": "object",
    "properties": {
        "sensorthing_id": {
            "$ref": "#/external_id"
        },
        "{{ N items }}": {
            "$ref": "#/timeseries"
        },
        "date": {
            "$ref": "#/date"
        }
    }
}
```

And following, the field definition of each value of the schema:

```
{
    "type": "object",
    "properties": {
        "sensorthing_id": {
            "value": {"type": "string"}
        },
        "{{ N items }}": {
            "value": {"type": "float"},
            "time": {"type": "string"}
        },
        "date": {
            "value": {"type": "string"}
            }
        }
}
```

4.2.2 Sensor API

This part of this functional module is the link between the *Sensors Connectors* functional module and the *Business Rules* functional module, with an external endpoint to configure it.

The idea of this microservice is not just to let the user to see the information of his sensors in the system, but also to let the user modify the settings of these sensors. In this stage, information such as which ID has the sensor, where it is located among other information.

4.3 Core - Business Rules

After the ingestion of data streams, and its pre-processing, the main objective of the Public Warning System is to detect possible anomalies from the data streams being monitored.

It is, the system reads raw data from a sensor and when it arrives to this component, this data needs to be analysed to know if it is just normal data (discard), or maybe it is data that contains dangerous values (tentative alert).

This component is the core of the architecture. It contains a timeseries database (*InfluxDB*, 5.3) and a triggering service (*Kapacitor*, 5.3). The *Sensor Management* gets data from the sensors and inserts them into the timeseries database (*InfluxDB*). Then this component, the triggering service (*Kapacitor*), reads this database and the rules are evaluated to check if the sensor's data fulfil the requirements to generate an alert, an anomaly. A schematic image is shown below, where *WorldSensing COS Adapter* refers to the *Sensor Management*.

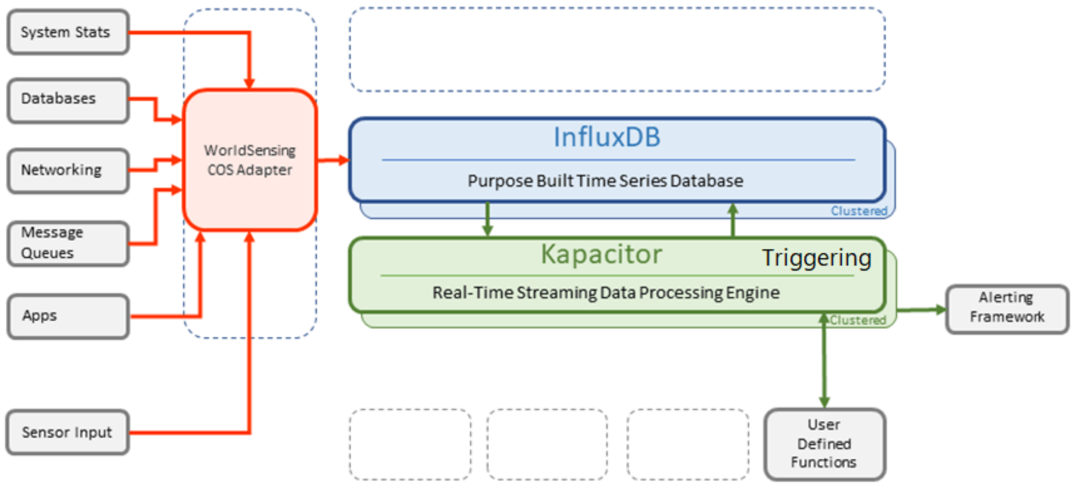


Figure 19: Triggering Service in the PWS.

4.3.1 Triggering Service

The software used as Triggering Service in this project is called *Kapacitor*, an *Open Source* Real-time Streaming Data Processing Engine, and will be explained in Section 5.3.

This component is continuously monitoring (Continuous Queries) the data stream and triggering actions when the conditions are met. Continuous Queries means that the engine is constantly querying a database or service in order to have a result or action to execute. This triggering service queries every predefined time the data stream of a timeseries database and with the result, generates a defined action: it could send an HTTP request, add information to another database or even *User Defined Functions*.

For now, it is only supported the data structure of a “two axis sensor”, for example the Loadsensing Tiltmeter. But in the future more sensors will be supported in the triggering system. In the image below it is shown how the triggering service works.

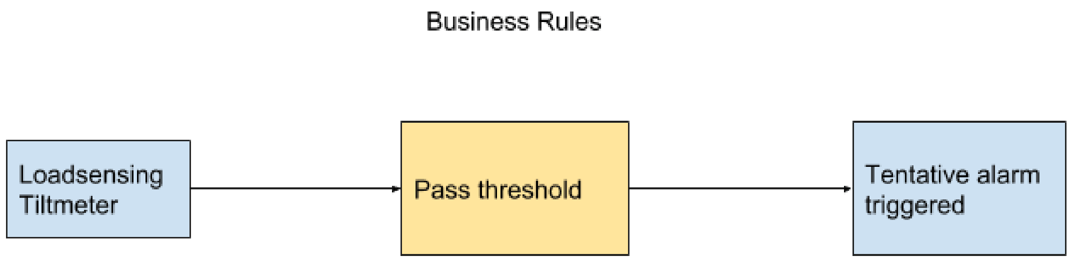


Figure 20: Business Rules in the PWS.

4.3.1.1 Loadsensing Tiltmeter

This triggering service reads every 2 seconds the database with the latest 10 seconds of measurements. This is, in a time *X+2* the service reads the values from *X-8* to *X+2*. For each sensor, when its *axis_one* or *axis_two* values exceeds the threshold, an HTTP and a QUEUE message are sent.

This actions are represented at the *TICK Script* file, the *SQL-like* language that is used by Kapacitor.

```
SELECT object_type, axis_one, axis_two, loadsensing_tiltmeter_id
   FROM "db"."measurements"
   GROUP BY "loadsensing_tiltmeter_id"
   WHERE "object_type == 'loadsensing_tiltmeter'"
   WHERE "axis_one > threshold OR axis_two > threshold"
TRIGGER HTTPOUT as 'alert'
TRIGGER QUEUEOUT as 'alert'
EVERY 2s
PERIOD 10s
```

Finally, when an alert is triggered the information is submitted to a queue (RabbitMQ, 5.3), where the *Alert Enricher* module (4.5.1) includes metainformation for the *Alert* (such as location coordinates which are available at the sensor description).

The following is an example of a *Tentative Alert* generated by our *Business Rules Engine* with the *Loadsensing Tiltmeter* as a sensor:

```
{
    "action_stamps": {
        "created_by": "BRE"
    },
    "alert": {
        "absolute_value": -18.14668,
        "description": "Axis axis_one of tiltmeter with id 7292 exceeded
        ↪  19.14668 degree threshold during more than 4 seconds",
        "forecast_value": 0.0,
        "score": 366.5953550224,
        "severity": 1
    },
    "related_item": {
        "item_id": "7292",
        "item_type": "tiltmeter",
        "measure_name": "tilt",
        "measure_unit": "degrees"
    },
    "rule": {
        "rule_id": "RULE#TILTMETER#1",
        "rule_name": "tiltmeter rule",
        "threshold": 1.0
    },
    "timestamps": {
        "created_at": 1540907610000.0
    }
}
```

4.4 Alert Connectors

The PWS is able to receive external alerts from structured sources. Once the data reaches our system and is read in the *Alert Connector* functional module, these data have to be homogenised so that it can be treated by our system.

This functional module will have two-way inputs. The first one is the normal flow of our system, that means when the alert is generated in our Business Rules Core. And the second way of input is that the system has an input API to receive pre-generated alerts. These alerts will come from another third party reporting organizations. Thus we can treat in our system our alerts but also alerts generated by other certified organization.

4.4.1 Alert Input Data

In the frame of this *Master Thesis three* types of inputs were defined, which will be specified in the following section:

- Particular implementation from *Worldsensing*'s corporate IoT device, *Fastpark Enforcement*.

- Particular implementation from *Worldsensing*'s corporate IoT device, *Fastpark Enforcement* through the *BIG-Iot* platform.
- Information from *Twitter*.

4.4.1.1 Fastpark - Enforcement

Fastpark is the *Worldsensing*'s device explained in the Section 1.2.1.1.1. And the *Enforcement* part is a *software* improvement with the objective to identify and report which *Fastpark* sensors don't have an active payment, so, potential unpayments in the parking occupation.

This connector will have internal use in the project since it serves as an added feature for *Fastpark*. But it will also be submitted for the <IMPACT>.project. In the same way, the data messages which are accessed through the BIG-IoT platform will be used for this project.

Then, the following is the definition of the information received in the *Alert Connectors* functional module.

4.4.1.1.1 Message Ingestion

Full message retrieved from the *Fastpark Connector*. It shows all the available information of a *Fastpark* sensor.

```
[
   {
      "id":"5a2fb6401170857bd41f0cd4",
      "customId":"Regular",
      "name":"Regular",
      "displayedName":"Regular",
      "occupation":{
         "type":"sectorOccupation",
         "numberOfFreeSpots":11,
         "numberOfOccupiedSpots":0,
         "totalSpots":11,
         "lastOccupationChange":1547747749000,
         "occupationStatus":"free",
         "averageStayLength":0.0,
         "occupations":[
            {
               "type":"delimitedOccupation",
               "name":"default",
               "freeSpaces":11,
               "totalSpaces":11
            }
         ]
      },
      "sectorType":"5b83d2c61170b9d04182617b",
      "active":true,
      "position":{
         "type":"sectorPosition",
         "positioned":true,
         "lon":2.113889927013216,
            "lat":41.38899333731589,
            "points":[
               [
                  {
                     "lon":2.113726312263294,
                     "lat":41.388886685660665
                  },
                  {
                     "lon":2.114053541763137,
                     "lat":41.38909998897111
                  }
               ]
            ]
         }
      }
]
```

4.4.1.1.2 Message Output to system

An example of a well-structured alert that would be the output towards the PWS as a *Fastpark Enforcement* alert would look like this example:

```
{
   "meta_type":"meta_data",
   "related_item_id":"2835",
   "related_item_type":"parking_enforcement",
   "rule_id":"PARKING#ENFORCEMENT#2",
   "status":10,
   "sub_type":"n/a",
   "the_geom":{
      "coordinates":[
         2.134899,
         41.387236
      ],
      "type":"Point"
   },
   "title":"Ilegal parking occupation in sector Numancia / Illa Diagonal",
   "type":"Enforcement with trackers"
}
```

Rule Example

And this is the rule that was generated by the alert also in the *Fastpark Enforcement* connector module:

```
{
   "id":"PARKING#ENFORCEMENT#2",
   "measure_name":"infractions",
   "measure_unit":"number of infractions",
   "name":"parking enforcement with trackers rule",
   "reference_type": "threshold"
}
```

4.4.1.2 Social Media - Twitter

An important part of the PWS are the alerts raised by social media. The PWS will be gathering data from social media and also analysing it to get the relevant information. A *Twitter* connector that tracks the messages following a certain criteria has been developed.

In this first version, the criteria is for a defined string which is found in the message's body. This could be easily extended to searching for multiple strings, selecting tweets containing only a given hashtag or mentioning a specific user. More advanced filters could include filtering based on the user's reputation (minimum number of followers, having a profile picture, ...) to prevent bots from spamming the system.

In this connector, a list of strings can be set in order to change the *Twitter* query. This connector connects to *Twitter* using the *Python* library *Twython*[28].

The results containing a geo-tag will be sent to the system as a tentative alert to be shown on the map. Additionally, the number of messages received over time will also be inserted into a timeseries database using *Worldsensing*'s Custom Object Service (COS). In the future we will also be able to generate alerts based on that (e.g. number of messages containing certain strings received over a time period exceeding a given threshold).

4.4.1.2.1 Message Ingestion

Full message retrieved from the *Twitter Connector* when searching for tweets with the keywords "water leak" can be found in the Annex 10.3.

4.4.1.2.2 Message Output to system

From the full message, only the following parameters will be ingested in the PWS by the *Twitter* connector:

```
{
  "tweet_id": {"type": "string"},
  "username": {"type": "string"},
  "created_at": {"type": "string"},
  "coordinates": {"type": "string"}
}
```

Alert Example

An example of a well-structured alert that would be the output towards the PWS as a *Twitter* alert would look like this example:

```
{
   "meta_type":"meta_data",
   "related_item_id":"tmartinez worldsensing",
   "related_item_type":"twitter user",
   "rule_id":"water leak",
```

[28]Twython: https://twython.readthedocs.io/en/latest/

```
    "status":10,
    "sub_type":"n/a",
    "the_geom":{
       "coordinates":[
          2.1412879,
          41.380824
       ],
       "type":"Point"
    },
    "title":"Alert reported by tmartinez worldsensing via Twitter refering to
    ↪  'pws_ws_prueba'",
    "type":"Twitter"
}
```

Rule Example

And this is the rule that was generated by the alert in the *Business Rules* module:

