# D4.4 Implementation and Integration of Adapters

| | |
|---|---|
| Deliverable ID: | D 4.4 |
| Deliverable Title: | Implementation and Integration of Adapters |
| Revision #: | 1.0 |
| Dissemination Level: | Public |
| Responsible beneficiary: | CGI |
| Contributing beneficiaries: | All |
| Contractual date of delivery: | 31.5.2019 |
| Actual submission date: | 24.6.2019 |

# Table of Contents

# Definitions & abbreviations

**AES (Advanced Encryption Standard)** is a symmetric block cipher chosen by the U.S. government to protect classified information and is implemented in software and hardware throughout the world to encrypt sensitive data.

**Architecture** - Is both the process and the product of planning, designing, and constructing data solutions. It contains following elements: information technology specifications, models and guidelines, using a variety of Information Technology notations. Architecture of DMP is covered further in this document by describing all aspects of application Azure services and processes to fulfill project objectives.

**ERD (Entity Relationship Diagram)** An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and specifies relationships that can exist between entities.

**ETL (Extract, Transform, Load)** - Refers to a process in database usage and especially in data warehousing. Data extraction is where data is extracted from homogeneous or heterogeneous data sources; data transformation where the data is transformed for storing in the proper format or structure for the purposes of querying and analysis; data loading where the data is loaded into the final target database.

**GPS (Global positioning System)**

**GUI (Graphical User Interface)** is a form of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

**IAQ (Indoor Air Quality)**

**IoT (Internet of Things)** - Is the network of physical devices, vehicles, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to collect and exchange data.

**OAQ (Outside Air Quality)**

**OPC (Optical Particle Counter)** an instrument that detects and counts physical particles.

**REST API (Representational State Transfer Application Protocol Interface)** - Is a way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations.

**SQL (Structured Query Language)** - Domain-specific language used in programming and designed for managing data held in a relational database management system.

**TCP (Transmission Control Protocol)** – One of the protocols of the transport layer of the Internet protocol suite. TCP provides the connection between two hosts using IP packets. Provides reliable, ordered and error-free delivery of packets.

# 1 Introduction

The objective of this deliverable is to give an overview of how the adapters are implemented and integrated in the different pilots and jointly in the ESTABLISH project. The report will focus on the interoperability of the pilots and how the pilots have worked together with data, tools and platforms to create synergies between the pilots.

Interoperability in Establish has been achieved via a GUI, Sensor data validation, data sharing and EviF framework (visualization framework) between partners and pilots. Collaboration in the integration and implementation of adapters have also been done outside of the ESTABLISH project with other projects.

This document references the deliverables in work package 3, which should be referred to for more information.

# 2 Interoperability in ESTABLISH

## 2.1 ESTABLISH GUI

The present chapter refers to the part of ESTABLISH GUI related to data presentation for data received from sensors and wearable.

The whole GUI is more complex and is subject to the document D3. 2 ESTABLISH GUI.

The current presentation is strictly related to the interoperability between modules with sensors, modules with wearable, and backend analysis and presentation system. The presentation part (GUI) is the common place where all data is visible for the final user, while the interoperability procedures are collaborating to get a consistent set of data.

There are two types of data presentations performed:

- Data presentation and analysis in real time. The received data is analysed, processed by the rule engine or the automatic learning system. The results are immediately presented to the user.
- Long-term presentation and analysis. Historical data is analysed, processed by the rule engine or automatic learning system. The results are presented to the user.

The presentation is in the form of diagrams and lists. For graphs, Grafana (Grafana, 2018) and Primefaces (Primefaces, 2018)

Data presentation is different for different roles in the application.

We consider that the main roles are GUEST, PATIENT and KINETO

For all roles, the presentation is grouped in

- Main Dashboard

- Real Time evolution of data from sensors (IoT) and wearable (FitBit)

**Main Dashboard (figure 1)**

The main dashboard is accessible to all roles.

The graphical components that represent averages of measurements of some sensors are displayed in the reduced view (link *View more* is available to expand). By default, the tab *Current Data* is displayed. This tab is supposed to display real-time measurements from up to 8 environmental sensors and weather data. The displayed values are measured by sensors associated to the current location of the patient (measured by GPS tracker of the smartphone).

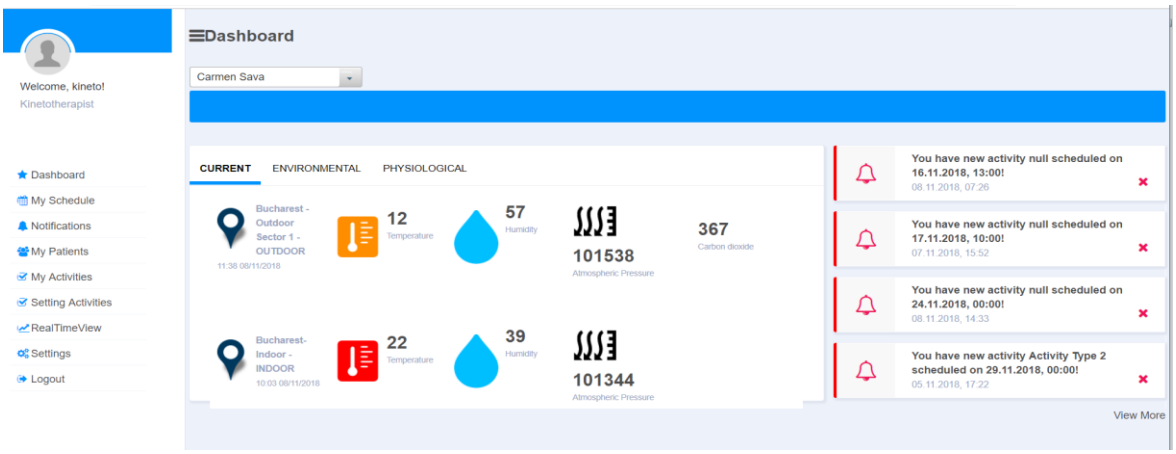


*Figure 1. Main dashboard*

**Weather conditions, exterior (figure 2)**

This part is accessible to all roles.

The graphical components that represent averages of measurements of some sensors specific for weather conditions, installed in the exterior of monitored building.

Temperature, Humidity and Pressure are registered using Libelium devices, then read from IoT MQTT broker, and stored in the ESTABLISH database.

Based on the three values, a comfort index is computed.


*Figure 2. Weather conditions*

**Air quality, exterior (figure 3)**

This data is accessible to PATIENT and KINETO roles

The graphical components that represent averages of measurements of some sensors specific for air conditions, installed in the exterior of monitored building.

Nitrogen