```
{
    "id":"pws_ws_prueba",
    "measure_name":"Incidence",
    "measure_unit":"Incidence geotag",
    "name":"Tweet matching rule"
}
```

4.5 Alert Management

This module receives alerts being posted to it, either by the *Business Rules* module or by an external alerts generator such as *Fastprk Enforcement* service or the *Twitter* connector. When tentative alerts are received, they are inserted in a database and forwarded to be filtered/validated. If the tentative alert is received with the flag *validated* in the *status field*, it is forwarded directly to the *Alert Forwarder* module.

4.5.1 Alert enricher

When the tentative alert is received from the *Business Rules* module, it follows the *Alert* format. But some fields of this alert can be incomplete. The goal of this part of the module is to have a field enricher, filling the empty fields with information from the *Sensor Management* module.

For example, the *Loadsensing Tiltmeter* data that flows from the *Business Rules* module does not have geolocalization. So this module gets this information from the *Sensor Management* module.

4.5.2 Alerts core

This microservice is in charge of maintaining all the management of the alert system. From the definition of the *Alert* model, its filtering, validation and subsequent forwarding to the external modules or internal response plan module.

4.5.2.1 Rule definition

Below is the definition of the *Rule* in our system. The fields to be patched will be exchanged in JSON format following the structure explained below:

Attribute	**Type**	**Values**	**Description**
id	Integer	-	Autogenerated
name	String	-	Brief rule description
measure_name	String	-	Name of the measure
measure_unit	String	-	Unit of the measure

Table 4: Model of a Rule in Alerts Management's Core

4.5.2.2 Alert definition

Below is the definition of the *Alert* in our system. The fields to be patched will be exchanged in JSON format following the structure explained below:

Attribute	**Type**	**Values**	**Description**
id	Integer	-	Autogenerated
rule_id	String	-	Mandatory ID of the corresponding rule that generated the alert
title	String	-	Mandatory Brief alert description
description	String	-	Long alert description
recurrence	Integer	-	Number of times the same alert has been received. Alerts equality is based on same “rule_id”, “related_item_id” and “related_item_type”
resolved_at	Timestamp	-	Epoch when the alert operative_status was set to resolved
related_item_type	String	e.g. Tiltmeter	Alert sensor source type
related_item_id	Integer	e.g. 4959	Alert sensor source ID
address	String	e.g “crossing between Numància and Viriat Street”	Physical location of the alert in natural language
the_geom	List Float	-	Where latitude and longitude correspond to the location where the alert took place.
type	Integer	10 (Geo), 20 (Met), 30 (Safety), 40 (Security), 50 (Rescue), 60 (Fire), 70 (Health), 80 (Env), 90 (Transport), 100 (Infra), 110 (Social Media), 200 (CBRNE), 300 (Other)	Mandatory Category of the alert. This field is used to match alerts and response plans.
score_value	Float	-	Magnitude of the alert.
absolute_value	Float	-	x if axis 0 exceeded threshold, y if axis 1 exceeded threshold
absolute_difference	Float	-	x - x0 if axis 0 exceeded threshold, y - y0 if axis 1 exceeded threshold
status	Integer	0 (no action taken), 1 (forward to external module, e.g STOP-IT), 2 (AUTOMATIC response plan), 4 (MANUAL response plan)	
operative_status	Integer	0 (new), 10 (seen), 20 (attended), 25 (confirmed), 30 (resolved)	Status given by the operators handling.

Table 5: Model of a Alert in Alerts Management’s Core

4.5.3 Output Data

Here the alert will be sent as *tentative alert* to the STOP-IT external module, as defined in the Section 6.1.2.2. This alert will be sent following our internal schema.

4.6 Alert Filtering

Now we have the *Tentative Alerts* in the system. These *Alerts* may have been inserted by our *Business Rules* core, but also from external sources. So, here these *Alerts* will receive a filtering to check whether if they are correct and reliable.

Different implementation for STOP-IT, check Section 6.1.

4.7 Alert Validation

After the filtering, if the alert arrives to the *Validation* functional module, it means that for our system the alert is programatically validated and well formed. And the next step is for the operator to validate it.

One way of doing it is validating visually in our frontend. From the backend side, validating an alert means sending a PATCH request to the *Alerts API* with an updated status and the alert ID. Alert statuses are documented in Section 6.1.2.2.

Different implementation for STOP-IT, check Section 6.1.

4.8 Alert Forwarder

This module receives alerts that have been validated and forwards them to either another external module or to the *Actionables* module.

These alerts can be received via *OneMind* frontend by the operator, or via external module such as *STOP-IT*'s, but also other *Third Parties*, as long as their message matches our data format.

This module is part of the *Alerts Management API*. Every time there is a change in an alert's status it checks if the previous status corresponded to a *tentative alert* and the current status to a *confirmed alert*. In this case it will execute two actions.

The first action to execute will be to forward the alert in the PWS's internal alerts format to the external module that has been previously configured to be the “alerts external module”. The external module must be an API accepting POST requests.

The alerts status change that will trigger the forwarding can come either from an operator validating the alert in OneMind's frontend, or from another “Filter and Validation” external module.

While the consortium's partners don't have the “alerts external module” ready, the “alert forwarding” functionality will be demonstrated thanks to a module that has been developed to emulate it. This module will receive the forwarded alerts, and simply show in the system logs the alerts information.

The second action to be executed by the Alert Forwarder will be to tell the *Actionables* module to launch all the automatic response plans corresponding to this alert's category/type. To do so, it will query the *Response Plans API* to obtain a list of response plans matching the alert's category/type. Afterwards, it will filter the retrieved response plans to obtain those whose execution should be automatic (and not manual by an operator). Finally, both the alert and the response plan will be posted to the *Actionables* module, which in turn will connect with the appropriate module(s).

4.8.1 Input Validated Alerts

Our system will know that an alert is validated when receiving this alert with its ID and also the *alert_status* as validated. The expected input is the reported status of the tentative incident after being shown to the visualisation interface and being managed by an operator.

The input to the PWS received from the visualisation module has the following parameters:

Attribute	**Type**	**Values**	**Description**
alert_id	Integer	-	Mandatory. The alert ID sent to validate.
alert_status	Integer	0 (Open), 10 (Seen), 20 (Attended), 25 (Confirmed), 30 (Resolved)	Mandatory. The status of the alert.

Table 6: Input of Validated Alerts

4.8.2 Forward Alerts

The same *Alert* that was received is sent to its destination. For project internal use, the alert is sent to the *Actionables/Response Plan* functional module. Also, an emulator of an "external alerts module" was implemented to demonstrate the functionalities.

Example of a forwarded alert. The alert source was *Twitter*, where a user "tmartinez worldsensing" sent a tweet:

```
{
   "id":37,
   "title":"Alert reported by tmartinez worldsensing via Twitter referring
   ↪  to 'pws_ws_prueba'",
   "description":null,
   "recurrence":1,
   "resolved_at":null,
   "assigned_to":null,
   "created_by":null,
   "updated_by":null,
   "resolved_by":null,
   "related_item_id":"tmartinez worldsensing",
   "related_item_type":"twitter user",
   "address":null,
   "the_geom":{
        "type":"Point",
        "coordinates":[
           2.1413403,
           41.3808141
        ]
     },
     "meta_type":"meta_data",
     "type":"Twitter",
     "sub_type":"n/a",
     "absolute_value":null,
     "absolute_difference":null,
     "score_value":null,
     "severity":null,
     "status":10,
     "operative_status":10,
     "rule_id":"pws_ws_prueba",
     "updated_at":"2019-01-17 10:42:15",
     "created_at":"2019-01-17 10:41:50",
     "_etag":"6939cf4109008b0a38def5a9d242b0f30d0d77fe"
}
```

4.8.2.1 Forward Alerts Receiver - DEMO

A service that simulates an external module receiving the forwarded alerts has been developed. It is not inside the production system, just for demo purposes and can be instantiated separately.

4.9 Response Plan

The component allows the management of response plans, which are a structure of actions to perform when an alarm situation is presented. The response plans should be informed before the alarm situation occurs, as the actions that should be taken would be always the same ones as initially defined for a specific alarm situation. The service allows the creation, update, query and deletion of such response plans through HTTP requests. An example can be found in the Annex 10.4.3.

This functional module contains the information regarding the actions to execute given an alert type. When the system receives a *Validated Alert* the system checks and gather its associated *Response Plan* in order to know which actions has to do.

4.9.1 External Response Plans

This module lets our system have an input for defining the Response Plan actions to do whenever an alert gets activated (or validated).

The file expected to be received by the PWS is a JSON structured file, and will be submitted to an API for this purpose.

Definition of a Response Plan

Attribute	**Type**	**Values**	**Description**
response_plan_id	String	-	Mandatory. Unique. Name to identify the Response Plan.
message_status	Integer	10 (Real), 20 (Test)	Mandatory. Code to define the nature of the message being sent.
category	Integer	10 (Geo), 20 (Met), 30 (Safety), 40 (Security), 50 (Rescue), 60 (Fire), 70 (Health), 80 (Env), 90 (Transport), 100 (Infra), 110 (Social media), 200 (CBRNE), 300 (Other)	Mandatory. Category of the subject event of the response plan message.
actions	List Integer	10 (Email), 20 (SMS), 30 (HTTP_POST)	Mandatory. Actions to do of the response plan message.
parameters	List String	-	Mandatory. Recipients of the action sent.
format	List Integer	10 (Plain_Text), 20 (CAP), 30 (Other)	Mandatory. Format of the action sent.
description	List String	-	Mandatory. Description of the action.
automatic	List Integer	10 (Automatic), 20 (Manual)	Default is 10. Trigger of the response plan message.
severity	Integer	10 (Extreme), 20 (Severe), 30 (Moderate), 40 (Normal), 50 (Minor), 100 (Unknown)	Default is 40. Severity of the subject event of the alert message.
accessibility	Integer	10 (Operator), 20 (Administrator)	Default is 20. Restriction to modify the response plan.
area	String	-	Description of the area affected by the response plan message.
geolocation	Str (WKB)	-	Geometric representation of the area.

Table 7: Model of a Response Plan

Get a Response Plan

These messages will be received through the API as an HTTP GET. And it will return a Response Plan object as JSON.

Method	**GET**
Endpoint	/response-plans/{name-ID}/
Output	
200 *404*	*OK*, Response Plan returned *NOTFOUND*, Response Plan not found

Table 8: GET method for Response Plans

Insert a Response Plan

These messages will be received through the API as an HTTP POST. The Enumerators should be done with *StatusCode* integers instead of Strings in order to maintain a more consistent way of communication.

Method	**POST**
Endpoint	/response-plans/
Body	
application/json	Response Plan instance
Output	
201	*CREATED*

Table 9: POST method for Response Plans

Update a Response Plan

These messages will be received through the API as an HTTP UPDATE.

Method	**PUT**
Endpoint	/response-plans/{name-ID}/
Body	
application/json	Response Plan instance
Output	
200 *404*	*OK*, Response Plan returned *NOTFOUND*, Response Plan not found

Table 10: PUT method for Response Plans

Delete a Response Plan

These messages will be received through the API as an HTTP DELETE.

Method	**DELETE**
Endpoint	/response-plans/{name-ID}/
Body	
application/json	Response Plan instance
Output	
200 *404*	*OK*, Response Plan with ID {name-ID} deleted *NOTFOUND*, Response Plan not found

Table 11: DELETE method for Response Plans

4.10 Actionables

This module is the last part of the system, and will hold the communication with the outer world. The triggering event to launch warning messages to relevant stakeholders, which might include the citizens, is the reporting that an *Alarm* situation has been confirmed.

At this point of the system we know which alert was generated and which will be the *Response Plans* to send and to whom we need to send them.

On this first iteration of the project, predefined ways of *Response Plans* will be set. Thus, these predefined methods of communication will be:

1. Do an HTTP call.
2. Send an email.
3. Send an SMS.

4.10.1 Output Notifiers

The fields needed to use each of communication means defined below, such as how the recipients are specified or what is being sent to them, can either be pre-defined as part of the response plan (mostly for automatically executed response plans), or configurable by the operator from the response plans tab (before hitting "execute response plan").

4.10.1.1 HTTP

A *Response Plan* message can be sent in an HTTP call to an API. The API will be defined as a URL, port number (optional) and endpoint (optional). The format of the message to be sent will be defined as part of the response plan. At the moment the available formats are the Common Alerting Protocol (Section 2.1.1) or simply a plain text message.

4.10.1.1.1 Plain text

Sending the message in *Plain Text* format won't follow any standard. It will send the message using our *Alert* definition.

4.10.1.1.2 Common Alerting Protocol - CAP

Sending the message in *CAP* format will allow the submission of a warning message to a Cell Broadcasting Centre, so that the warning message is distributed through reverse 112 (EU-Alert). The internal PWS alert object fields will need to be translated to match the standard. After that, the data will be written into an XML. Finally, the *capparselib*[29] open source Python library will be used in order to validate the correct CAP message generation.

4.10.1.2 Email

A platform operator, when presented with an alert, can go to the list of actions associated with that particular instance and select "Send an email", filling the destination address and the body message. This information can also be pre-filled. These emails are sent using the Simple Mail Transfer Protocol (SMTP).

[29]Capparselib Python:
https://pypi.org/project/capparselib/

4.10.1.3 SMS

A platform operator, when presented with an alert can go to the list of actions associated with that particular instance and select "Send an SMS", filling the destination number and the body message. This information can also be pre-filled. These SMS are sent using Twilio[30].

4.11 Authentication

The whole system is protected by the *API-Gateway*, *Kong*. This gateway is a router within our system. That is, with a single external IP you can access all microservices of the system. In addition, it has the advantage that you have the certainty that all your microservices can only be accessed through this single point.

For this reason, *Keycloak* has been integrated with *OpenID* in this layer, so that our entire system requires authentication, with the advantage that there is a single access site, and permissions are controlled by roles.

The following image shows an example of a GET request for an API resource. First the client needs to have the access code of *OpenID* and then add it to the request.

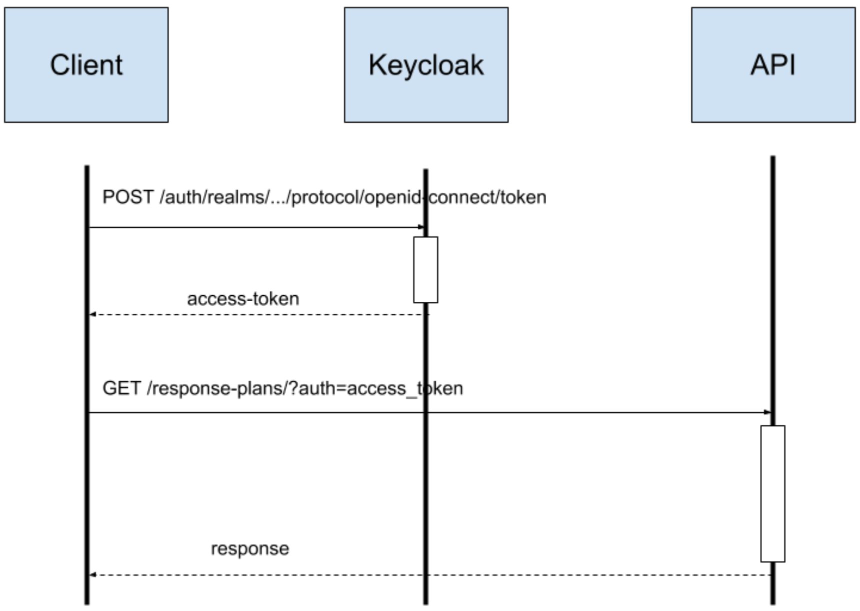

Client
Keycloak
API
POST /auth/realms/.../protocol/openid-connect/token
access-token
GET /response-plans/?auth=access_token
response

Figure 21: OpenID in *Public Warning System*.

[30]Twilio is a cloud communications platform as a service company. Allows software developers to programmatically make and receive phone calls, send and receive text messages using its web service APIs.

4.12 Business Intelligence

During the execution of the project flow information is stored that can later be processed in a data display panel. That is the objective of this module, to be able to visualize the data that we store in the timeseries database. Mainly to be able to obtain KPIs of the sensors that are analyzed continuously as well as of the actionables that generate these sensors.

4.13 Frontend

For the development of this project a combination of frontends has been implemented. The base of the frontend is derived from a *Worldsensing* product called *OneMind*. It is a dashboard for Internet Of Things applications ranging from Smart Cities to construction, consists of a map with sidebar, and permits to show data from IoT devices in realtime, among other things. In this map layers are shown with data from sensors in real time, in which you can get the information they report.

OneMind has been modified to accommodate the requirements of this project, new windows have been added to it, below are described the windows that have been developed in the frontend.

- **Real Time**: This tab shows an instance of *OneMind*. It is used as a data viewer on the map.
- **Config**: Configuration. This tab has been developed entirely in this project and aims to be used for the configuration and management of the *Public Warning System*. It is a React website.
- **BI**: Business Intelligence. This tab has been developed entirely in this project and aims to be used to obtain graphs with the data that have been obtained in the project. Both the part of sensorization and notifications to end users. And showing KPIs of the system. *Grafana* (5.3) has been used for this dashboard.
- **Alerts**: This tab shows a table with the alerts in the Real Time map. It is meant to be open while in another screen you can see the Real Time tab, and thus interact in real time.

4.14 Documentation

Each API of the project has been documented with Swagger[31]. Which is an open-source software framework that helps with the documentation, testing and consuming of RESTful Web services.

A JSON formatted document is defined, in which API endpoints, call format and response format are indicated. And, what generates the framework, is an environment where you can call and test the APIs.

In Section 7.2 there is an image of the Swagger environment with the example of the Response Plan API.

[31]Swagger:
https://swagger.io/

5 Architecture and Technology

In this section, it will be explained which are the technologies used in this project and which is the architecture decided to use here.

The main idea is that the whole project follows the microservice architecture as software development technique. That means all services are fine-grained (granular) and the protocols used are lightweight.

5.1 Why microservices?

In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

The benefits of using microservices as architecture are that decomposing an application into different smaller services improves modularity. This makes the application easier to understand, develop, test, and become more resilient to architecture erosion. Also, it helps to divide the development into several tasks. And afterwards, having the ability to deploy these services independently. Therefore, helps to have continuous delivery and deployment. [14]

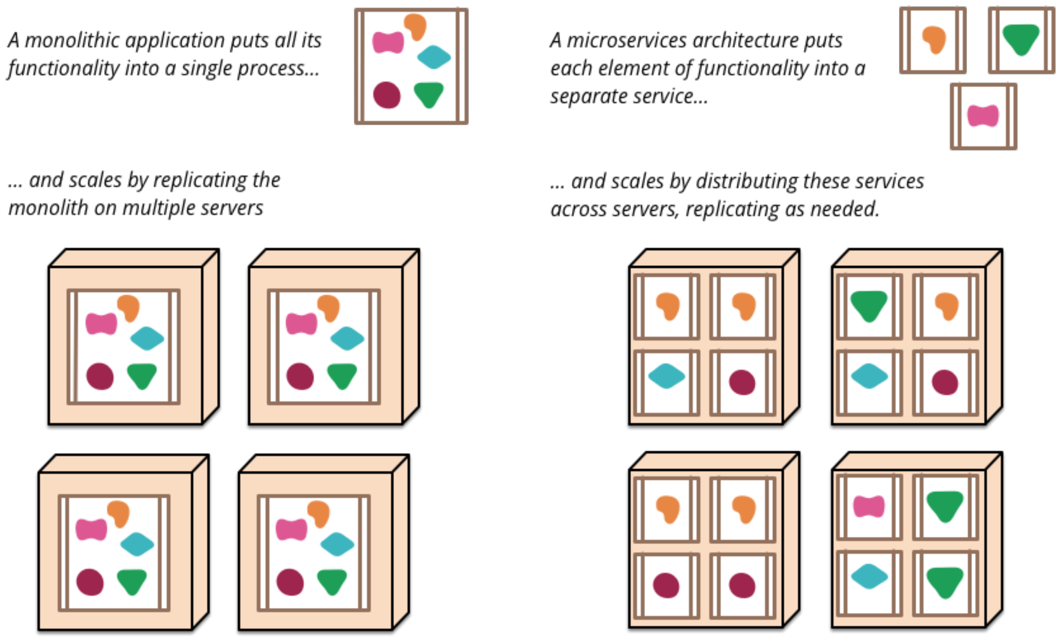

A monolithic application puts all its functionality into a single process...
... and scales by replicating the monolith on multiple servers
A microservices architecture puts each element of functionality into a separate service...
... and scales by distributing these services across servers, replicating as needed.

Figure 22: Monoliths and Microservices.[32]

[32]Martin Fowler:
https://martinfowler.com/articles/microservices.html

5.2 Architecture

Below is a diagram indicating the microservices developed in the system. Each box represents at least one microservice. The edges indicate the direction of the communication.

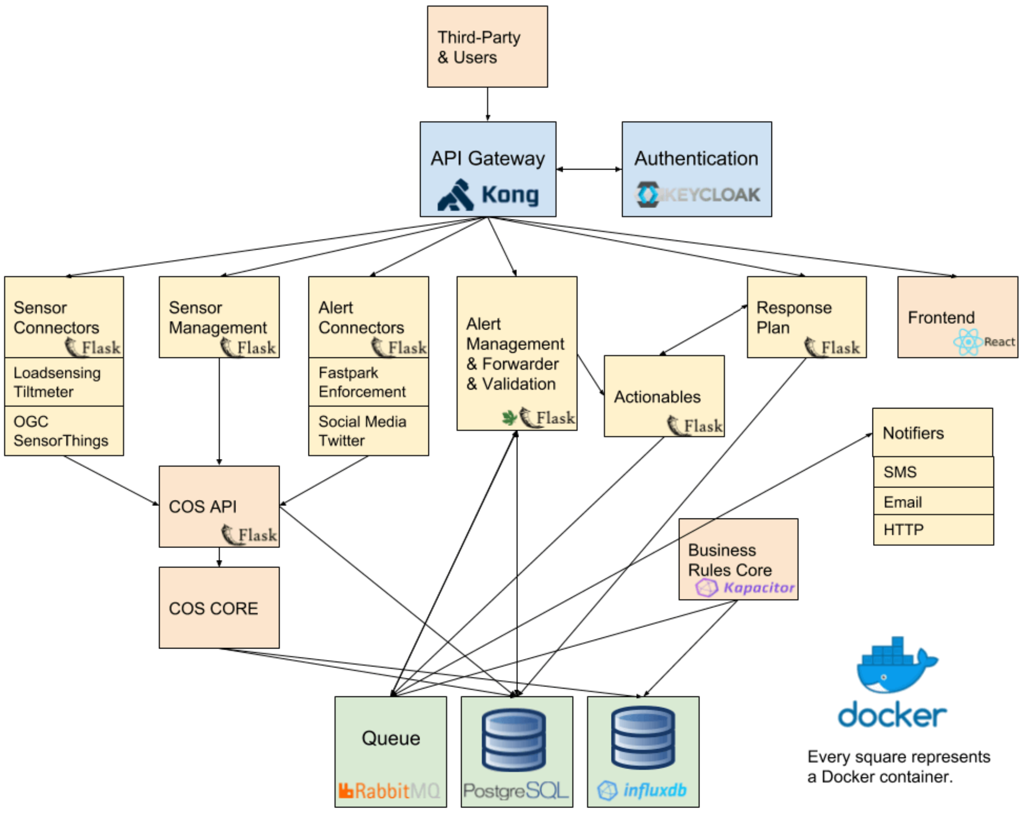


Figure 23: Overview of the Architecture and Technology in the *Public Warning System.*

5.3 Technology

Docker

Docker performs operating-system-level virtualization, also known as *containerization*. Used to run software packages called *containers*. Containers are isolated from each other and bundle their own application, tools, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and are thus more lightweight than virtual machines. Containers are created from *images* that specify their precise contents. Images are often created by combining and modifying standard images downloaded from public repositories.

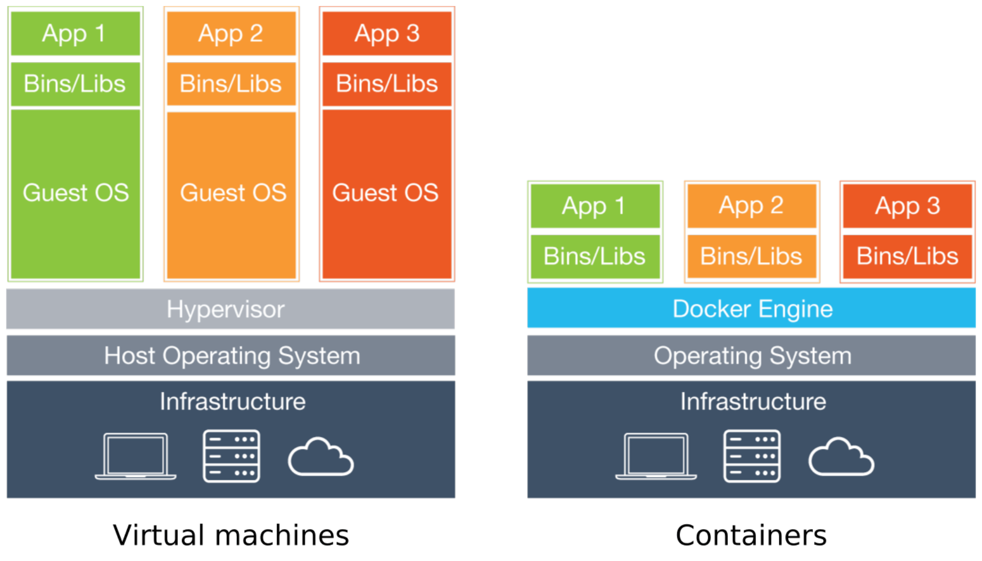


Figure 24: VM vs Containers.[33]

Kong API Gateway

Kong is an open source Orchestration Microservice API Gateway. Kong provides a flexible abstraction layer that securely manages communication between clients and microservices via API.

Once Kong is running, every client request being made to the API will hit Kong first and then be proxied to the final API. In between requests and responses Kong will execute any installed plugins, extending the API feature set. Kong effectively becomes the entry point for every API request.

[33]Docker:
https://crs4.github.io/Galaxy4Developers/lectures/06.docker_and_galaxy/

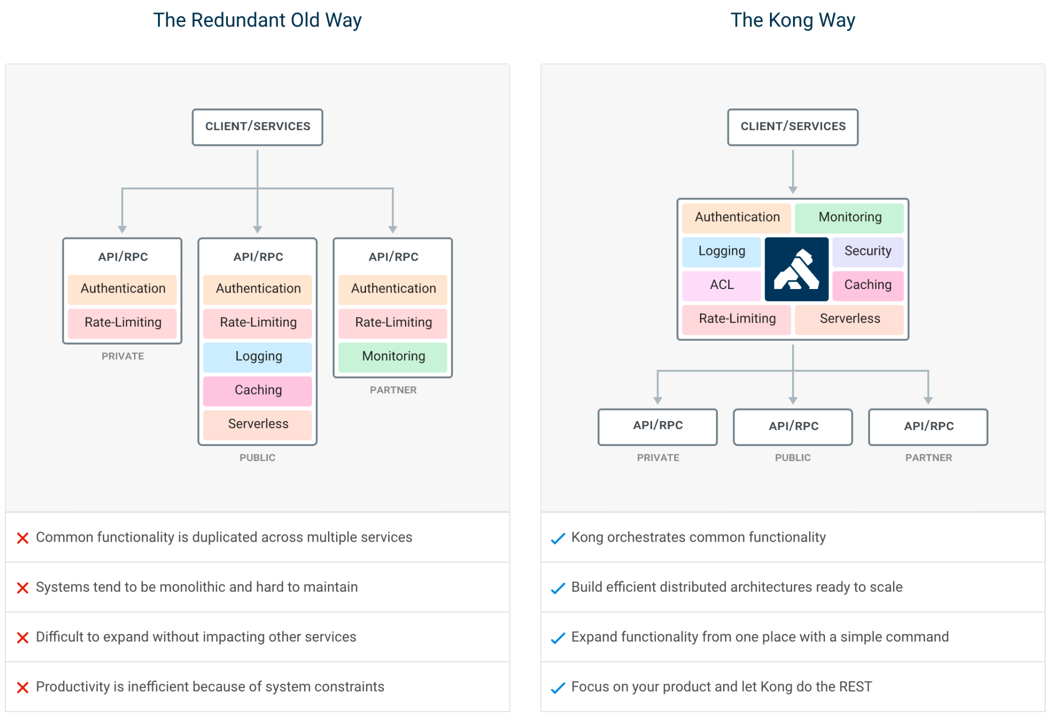

The Redundant Old Way
The Kong Way
CLIENT/SERVICES
API/RPC
Authentication
Rate-Limiting
Logging
Caching
Serverless
Monitoring
Security
ACL
PRIVATE
PUBLIC
PARTNER
Common functionality is duplicated across multiple services
Systems tend to be monolithic and hard to maintain
Difficult to expand without impacting other services
Productivity is inefficient because of system constraints
Kong orchestrates common functionality
Build efficient distributed architectures ready to scale
Expand functionality from one place with a simple command
Focus on your product and let Kong do the REST

Figure 25: Kong functionalities[34]

Keycloak

Keycloak is an open source software product to allow single sign-on with Identity Management and Access Management aimed at modern applications and services.

Flask

Flask is an open source web framework written in Python. This means flask provides you with tools, libraries and technologies that allow you to build a web application.

```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

With this code you are deploying a web server with an endpoint in *"/"* that will show a *"Hello World!"*. Accessible through a browser or a *CURL* request.

[34]Kong:
https://github.com/Kong/kong

EVE-Flask

Eve is an upper layer above *Flask*. It is also open source. Defining the endpoint and the database model, this *framework* can generate automatically a full *CRUD* (Create, read, update and delete) storage.

EVE-SQLAlchemy

EVE is pretended to be used with *MongoDB* storage, but since we use *PostgreSQL* as database, we need to use an extension for its use with *SQL*.

Note: The author of this *Master Thesis* has contributed to improving the documentation of this open source project: https://github.com/pyeve/eve-sqlalchemy/commits?author=LaQuay

PostgreSQL

PostgreSQL is an open source object-relational database management system (ORDBMS) with an emphasis on extensibility and standards compliance. It can handle workloads ranging from small single-machine applications to large Internet-facing applications (or for data warehousing).

PostgreSQL is ACID-compliant and transactional. Has updatable views and materialized views, triggers, foreign keys; it also supports functions and stored procedures.

InfluxDB

InfluxDB is an open source time series database (TSDB). It is written in Go and optimized for fast, high-availability storage and retrieval of time series data in fields such as operations monitoring, application metrics, Internet of Things sensor data, and real-time analytics. *InfluxDB* also offers an *SQL-like* query language for interacting with data.

Kapacitor

Kapacitor is an open source data processing framework that makes it easy to create alerts, run ETL jobs and detect anomalies. It can process both stream and batch data from *InfluxDB*. *Kapacitor* lets you plug in your own custom logic or user-defined functions to process alerts with dynamic thresholds, match metrics for patterns, compute statistical anomalies, and perform specific actions based on these alerts like dynamic load rebalancing.

Grafana

Grafana is an open source visualization tool that can be used on top of a variety of different data stores but is most commonly used together with *InfluxDB*, *Graphite*, and also *Elasticsearch*.

RabbitMQ

RabbitMQ is an open source message broker software that originally implemented the Advanced Message Queuing Protocol and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol, Message Queuing Telemetry Transport, and other protocols.

React

React is an open source JavaScript library for building user interfaces. *React* makes no assumptions about the rest of the technology stack used. Thus it's easy to try it out on a small feature in an existing project. In a *Model-View-Controller* (MVC) arquitecture it provides the *View* component.

6 Use-case testbed

So far we have talked about the base specification of our project but we have not discussed in detail the specific ramifications per project that co-exist. Only three European Projects appear because these are the ones that have been implemented and delivered. The details of the specifications and use cases for European Union projects, which arise from ours, are explained below.

6.1 STOP-IT

6.1.1 Functional Description

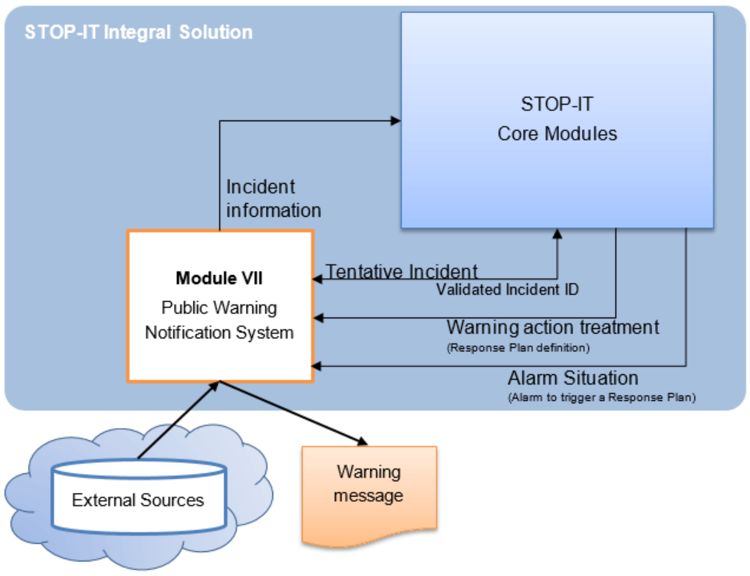


Figure 26: STOP-IT - Dataflow of STOP-IT Public Warning System

STOP-IT expects a PWS solution which is able to facilitate two-way communication for incidents and warning messaging (Figure 26).

The first area is covering the input flow of data for the STOP-IT solution. We are required to acquire data to inform about an incident from third parties, validate them, and report such incidents to the internal risk management system.

The second area is alerting users/citizens about a warning situation. The solution is required to contact user groups which are assumed to take some actions according the warning situation [Inform action due to situation, to a group]. The solution should be able to differentiate technical operators, authorities, citizens, and alert them by the most appropriate channel for each of them and with the level of detail required.

6.1.1.1 Incident detection and reporting

Data ingestion from Sensor data acquisition, to sensor data processing, basic business rule validation, Trigger tentative incident, an generating responses from the alerts generated.

In the figure below are shown the functionalities for this module. Which are explained below the figure.

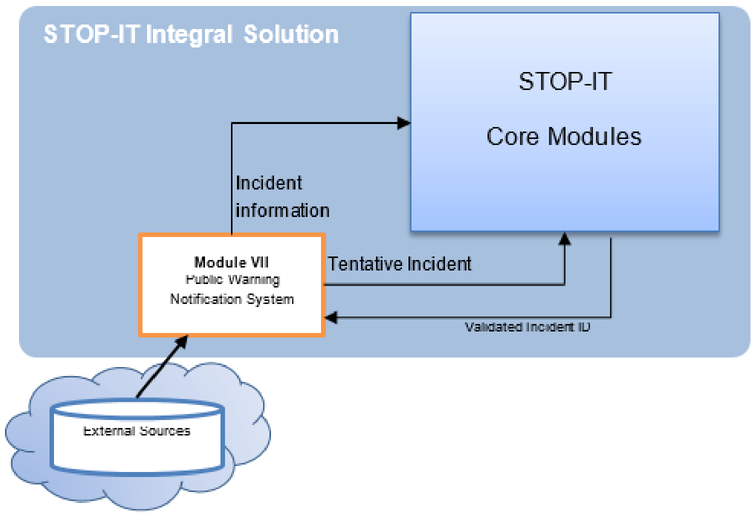


Figure 27: STOP-IT - Incident detection and reporting

1. Information is gathered from external sources (input - External Sources in diagram)
2. Incident is reported to an API, such as the Visualisation tool (output - Tentative incident in diagram)
3. Visualisation tool reports to the PWS API when an incident is validated by the operator (input - Validated Incident ID in diagram)
4. The incident is then reported to the Real Time anomaly detection system (output - Incident Information in diagram)

6.1.1.2 User/citizen alerting

In the figure below are shown the functionalities for this module. Which are explained below the figure.

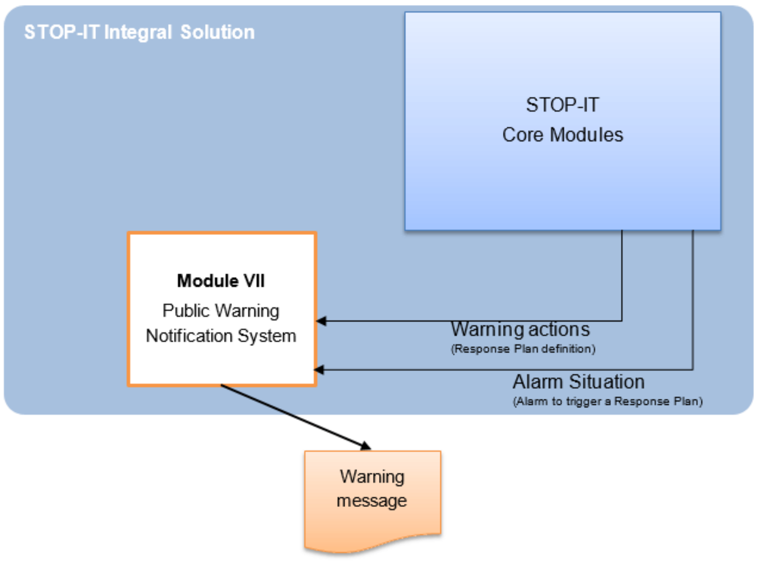

STOP-IT Integral Solution
STOP-IT Core Modules
Module VII
Public Warning Notification System
Warning actions
(Response Plan definition)
Alarm Situation
(Alarm to trigger a Response Plan)
Warning message

Figure 28: STOP-IT - User/citizen alerting

1. The visualisation module reports to the PWS Alerts API a warning message to be disseminated, as a response plan to be executed when specific situations arise (input - warning action treatment in diagram).
2. When a specific alarms situation is finally presented, and action has to be taken, the visualisation tool module informs the PWS API about the alert alarm situation presented, so that an action response plan can be activated.
3. PWS uses the most appropriate channels to distribute the information and actions to be done to the users/citizens through the Response Plan API (output - Warning message in diagram).

6.1.2 Technical documentation

6.1.2.1 Incident detection and reporting

Input from External Sources as raw sensor

The system is compatible with the *OGC SensorThings* standard defined in Section 2.1.2.

Input from External Sources as tentative alert

1. **Common Alert Protocol**
   A CAP compliant XML file will be received and parsed. This will be reported as an external incident, possibly coming from other critical infrastructures reporting a warning situation.
   - Standard defined in 2.1.1

- Implementation still not implemented, but yet defined.

2. **Social Networks - Twitter ingestion**
   The PWS accepts data coming from social networks. Particularly, a connector for *Twitter* has been developed that tracks the messages following a certain criteria.
   - Definition and implementation in Section 4.4.1.2.

6.1.2.2 Tentative Incident report and validation

The interface with the visualisation module (Module IX) is two-ways through two APIs. The Public Warning System sends information regarding a tentative incident and the visualisation module returns a status for that incident.

Expected output to the external visualization module

The information generated by the PWS will be shared using an HTTP CURL with JSON structured files.

The output from the PWS has the following parameters, defined in Alert Definition, Section 4.5.

Expected input from the external visualization module

The expected input is the reported status of the tentative incident after being shown to the visualisation interface and being managed by an operator. The data should be received as an HTTP PATCH request.

The input to the PWS received from the visualisation module has the following parameters:

Attribute	**Type**	**Values**	**Description**
alert_id	Integer	-	Mandatory. Unique. The alert ID sent to validate.
alert_status	Integer	0 (Open), 10 (Seen), 20 (Attended), 25 (Confirmed) 30 (Resolved)	Mandatory. The status of the alert.

Table 12: Definition of the input to change the *Alert Status*

The file expected to be received by the PWS is a JSON structured file, and will be submitted to a published API for this purpose.

6.1.2.3 Incident reporting

The PWS interface with the Real Time Anomaly detection system (Module VI) only sends information. Information regarding a validated incident to the Real Time Anomaly detection system.

The information generated by the PWS will be shared to an API that the Real Time Anomaly detection module should implement using a JSON structured file.

6.1.2.4 User/citizen alerting

Input of a Warning action treatment - Response Plan API

The expected input from the Visualisation module (Module IX) is the information from the warning situation that is produced, and the relevant communication actions that are required to execute.

Specification can be found in Section 4.8.

Output of a Warning action treatment - Response Plan API

1. **Email message**
   Specification can be found in Section 4.10.1.2.

2. **SMS message**
   Specification can be found in Section 4.10.1.3.

3. **HTTP CURL message**
   STOP-IT Public Warning System provides an output as an XML file which is structured as an OASIS CAP file, defined in Section 2.1.1. Specification can be found in Section 4.10.1.1.

6.2 <IMPACT>

The goal of this EU funded project is to integrate a *Parking Enforcement* solution into the Public Warning System, using parking sensor and payment occupation data as the main sources. If the system finds a mismatch between the occupation and the payments, an alert should be triggered and the warning system should deal with the subsequent actions, for example sensing an agent on-site to confirm there is an overstate offender and taking the corresponding actions.

This use case is an interesting one for our projects because it involves several aspects related to an alert management system, such as: configuring it so that it can accept data from additional sources, configure some business-specific rules that will trigger and alert, among other aspects.

Therefore, the technical documentation of this use-case can be found in the Section 4 as it is a use case with a very similar flow that our main solution, and specifically in Section 4.4.1.1 can be found the implementation for this *Fastpark* connector.

6.3 BIG-IoT

The goal of this project will be to integrate the alert platform with a marketplace (*BIG IOT*'s marketplace). This is an international initiative (BIG-IoT[35]) where several stakeholders from different countries have joined efforts to develop a horizontal IOT platform. *Worldsensing*, through its close relationship with the *Universitat Politècnica de Catalunya - Barcelona Tech* was selected as a great testbed to check the possibilities of the marketplace.

The use case is around parking sensors (but not limited to *Worldsensing*'s sensors exclusively) and the possibility of accessing multi-vendor parking information and managing their occupation-payments mismatching alerts using our alert platform.

In addition, this system must collect data from the BIG-IoT library, rather than using the corporate system of *Fastpark* itself.

Therefore, the technical documentation of this use-case can be found in the Section 4 as it is a use case with a very similar flow that our main solution, and specifically in Section 4.4.1.1 can be found the implementation for this *Fastpark* connector. And in Section 2.1.3 can be found the specification for this library.

[35]BIG-IoT:
http://big-iot.eu/

7 Results

At the beginning of the project, some objectives were proposed to be fulfilled in the project. These objectives were both the internal ones of the company and those that were proposed through European projects.

7.1 Storyboard

Since the beginning of the project, there has been a constant analysis of what should be part of the project's architecture and for what purpose. The project started from scratch (talking about the backend). Only libraries were reused to manage the different tools used within the company.

Six months later, at the conclusion of this Master Thesis, we can state that we have a functional end-to-end system for the cases specified in this document.

- We are able to read data from a *Loadsensing Tiltmeter* sensor, detect anomalies, generate alerts and proceed with the execution plan of the response to such anomaly.
- We are able to read data from sensors that implements the *OGC Sensorthings* standard.
- We are able to read data from a system that generates tentative alerts, *Fastpark Enforcement* (and its variation of *BIG IoT* and *<IMPACT>*), and proceed with the execution plan of the response to that possible alert.
- We are able to read data from an external system that generates tentative alerts, *Twitter*, and proceed with the execution plan of the response to that possible alert.

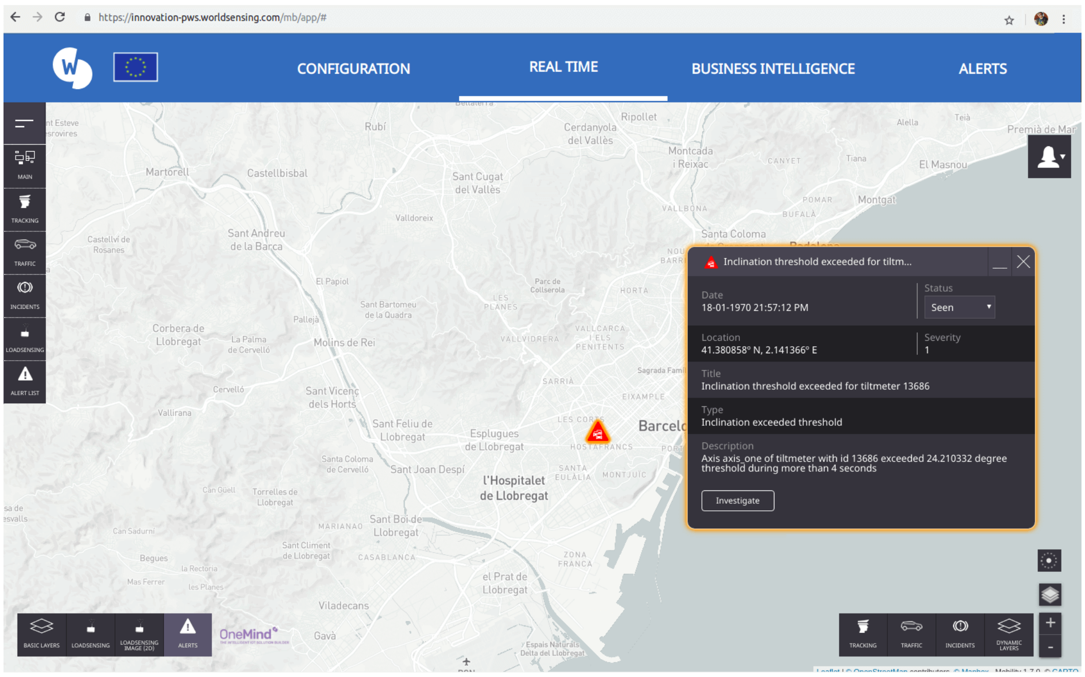
Furthermore, it has also been possible to set up a *OneMind* environment, to integrate our system with it. And along with the expansion that has been made to implement our frontend in the same environment. This process has meant an over effort because it was not contemplated in the initial planning of the project. However, having made the integration with *OneMind* will be a good point for the near future. Since it is an environment made in the same company and will allow us to continue making extensions of this project in a more convenient way.

In the following sections we will show the results of the specified use cases. And along with them, some screenshots of our frontend integrated with this *OneMind* environment.

7.1.1 Real Time

This tab shows an instance of *OneMind*. It is used as a data viewer on the map. It allows to visualize all the alerts in the shown area and, in addition, it makes possible to see the information of each one of them to make a later analysis of the shown data.

It also allows you to view more elements than just alerts. With the sidebar you can select which data package you want to visualize. For example we could see a simulator of security agents (police) circulating through the city, the flow of traffic in the city (with the corporate *Bitcarrier* technology), among other things. In the screenshot below it will be shown the *Real Time* tab with an alert. The use case will be explained in Section 7.3.1.



Inclination threshold exceeded for tiltm...

Date
18-01-1970 21:57:12 PM

Status
Seen

Location
41.380858° N, 2.141366° E

Severity
1

Title
Inclination threshold exceeded for tiltmeter 13686

Type
Inclination exceeded threshold

Description
Axis axis_one of tiltmeter with id 13686 exceeded 24.210332 degree threshold during more than 4 seconds

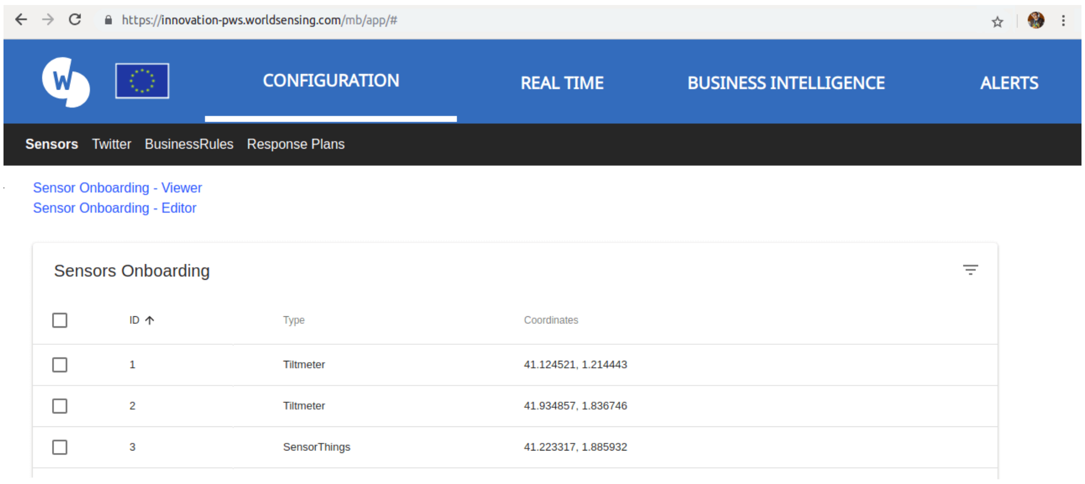
Figure 29: Real Time tab showing *OneMind*.

7.1.2 Config - Configuration

This tab has been developed entirely in this project and aims to be used for the configuration and management of the *Public Warning System*. The development of this frontend has been made by our team, and it is a React website, connected with the backend APIs that are deployed.

7.1.2.1 Sensor Onboarding

This screen snapshot shows the onboarding configuration of sensors in the system. This view is what an operator of our platform would see. In the viewer you can see the *identifiers*, *sensor type* and *coordinates* of the sensor registered in the system. Also in the editor tab, you can modify and add fields to the backend, and would be reflected in the viewer table. And it will allow the operators to manage the current sensors in the system, with all its information.


https://innovation-pws.worldsensing.com/mb/app/#
CONFIGURATION
REAL TIME
BUSINESS INTELLIGENCE
ALERTS
Sensors
Twitter
BusinessRules
Response Plans
Sensor Onboarding - Viewer
Sensor Onboarding - Editor

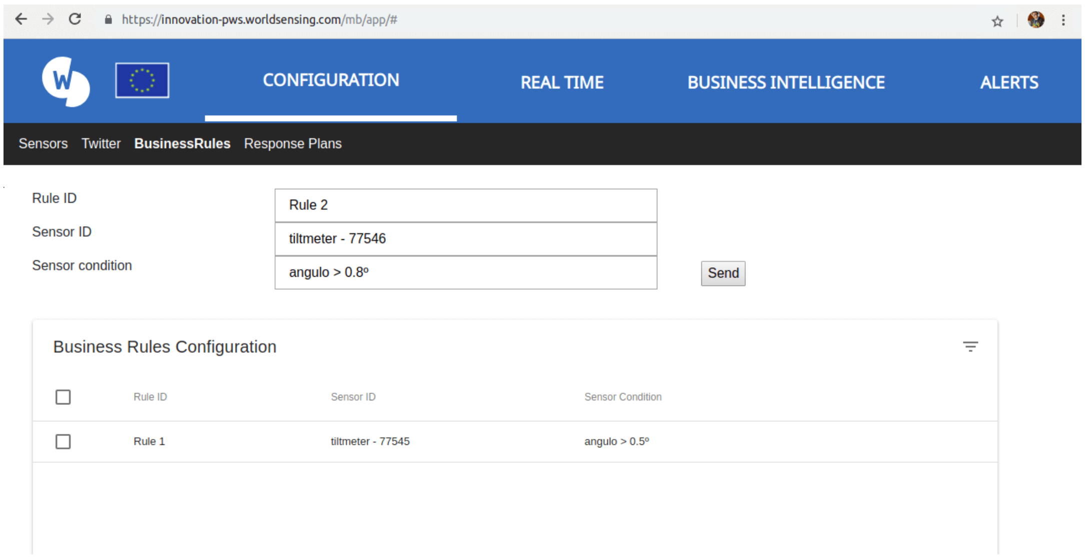
Sensors Onboarding

	ID	Type	Coordinates
☐	1	Tiltmeter	41.124521, 1.214443
☐	2	Tiltmeter	41.934857, 1.836746
☐	3	SensorThings	41.223317, 1.885932

Figure 30: Sensor Onboarding in *Public Warning System.*[36]

7.1.2.2 Business Rules

This screen snapshot shows the configuration of the business rules in the system. This view is what an operator of our platform would see. The operator would be able to identify which business rules are configured in the system and to which sensors do they apply, along with the condition that have to meet the inbound data input from that sensor to generate a tentative alert. Furthermore, the operator will be able to modify these settings and generate new business rules within the frontend.



Rule ID	Rule 2
Sensor ID	tiltmeter - 77546
Sensor condition	angulo > 0.8º

Business Rules Configuration

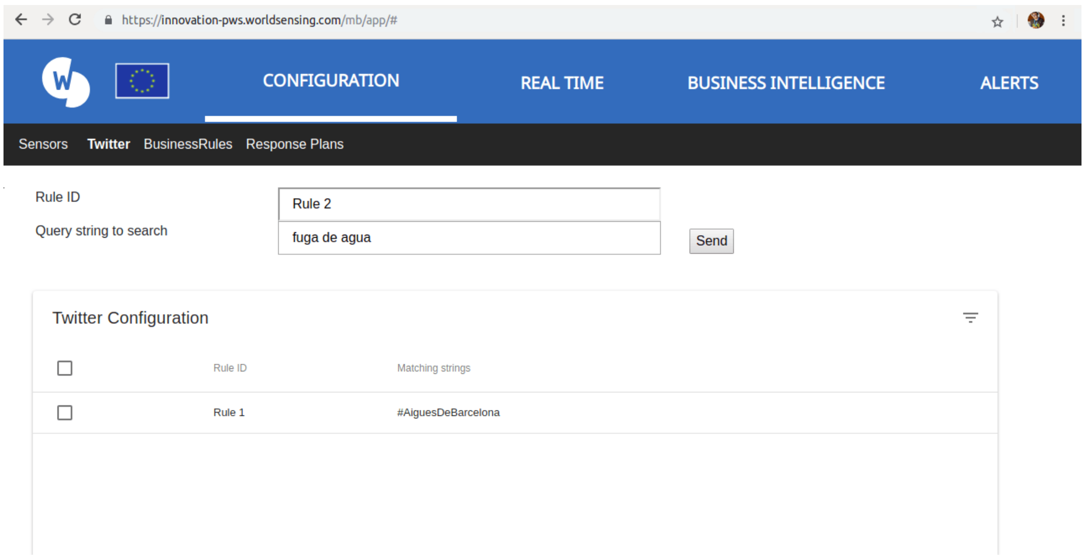
	Rule ID	Sensor ID	Sensor Condition
☐	Rule 1	tiltmeter - 77545	angulo > 0.5º

Figure 31: Configuration of Business Rules in *Public Warning System.*

(Note: This view is functional in the frontend, but it is not yet implemented in the backend. It is out of scope for this Master Thesis to configure the business rules in runtime. The backend has a predefined business rules, it can be changed while deploying the system.)

7.1.2.3 Twitter

This screen shows the configuration of the *Twitter* alert system. This view is what an operator of our platform would see. The operator would be able to identify which *Twitter* triggers are configured in the system and the matching strings that should happen in the tweets to generate a tentative alert. Furthermore, the operator will be able to modify this settings and generate new ones within the frontend.



Rule ID: Rule 2

Query string to search: fuga de agua

Twitter Configuration

Rule ID	Matching strings
Rule 1	#AiguesDeBarcelona

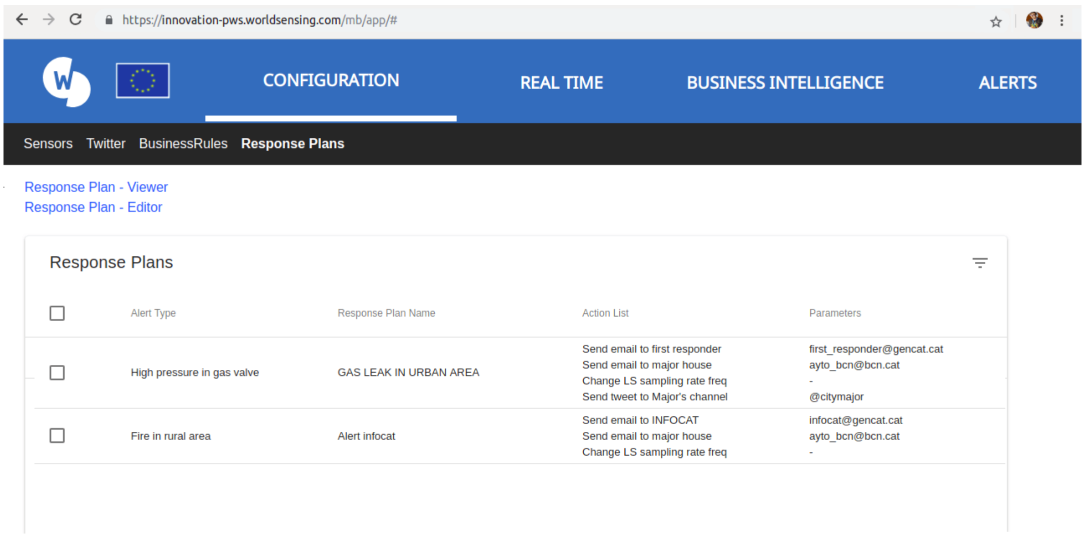
Figure 32: Configuration of *Twitter* alerts in *Public Warning System*.

(Note: This view is functional in the frontend, but it is not yet implemented in the backend. It is out of scope for this Master Thesis to configure the twitter alerts in runtime. The backend has a predefined string to match, it can be changed while deploying the system.)

7.1.2.4 Response Plans

This snapshot shows the configuration of the response plans system. This view is what an operator of our platform would see. The operator would be able to identify which response plans are configured in the system and also the information about them.

He also will be able to identify the category of the *Alerts* that is used to match the *Response Plan* and the actions that the system will execute (if the alert has the *automatic-trigger* flag activated) or propose (if the alert has the *automatic-trigger* flag deactivated) to the operator. Furthermore, the operator will be able to modify this settings and generate new *Response Plan* within the frontend.



Response Plan - Viewer
Response Plan - Editor

Response Plans

	Alert Type	Response Plan Name	Action List	Parameters
☐	High pressure in gas valve	GAS LEAK IN URBAN AREA	Send email to first responder Send email to major house Change LS sampling rate freq Send tweet to Major's channel	first_responder@gencat.cat ayto_bcn@bcn.cat - @citymajor
☐	Fire in rural area	Alert infocat	Send email to INFOCAT Send email to major house Change LS sampling rate freq	infocat@gencat.cat ayto_bcn@bcn.cat -

Figure 33: Configuration of Response Plans in *Public Warning System*.

7.1.3 BI - Business Intelligence

This tab has been developed entirely in this project and aims to be used to obtain graphs with the data that have been obtained in the project, both from sensorization and response plans sent to end users. Also to be used as a visual reference for metrics that are meaningful and relevant for the business. This way, we can inspect the variation over time of each sensor's input. *Grafana* has been used for this dashboard.

The image below shows the result of the implementation of *Grafana* in our system. *Grafana* is reading the *InfluxDB* database and generating the pre-configured dashboard. It shows the dashboard with the results of the different sensors available in the system. In the example below, we can see the variation in time of the *Loadsensing Tiltmeter* sensor (node 13686) reporting data from its two tilt axes in reference to time. The same sensor also reports the local temperature, that in our use case is only used for context information.

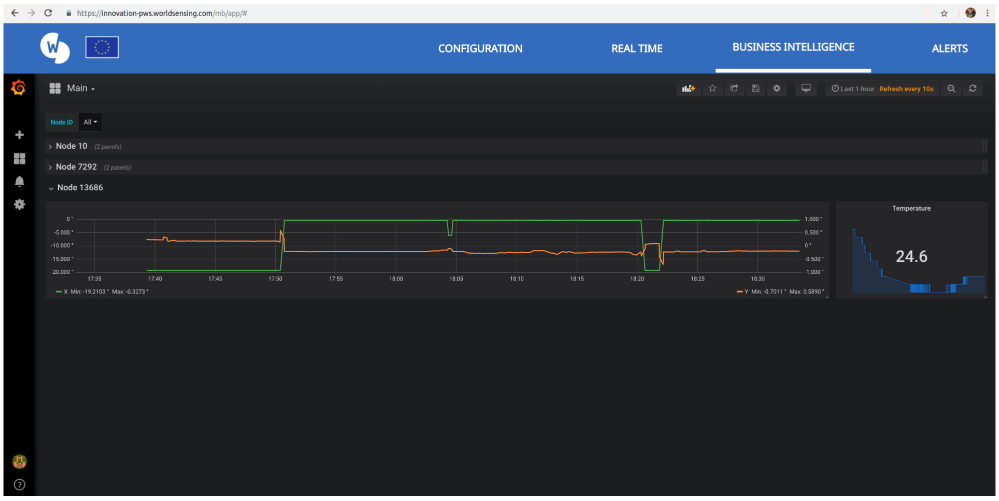
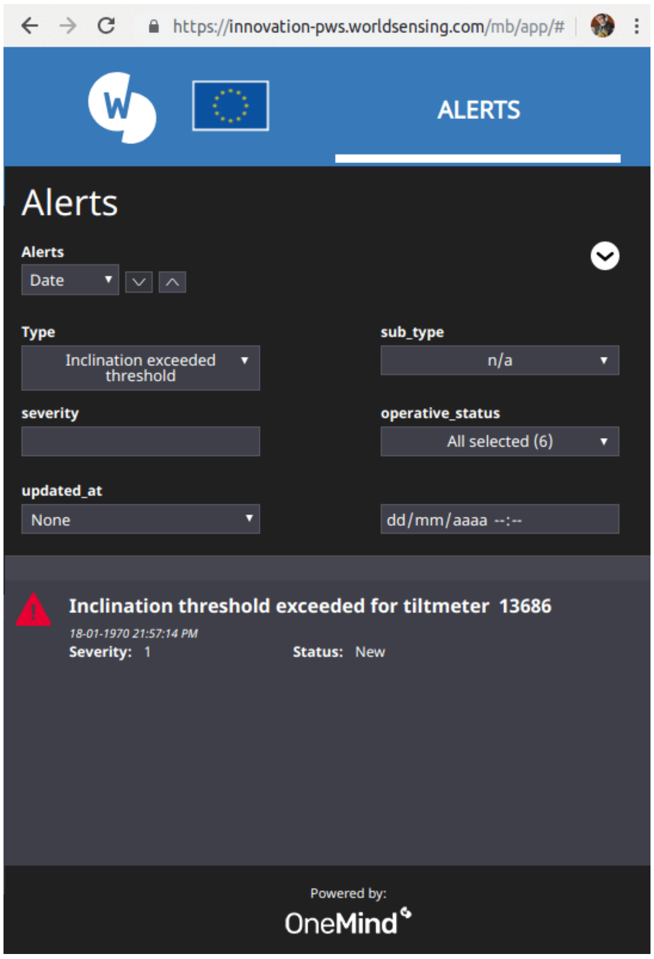


Figure 34: Business intelligence tab in *Public Warning System*.

7.1.4 Alerts

This tab shows a table with the alerts in the Real Time map. It is meant to be open while in another screen you can see the Real Time tab, and thus interact in real time with the other tab.


https://innovation-pws.worldsensing.com/mb/app/#

ALERTS

Alerts

Alerts: Date

Type: Inclination exceeded threshold

sub_type: n/a

severity:

operative_status: All selected (6)

updated_at: None

dd/mm/aaaa --:--

Inclination threshold exceeded for tiltmeter 13686

18-01-1970 21:57:14 PM

Severity: 1 Status: New

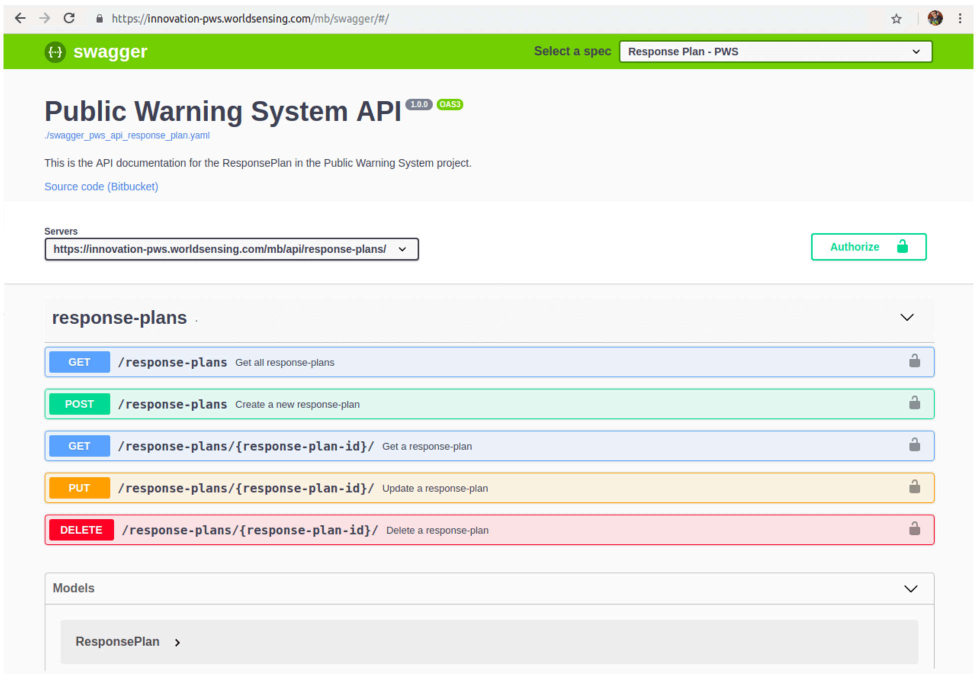
Powered by: OneMind

Figure 35: Alerts tab in *Public Warning System*.

7.2 Documentation

In addition to the internal tests code that have been implemented, an API documentation and testing framework, Swagger, has been implemented. This system reads a file in JSON, made by the developer, which follows the OpenAPI[37] standard.

Each microservice API has its own documentation and testing system. Thus, each of these APIs can be tested externally without the need for technical knowledge. Moreover, this way the work can be verified without having the source code. As an example, the image below shows a *Swagger* content for the *Response Plans API*.



https://innovation-pws.worldsensing.com/mb/swagger/#/

swagger — Select a spec: Response Plan - PWS

Public Warning System API 1.0.0 OAS3

./swagger_pws_api_response_plan.yaml

This is the API documentation for the ResponsePlan in the Public Warning System project.

Source code (Bitbucket)

Servers: https://innovation-pws.worldsensing.com/mb/api/response-plans/

Authorize

response-plans

```
GET     /response-plans  Get all response-plans
POST    /response-plans  Create a new response-plan
GET     /response-plans/{response-plan-id}/  Get a response-plan
PUT     /response-plans/{response-plan-id}/  Update a response-plan
DELETE  /response-plans/{response-plan-id}/  Delete a response-plan
```

Models

ResponsePlan

Figure 36: Swagger documentation in *Public Warning System*.

7.3 Use case validation

Below are examples of use cases that have been successfully completed. All use cases are valid in the internal objectives, but some of them are also for specific projects, which are identified in the table below.

Project	**Internal**	**STOP-IT**	**<IMPACT>**	**BIG-IoT**
Use Case				
Tiltmeter	X			
OGC + Tiltmeter	X	X		
Twitter	X	X		
FastPark	X		X	
FastPark w/ BIG-IoT				X

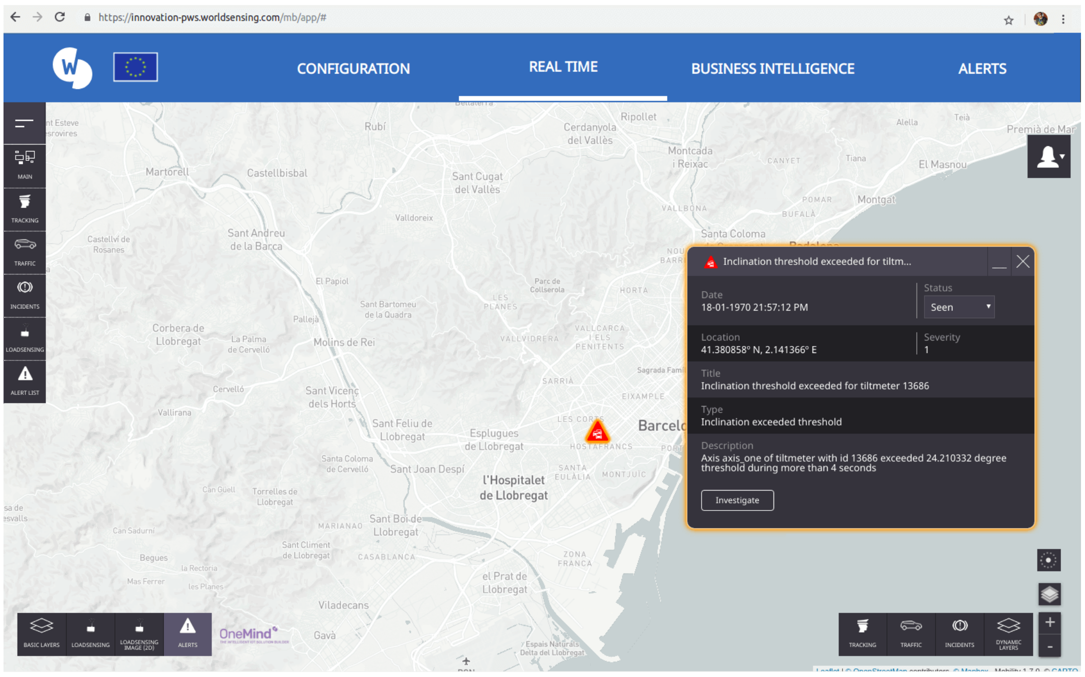
Table 13: Table showing the use-cases for the *Public Warning System*.

[37] OpenAPI standard:
https://github.com/OAI/OpenAPI-Specification

7.3.1 Use case: Loadsensing Tiltmeter

The following use case will be used internally. This is a *Tentative Alert* generated by our system from a *Raw Data Sensor*. This is the use case:

1. *Tiltmeter* sensor schema uploaded in our *Sensor API*.
2. *Tiltmeter* sensor information (such as *ID* and *coordinates*) added in our Sensors API.
3. Our *Business Rules* are set up to analyze the *Tiltmeter* information.
4. The *Response Plans* are set up.
5. The *Actionables* are defined.
6. A LoRa Gateway reports the data of this sensor to our *Sensor API*.
7. Our system detects that an anomaly has occurred. This is, the sensor detected a 20 degree deviation during more than 4 seconds. And reports this anomaly to the *Alerts API*.
8. The *Alert Management* adds this image to the map. (Current state of the image).
9. Next step is for the operator to validate the alert with its available information.
10. The alert is validated. A *Response Plan* is gathered.
    (a) If the *Action* in the *Response Plan* is configured as *automatic*. An *Actionable* is executed (e.g. send an email).
    (b) Otherwise, the system will wait for the operator confirmation to execute the *Actionable*.


https://innovation-pws.worldsensing.com/mb/app/#

Inclination threshold exceeded for tiltm...

Date
18-01-1970 21:57:12 PM

Status
Seen

Location
41.380858° N, 2.141366° E

Severity
1

Title
Inclination threshold exceeded for tiltmeter 13686

Type
Inclination exceeded threshold

Description
Axis axis_one of tiltmeter with id 13686 exceeded 24.210332 degree threshold during more than 4 seconds

Figure 37: Tiltmeter alert in *Public Warning System*.

The *Tiltmeter* that generated the alert is shown below. It is a fully functional production *Tiltmeter*. And it connects to a LoRa Gateway that is installed in the Worldsensing's office.

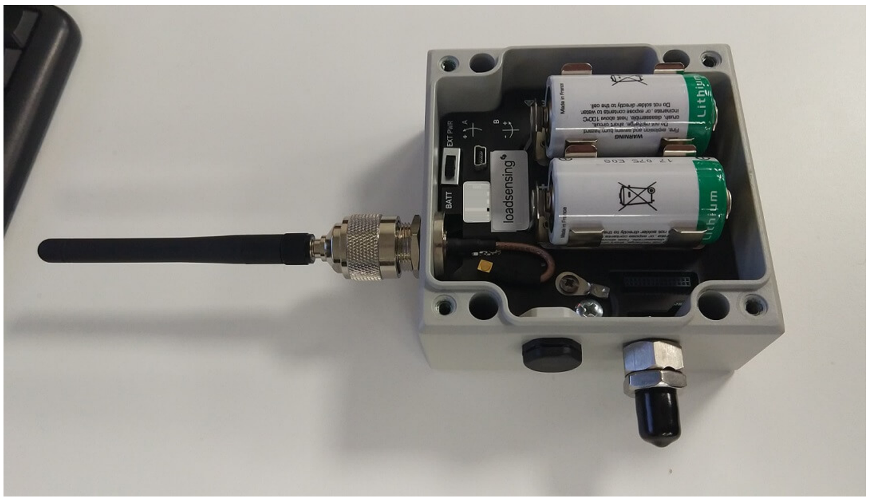


Figure 38: Working Tiltmeter from *Worldsensing*.

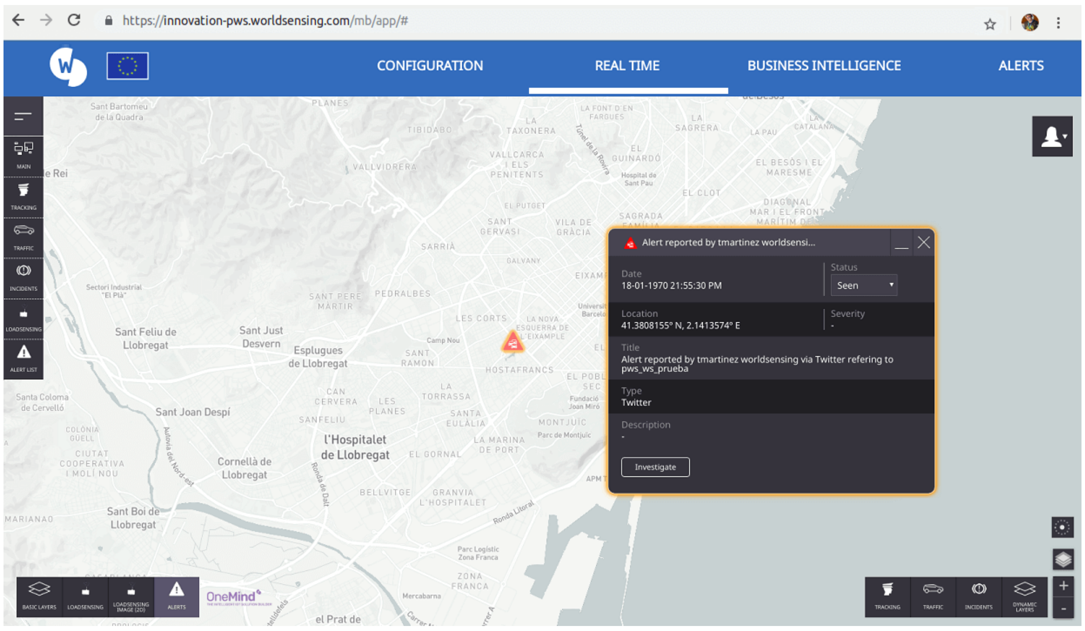
7.3.2 Use case: OGC SensorThings API with Loadsensing Tiltmeter

The use case would be similar as in Section 7.3.1, but the schema and the data of the sensor would be uploaded through the *OGC Sensorthings API* and its Sensor Connector. So that can be used for the STOP-IT European project.

7.3.3 Use case: Twitter Information

The following use case is for internal use but also for the STOP-IT European project. This is a *Tentative Alert* generated by our system from an *Alert Data Input*. This is the use case:

1. *Twitter* data schema is in our *Alerts API*.
2. The *Response Plans* are set up.
3. The *Actionables* are defined.
4. An external source reports the data to our *Alerts API* (e.g. a *Twitter* user '*tmartinez*'), with a match of pre-defined the string.
5. The *Alert Management* adds this image to the map. (Current state of the image).
6. Next step is for the operator to validate the alert.
7. The alert is validated. A *Response Plan* is gathered.
   (a) If the *Action* in the *Response Plan* is configured as *automatic*. An *Actionable* is executed (e.g. send an SMS).
   (b) Otherwise, the system will wait for the operator confirmation to execute the *Actionable*.



Alert reported by tmartinez worldsensi...

Date
18-01-1970 21:55:30 PM

Status
Seen

Location
41.3808155° N, 2.1413574° E

Severity
-

Title
Alert reported by tmartinez worldsensing via Twitter refering to pws_ws_prueba

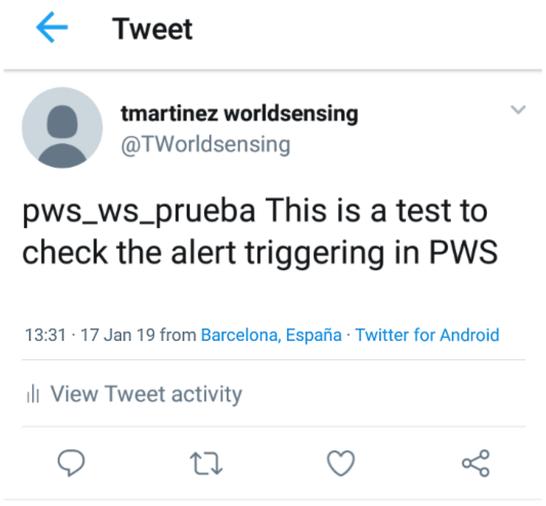
Type
Twitter

Description
-

Investigate

Figure 39: *Twitter* alert in *Public Warning System.*

Below is a tweet made in test mode, but that could be considered as a real case. The use case is that there is a user who has written the word to match, *'pws_ws_test'* and then a descriptive message. Our system detects that a new tweet has been made with this keyword and reports it to the alert system.



Tweet

tmartinez worldsensing
@TWorldsensing

pws_ws_prueba This is a test to check the alert triggering in PWS

13:31 · 17 Jan 19 from Barcelona, España · Twitter for Android

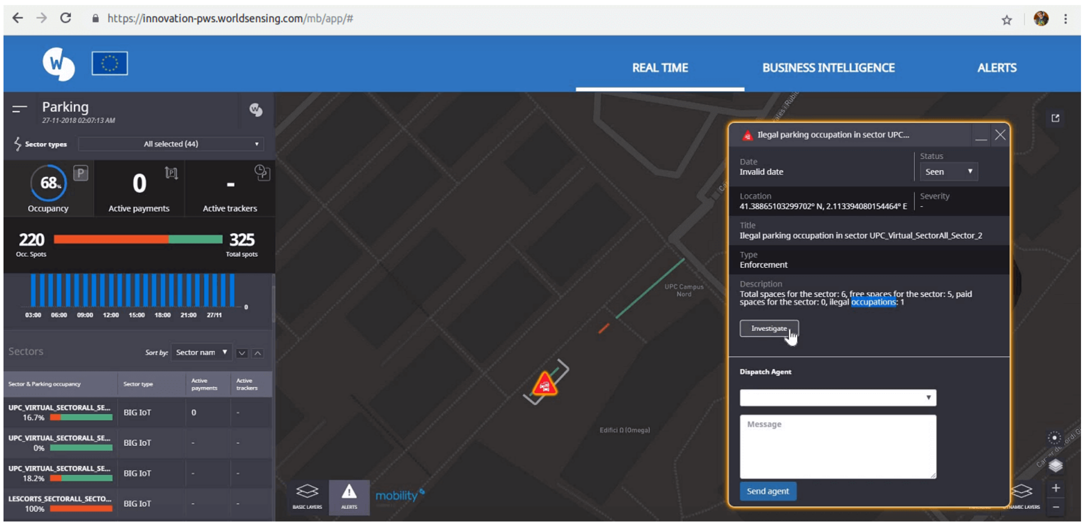
View Tweet activity

Figure 40: Tweet that generates a *Twitter* alert.

7.3.4 Use case: Fastpark Enforcement

The following use case is for internal use but also for the <IMPACT> and BIG-IoT European project. This is a *Tentative Alert* generated by our system from an *Alert Data Input*. This is the use case:

1. *Fastpark* sensor schema is in our *Alerts API*.
2. The *Response Plans* are set up.
3. The *Actionables* are defined.
4. Data gathering of the difference between the current occupation number and the number of payments:
   (a) The Fastpark Gateway reports the data to our *Alerts API*.
   (b) The BIG-IoT Gateway reports the data to our *Alerts API*.
5. The *Alert Management* adds this image to the map. (Current state of the image).
6. Next step is for the operator to validate the alert.
7. The alert is validated. A *Response Plan* is gathered.
   (a) If the *Action* in the *Response Plan* is configured as *automatic*. An *Actionable* is executed (e.g. send an email to the city council).
   (b) Otherwise, the system will wait for the operator confirmation to execute the *Actionable*.


https://innovation-pws.worldsensing.com/mb/app/#

Parking
27-11-2018 02:07:13 AM

Sector types: All selected (44)

68% Occupancy | 0 Active payments | - Active trackers

220 Occ. Spots | 325 Total spots

Sectors — Sort by: Sector nam

Sector & Parking occupancy	Sector type	Active payments	Active trackers
UPC_VIRTUAL_SECTORALL_SE... 16.7%	BIG IoT	0	-
UPC_VIRTUAL_SECTORALL_SE... 0%	BIG IoT	-	-
UPC_VIRTUAL_SECTORALL_SE... 18.2%	BIG IoT	-	-
LESCORTS_SECTORALL_SECTO... 100%	BIG IoT	-	-

Ilegal parking occupation in sector UPC...

Date: Invalid date
Status: Seen
Location: 41.38865103299702° N, 2.113394080154464° E
Severity: -
Title: Ilegal parking occupation in sector UPC_Virtual_SectorAll_Sector_2
Type: Enforcement
Description: Total spaces for the sector: 6, free spaces for the sector: 5, paid spaces for the sector: 0, ilegal occupations: 1

Dispatch Agent

Figure 41: Fastpark Enforcement alert in *Public Warning System*.

7.4 Alternative valoration

During the execution of this project, it has often been necessary to think of alternative solutions that would make it possible to follow the thread of what was planned. But at the same time adapting to the needs of the other projects in which it participates.

The project began with a simple end-to-end, in which we knew that development would have to adapt in the future. From the beginning, we opted for a microservices-based architecture, since they are the perfect tool when you have a modular system. And also when you know from the beginning that, even if it is minimal, the system must be able to adapt to specification changes.

But, in addition to having to support those changes, which were initially minimal, with the STOP-IT project we had to modify the behaviour of the system, as in this the process of filtering and validating an alert is performed by a third party module, that then feeds the results back into our platform. Because of this interaction with a external module, we had to provide a thorough description of the external interface, particularly of the *Alert Management* module and also in the *Alert Forwarder* module.

Also, at the beginning of the project it was planned that the frontend that would be used in this project would be one generated by ourselves, not as specific or complex as *OneMind.* Finally and after evaluating it, it was decided that although it would be an extra effort to make the integration with that system, then it would be a good starting point to be able to work with the data of our system.

7.5 Scalability

The entire system is designed to be completely scalable. The fact that it is based on microservices implies that the different modules of the system are designed to be scalable horizontally (adding more infrastructure), but if we have to scale it up to a very high level, we may begin to have drawbacks and analysis to perform[22].

Scaling would consist of analyzing the use of each microservice and splitting the heaviest ones into different machines. Although it should be kept in mind that this is not infinite and is limited to a certain extent, for example in the dependencies of databases. Several databases could be used and the system would relax in use, but it is not infinite. *"But scaling these traditional relational database management systems (RDBMS) isn't easy because they were originally designed to run on a single node"*[13].

In addition, while performing the architecture of the system, the main factors that could generate bottlenecks have been taken into account. Such as the entry of data from external sources and the generation of *Actionables.* These microservices communicate internally through queues. Therefore, the maximum available throughput will always be used without overloading the system, all data that cannot be processed in real time will be queued for further processing.

7.6 Validation

7.6.1 November 2018 - Smart City Expo World Congress 2018

The project had an early presentation in the *Smart City Expo World Congress 2018* in Barcelona. Showing a use case with the *Loadsensing Tiltmeter*, starting with the receiving of the data from the Tiltmeter and then applying the Business Rules and, if meets the rule requirements, generating an alert.

7.6.2 December 2018 - <IMPACT>

The project had a presentation at an event of *<IMPACT> Connected Car* in Málaga last December 2018. In this presentation the Public Warning System platform was pitched using the *Alerts* system generated by *Fastpark Enforcement.*

A group composed of technical, business and strategical evaluators attended the presentation, which was a success. And the PWS was approved by the evaluators as innovative and with a clear target and viability plan, consequently granting the committed funds to Worldsensing.

In the image on the right we can see *José Gorchs* presenting the project with *OneMind* and the *Fastpark Enforcement* integration in this project.

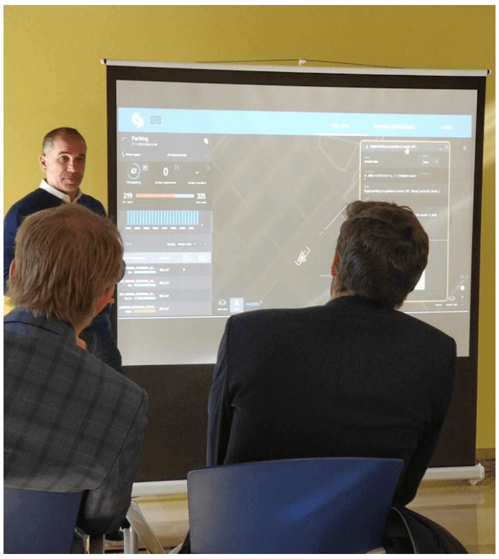


Figure 42: *Public Warning System* presentation in <IMPACT>

7.6.3 December 2018 - BIG IoT

The project had a presentation at an event of *BIG IoT* in Barcelona last December 2018. In this presentation the Public Warning System platform was shown together with the system of Alerts generated by *Fastpark Enforcement* through the *BIG-IoT* ecosystem.

The objective was to demonstrate that we could use the data from their marketplace for something useful, in our case to make a parking enforcement service with alerts. The presentation was a success.

7.6.4 January 2019 - STOP-IT

The technical documentation of the project will be presented on its first iteration in the *STOP-IT* review of January 2019.

7.6.5 February 2019 - Internal

There are several use case currently being evaluated internally at *Worldsensing* to leverage on the work that is being done with the PWS. These use cases range from rail applications to computing power management to cybersecurity. All of these use cases have different stakeholder internally at *Worldsensing* and we will be conducting several demos and market (through our sales department) validations in the upcoming months. The first milestone will be mid February, where we will present what we have achieved so far internally and gathering the feedback for further evaluation and consideration of new/enhanced features.

8 Conclusions

During the implementation of the project many ideas for potential improvements and new concepts have emerged to experiment with them. Some of them were discussed in Section 8.1. However, due to the limited time and resources available, they have been left for future implementations, as the project continues after this initial release.

As it has been reflected in the previous sections, the project has reached the January milestone correctly. Even with the congratulations of other partners of the European Projects. It has been possible to confirm a clear evolution in the project in the short period of time that has elapsed and with the limited resources available for its development. Although there is still a long way to go.

However, there is still a need to develop the configuration system externally of the Business Rules, and to be able to choose manually which response plans are desired to carry out of those proposed by the system.

We have reached this January milestone with a system working end-to-end for several cases of sensors and types of alerts. And being able to forward alerts to external operators. In addition to being able to generate response plans from consumable data, which is the main reason for this Master Thesis project.

8.1 Future implementation

So far, the system implements the full life-cycle of an alert: from reading the data, its subsequent analysis, and sending the response plans. However, the possibility of closing the cycle by the operator is not yet contemplated. Thus, having the possibility of ending the cycle with the feedback of the operator, who reads and executes the *Response Plan*, would be the next natural feature to onboard.

For this project it was planned to implement solutions for five *European* and two Catalan projects. In this same project the result for two *European* projects (*<IMPACT>* and *BIG-IoT*), and a first version for another (STOP-IT), has been completely delivered. It is therefore necessary to continue working to complete the three *European* projects and the two remaining *Catalan* projects.

Also, to finish implementing the compatibility with OGC, being such a wide standard, part of it has been implemented, specifically for the sensors that send numbers (integers and floats) or lists of numbers as data. With regards to our project, this level of functionality is more than enough. But this standard also supports *strings*, among other things.

By the end of this project we managed to support the output of the alerts in *Response Plans* in CAP format. However, the next steps are to be able to receive inputs with this standard.

In summary, as a result of the successful closure of this Master Thesis, it is expected to continue this project. Furthermore, it is planned that by this 2019 more *European* projects will be incorporated into it.

8.2 Personal assessment

Before the realization of this project, my knowledge in the architecture management environment of a software project was very limited, since my previous experience had almost always been in the field of development and mobile architectures.

That's why the fact of being able to start this project from scratch, has given me an enormous knowledge both in the field of how to make the architecture of a software system focused on APIs platforms all the way up to the implementation level. Before starting this project, I had managed at most two or three Docker containers simultaneously. While in this project, we have around thirty microservices running concurrently.

Moreover, the fact that I have made major decisions for the project architecture, as well as in the backend and frontend, has given me a wide vision of the elements to take into account when planning platforms of this kind.

Overall, I am very satisfied with this Master Thesis that has been achieved to this point. The project continues, and it consists of an introduction to a PhD, which will also be carried out in Worldsensing together with the UPC, within the Computer Science department. Let's hope that it continues in such a satisfactory way as this project.

9 References

[1] Warning Systems Inc E. Jones. “Common Alerting Protocol Version 1.2”. In: (2010). URL: http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.pdf.

[2] United Nations Environment Programme. “Early Warning Systems: A State of the Art Analysis and Future Directions”. In: (2012). URL: https://na.unep.net/siouxfalls/publications/Early_Warning.pdf.

[3] Practical Action. *Early Warning Systems.* URL: https://policy.practicalaction.org/projects/ews/. (accessed: 25.06.2018).

[4] cPrime - Agile Consultants. *What is Agile? What is Scrum?* URL: https://www.cprime.com/resources/what-is-agile-what-is-scrum/. (accessed: 21.06.2018).

[5] European Emergency Number Association. *EENA Operations Document.* URL: http://www.eena.org/uploads/gallery/files/pdf/2015_07_15_PWS_Final.pdf. (accessed: 25.06.2018).

[6] Generalitat de Catalunya. *RIS3CAT - Estratègia de recerca i innovació per a l'especialització intel·ligent de Catalunya.* URL: http://catalunya2020.gencat.cat/ca/estrategies/ris3cat/. (accessed: 24.06.2018).

[7] CORDIS Horizon 2020 - European Commission. *BIG IoT - Bridging the Interoperability Gap of the Internet of Things.* URL: https://cordis.europa.eu/project/rcn/200833_en.html. (accessed: 22.11.2018).

[8] CORDIS Horizon 2020 - European Commission. *¡IMPACT¿ Connected Car.* URL: https://cordis.europa.eu/project/rcn/209162_en.html. (accessed: 22.11.2018).

[9] CORDIS Horizon 2020 - European Commission. *mF2C - Towards an Open, Secure, Decentralized and Coordinated Fog-to-Cloud Management Ecosystem.* URL: https://cordis.europa.eu/project/rcn/206164_en.html. (accessed: 24.06.2018).

[10] CORDIS Horizon 2020 - European Commission. *RIS3CAT - Research and Innovation Strategy for the Smart Specialisation of Catalonia.* URL: https://ec.europa.eu/growth/tools-databases/regional-innovation-monitor/policy-document/research-and-innovation-strategy-smart-specialisation-catalonia-ris3cat/. (accessed: 24.06.2018).

[11] CORDIS Horizon 2020 - European Commission. *SMESEC - Protecting Small and Medium-sized Enterprises digital technology through an innovative cyber-SECurity framework.* URL: https://cordis.europa.eu/project/rcn/210805_en.html. (accessed: 24.06.2018).

[12] CORDIS Horizon 2020 - European Commission. *STOP-IT - Strategic, Tactical, Operational Protection of water Infrastructure against cyber-physical Threats.* URL: https://cordis.europa.eu/project/rcn/210216_en.html. (accessed: 22.06.2018).

[13] PJ Hagerty - Crate.io. *The Rise of the Distributed SQL Database.* URL: https://crate.io/a/rise-distributed-sql-database/. (accessed: 16.01.2019).

[14] Martin Fowler. *Microservices a definition of this new architectural term.* URL: https://martinfowler.com/articles/microservices.html. (accessed: 30.12.2018).

[15] Open Geospatial. *OGC SensorThings API Part 1: Sensing.* URL: http://docs.opengeospatial.org/is/15-078r6/15-078r6.html. (accessed: 18.12.2018).

[16] European Telecommunications Standards Institute. *Emergency Communications (EMTEL).* URL: http://www.etsi.org/deliver/etsi_ts/102900_102999/102900/01.01.01_60/ts_102900v010101p.pdf. (accessed: 25.06.2018).

[17] EU - BIG IoT. *Bridging the Interoperability Gap of the Internet of Things.* URL: https://big-iot.github.io/. (accessed: 01.01.2019).

[18] Agile Manifesto. *Manifesto for Agile Software Development.* URL: http://agilemanifesto.org/. (accessed: 25.06.2018).

[19] MISP. *Malware Information Sharing Platform and Threat Sharing.* URL: https://github.com/MISP/MISP. (accessed: 16.12.2018).

[20] Agile in a nutshell. *What is Agile?* URL: http://www.agilenutshell.com/. (accessed: 21.06.2018).

[21] PreventionWeb. *Warning System - Definition.* URL: https://www.preventionweb.net/terminology/view/478. (accessed: 29.11.2018).

[22] Production-Ready Microservices by Susan J. Fowler. *Chapter 4. Scalability and Performance.* URL: https://www.oreilly.com/library/view/production-ready-microservices/9781491965962/ch04.html. (accessed: 16.01.2019).

[23] Wikipedia. *Agile - Overview.* URL: https://en.wikipedia.org/wiki/Agile%5C_software%5C_development%5C#Overview/. (accessed: 25.06.2018).

[24] Wikipedia. *Emergency population warning.* URL: https://en.wikipedia.org/wiki/Emergency_population_warning/. (accessed: 30.07.2018).

[25] Wikipedia. *Warning System - Definition.* URL: https://en.wikipedia.org/wiki/Warning%5C_system. (accessed: 27.08.2018).

10 Appendix

10.1 OGC SensorThings example

```json
{
   "name":"Living Room",
   "description":"My Living Room",
   "properties":{
      "style":"Cozy",
      "balcony":true
   },
   "Locations":[
      {
         "name":"My Living Room",
         "description":"The living room of Fraunhoferstr. 1",
         "encodingType":"application/vnd.geo+json",
         "location":{
            "type":"Point",
            "coordinates":[
               8.4259727,
               49.015308
            ]
         }
      }
   ],
   "Datastreams":[
      {
         "name":"Temperature Living Room",
         "description":"The temperature in my living room",
         "observationType":"http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
         "unitOfMeasurement":{
            "name":"Centigrade",
            "symbol":"C",
            "definition":"http://www.qudt.org/qudt/owl/1.0.0/unit/Instances.html#DegreeCentigrade"
         },
         "Sensor":{
            "name":"DHT22/Temperature",
            "description":"Temperature sensor of a DHT22",
            "encodingType":"application/pdf",
            "metadata":"https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf"
         },
         "ObservedProperty":{
            "name":"Temperature",
            "definition":"http://www.qudt.org/qudt/owl/1.0.0/quantity/Instances.html#ThermodynamicTemperature",
            "description":"The temperature."
         },
         "Observations":[
            {
               "phenomenonTime":"2019-03-14T10:00:00Z",
               "result":21.0
            },
            {
               "phenomenonTime":"2019-03-14T10:01:00Z",
               "result":21.1
            },
            {
               "phenomenonTime":"2019-03-14T10:02:00Z",
               "result":19.0
            },
            {
               "phenomenonTime":"2019-03-14T10:03:00Z",
               "result":19.1
            },
            {
               "phenomenonTime":"2019-03-14T10:04:00Z",
               "result":19.2
            }
         ]
      },
      {
         "name":"Humidity Living Room",
         "description":"The humidity in my living room",
         "observationType":"http://www.opengis.net/def/observationType/OGC-O
         ↪  M/2.0/OM_Measurement",
         "unitOfMeasurement":{
            "name":"percentage",
            "symbol":"%",
            "definition":"https://en.wikipedia.org/wiki/Percentage"
         },
         "Sensor":{
            "name":"DHT22/Humidity",
            "description":"Relative humidity sensor of a DHT22",
            "encodingType":"application/pdf",
            "metadata":"https://www.sparkfun.com/datasheets/Sensors/Temperat
            ↪  ure/DHT22.pdf"
         },
         "ObservedProperty":{
            "name":"Relative Humidity",
            "definition":"https://en.wikipedia.org/wiki/Relative_humidity",
            "description":"The relative humidity"
         },
         "Observations":[
            {
               "phenomenonTime":"2019-03-14T10:00:00Z",
               "result":40.0
            },
            {
               "phenomenonTime":"2019-03-14T10:01:00Z",
               "result":39.1
            },
            {
               "phenomenonTime":"2019-03-14T10:02:00Z",
               "result":42.0
            },
            {
               "phenomenonTime":"2019-03-14T10:03:00Z",
               "result":41.9
            },
            {
               "phenomenonTime":"2019-03-14T10:04:00Z",
               "result":41.8
            }
         ]
      }
   ],
   "MultiDatastreams":[
      {
         "name":"Wind Balcony",
         "description":"The Wind on the balcony",
         "observationType":"http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_ComplexObservation",
         "multiObservationDataTypes":[
            "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
            "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement"
         ],
         "unitOfMeasurements":[
            {
               "name":"m/s",
               "symbol":"m/s",
               "definition":"ucum:m/s"
            },
            {
               "name":"deg",
               "symbol":"deg",
               "definition":"ucum:deg"
            }
         ],
         "Sensor":{
            "name":"Wind Sensor Type XYZ",
            "description":"Wind Velocity and direction",
            "encodingType":"text",
            "metadata":"A description of this sensor should go here."
         },
         "ObservedProperties":[
            {
               "name":"Wind Velocity",
               "definition":"http://dbpedia.org/page/Wind_speed",
               "description":"The wind speed."
            },
            {
               "name":"Wind Direction",
               "definition":"http://dbpedia.org/page/Wind_direction",
               "description":"The wind direction."
            }
         ],
         "Observations":[
            {
               "phenomenonTime":"2019-03-14T10:00:00Z",
               "result":[
                  5.1,
                  40
               ]
            },
            {
               "phenomenonTime":"2019-03-14T10:01:00Z",
               "result":[
                  5.3,
                  44
               ]
            },
            {
               "phenomenonTime":"2019-03-14T10:02:00Z",
               "result":[
                  5.2,
                  49
               ]
            },
            {
               "phenomenonTime":"2019-03-14T10:03:00Z",
               "result":[
                  4.7,
                  41
               ]
            },
            {
               "phenomenonTime":"2019-03-14T10:04:00Z",
               "result":[
                  4.1,
                  42
               ]
            }
         ]
      }
   ]
}
```

10.2 OGC SensorThings Tiltmeter simulator example

tiltmeter_datastream_template.json

```
{
    "name": "Inclination angle of the building",
    "description": "Inclination angle of the building",
    "observationType": "http://www.opengis.net/def/observationType/OGC-OM/2.
    ↪  0/OM_Measurement",
    "unitOfMeasurement": {
        "name": "Degrees",
        "symbol": "°",
        "definition": "https://en.wikipedia.org/wiki/Degree_(angle)"
    },
    "Thing": {"@iot.id": {thing_id}},
    "ObservedProperty": {"@iot.id": {observedproperty_id}},
    "Sensor": {"@iot.id": {sensor_id}}
}
```

Things.json

```
{
  "name": "Tilt Monitoring System",
  "description": "Sensor system monitoring buildings inclination",
  "Locations": [
        {
            "name": "Virtual location",
            "description": "Virtual location",
            "encodingType": "application/vnd.geo+json",
            "location": {
                "type": "Point",
                "coordinates": [8.4259727, 49.015308]
            }
        }
    ]
}
```

Sensors.json

```
{
  "name": "LS-G6/Tiltmeter",
  "description": "Tilt",
  "encodingType": "application/pdf",
  "metadata": "https://www.worldsensing.com/wp-content/uploads/2016/09/LS-G6
  ↪  -Tiltmeter-Datasheet.pdf"
}
```

ObservedProperties.json

```json
{
  "name": "Tilt",
  "description": "The angle",
  "definition":
   ↪  "http://www.qudt.org/qudt/owl/1.0.0/quantity/Instances.html#Angle"
}
```

10.3 Twitter ingestion example

```
{
   "created_at":"Mon Jan 14 08:51:42 +0000 2019",
   "id":1084734790316838913,
   "id_str":"1084734790316838913",
   "Text":"water leak in Sants Station",
   "truncated":"False",
   "entities":{
      "hashtags":[ ],
      "symbols":[ ],
      "user_mentions":[ ],
      "urls":[ ]
   },
   "metadata":{
      "iso_language_code":"pl",
      "result_type":"recent"
   },
   "source":"<a href='http://twitter.com/download/android'
   ↪  rel='nofollow'>Twitter for Android</a>",
   "in_reply_to_status_id":"None",
   "in_reply_to_status_id_str":"None",
   "in_reply_to_user_id":"None",
   "in_reply_to_user_id_str":"None",
   "in_reply_to_screen_name":"None",
   "user":{
      "id":1068544762968506374,
      "id_str":"1068544762968506374",
      "name":"tmartinez worldsensing",
      "screen_name":"TWorldsensing",
      "location":"",
      "description":"",
      "url":"None",
      "entities":{
         "description":{
            "urls":[ ]
         }
      },
      "protected":"False",
      "followers_count":0,
      "friends_count":0,
      "listed_count":0,
      "created_at":"Fri Nov 30 16:38:18 +0000 2018",
      "favourites_count":0,
      "utc_offset":"None",
      "time_zone":"None",
      "geo_enabled":"True",
      "verified":"False",
      "statuses_count":62,
      "lang":"en",
      "contributors_enabled":"False",
      "is_translator":"False",
      "is_translation_enabled":"False",
      "profile_background_color":"F5F8FA",
      "profile_background_image_url":"None",
      "profile_background_image_url_https":"None",
      "profile_background_tile":"False",
      "profile_image_url":"http://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png",
      "profile_image_url_https":"https://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png",
      "profile_link_color":"1DA1F2",
      "profile_sidebar_border_color":"C0DEED",
      "profile_sidebar_fill_color":"DDEEF6",
      "profile_text_color":"333333",
      "profile_use_background_image":"True",
      "has_extended_profile":"False",
      "default_profile":"True",
      "default_profile_image":"True",
      "following":"False",
      "follow_request_sent":"False",
      "notifications":"False",
      "translator_type":"none"
   },
   "geo":{
      "type":"Point",
      "coordinates":[
         41.3808509,
         2.1413252
      ]
   },
   "coordinates":{
      "type":"Point",
      "coordinates":[
         2.1413252,
         41.3808509
      ]
   },
   "place":{
      "id":"1a27537478dd8e38",
      "url":"https://api.twitter.com/1.1/geo/id/1a27537478dd8e38.json",
      "place_type":"city",
      "name":"Barcelona",
      "full_name":"Barcelona, Spain",
      "country_code":"ES",
      "country":"Spain",
      "contained_within":[

      ],
      "bounding_box":{
         "type":"Polygon",
         "coordinates":[
            [
               [
                  2.0524766,
                  41.3170475
               ],
               [
                  2.2299781,
                  41.3170475
               ],
               [
                  2.2299781,
                  41.4682658
               ],
               [
                  2.0524766,
                  41.4682658
               ]
            ]
         ]
      },
      "attributes":{

      }
   },
   "contributors":"None",
   "is_quote_status":"False",
   "retweet_count":0,
   "favorite_count":0,
   "favorited":"False",
   "retweeted":"False",
   "lang":"pl"
}
```

10.4 Examples

The following sections provide examples on how to simulate some API calls against our alert system.

10.4.1 Send tentative incident from Twitter

As it would be received if at least one Twitter user sent a message matching the string search pattern. Firstly the rule for this alert needs to be posted. In this case, the rule indicates that the alert comes from Twitter, and what the string search pattern was.

```
POST /api/alerts/rules
```

```
{
    "id":"pws worldsensing incidence test",
    "measure_name":"Incidence",
    "measure_unit":"Incidence geotag",
    "name":"Tweet matching rule"
}
```

After the rule has been created, an alert corresponding to this rule can be posted:

```
POST /api/alerts/alerts
```

```
{
   "meta_type":"meta_data",
   "related_item_id":"mvila worldsensing",
   "related_item_type":"twitter user",
   "rule_id":"pws worldsensing incidence test",
   "status":10,
   "sub_type":"n/a",
   "the_geom":{
        "coordinates":[
            2.1412879,
            41.380824
        ],
        "type":"Point"
   },
   "title":"Alert reported by mvila worldsensing via Twitter refering to
    ↪ 'pws worldsensing incidence test'",
   "type":110
}
```

If the rule and alert are correctly inserted, the system will reply with *201 CREATED* status. Bear in mind that, in order to prevent the PWS from receiving multiple incidents referring to the same event, repeated alerts are filtered out and their recurrence increased (See Section 4.5 for Alert model).

10.4.2 Confirm the Twitter tentative incident

An alert can be patched based on the alert's ID, which can be retrieved on alert POST (in this example, ID=39):

```
POST /api/alerts/alerts/39
```

```
{
    "operative_status":25,
}
```

Again, a successful patch will return a "200 OK" HTTP response.

10.4.3 Create a Response Plan

Similarly to how rules and alerts are created, new response plans can also be generated. In the following example, a response plan for alerts from the category 60 (Fire) is generated. The response plan has only 1 action, which is to send an email (action=10) to the developer (mvila@notreal.com) with information corresponding to the response. The field "automatic=10" indicates that the response will be executed automatically without the need of an operator doing so.

```
POST /api/response-plans/response-plans
```

```
{
    "response_plan_id": "TEST_RESPONSE_PLAN",
    "message_status": 20,
    "category": 100,
    "actions": [10],
    "parameters": ["mvila@notreal.com"],
    "format": [10],
    "description": "Send an email to the developer",
    "automatic": 10,
    "severity": 40,
    "accessibility": 10,
    "area": "Catalonia area",
    "geolocation": "000000000140000000000000004010000000000000"
}
```

Any alert in the system with type=60 being confirmed (or received as a confirmed alarm) will trigger the execution of this plan. To confirm the response plan was correctly added, return status should be *201 CREATED*.

10.4.4 Send a confirmed alarm that matches the created response plan

After an automatic response plan corresponding to a fire alert has been generated, a confirmed fire alarm can be sent in order to ensure the response plan is successfully triggered:

```
POST /api/alerts/rules
```

```
{
    "id":"fire detection sensors alarm",
    "measure_name":"Incidence",
    "measure_unit":"Incidence geotag",
    "name":"Fire detected"
}
```

After the rule has been created, an alert corresponding to this rule can be posted:

```
{
   "meta_type":"meta_data",
   "related_item_id":"Vallvidrera fire detection sensors",
   "related_item_type":"Collserola fire detection system",
   "rule_id":"fire detection sensors alarm",
   "operative_status":25,
   "sub_type":"n/a",
   "the_geom":{
          "coordinates":[
             2.094793, 41.414787
          ],
          "type":"Point"
   },
   "title":"Alert reporting by Collserola's fire detection system",
   "type":60
}
```

10.5 Guidelines

10.5.1 Software

Below are the guidelines that have been used during the project. The software guidelines, in Python, since all of the development of the backend has been done in Python. And for version control the GIT guidelines.

10.5.1.1 Coding in Python

Some of the characteristics that Worldsensing's coding in Python should follow:

- Use double quotes for strings, except when the text contains the double quote character; then use single quotes to avoid backslash
- Use PEP 8 as a style guide
- Maximum line length is 100 characters
- If more guidance is needed (on things not covered by PEP 8 or our custom recommendations), refer to Google's style guide

10.5.1.2 Microservices

Some of the characteristics that Worldsensing's microservices should follow:

- Deployed in DockerHub
- Deploy with docker-compose
- Use a makefile for building, running tests and running locally
- Pass configuration parameters through environment variables
- Document the service in the repository's README
- Have a Changelog in the repository
- All documentation should be in Markdown format
- Multiple dockerfiles for the processes of a same microservice go in the same repository
- Can be composed of one or several processes
- Clear API with other microservices
  - Preferably not the database, except when it is clearly defined as the communication between two microservices
  - The same database tables should not be updated by two microservices in any case
- Use queues for communicating microservices when possible/makes sense
- One repository per service

10.5.2 Git

- Commit messages style
  - Short first line summarizing the commit (<50 characters)
  - Add newline after first line
  - Give details on the feature afterwards (usually first line won't be enough)
  - Use the imperative of the verb
    - "Add X service" - OK
    - "Added X service" - KO
- Use ".gitignore" to prevent from tracking unnecessary files
  - Compiled objects (*/build, *.exe, *.o, *.pyc, *.javac, ...)
  - Credentials
- Don't meaningless commits
  - It works
  - Bugfixing
  - ".”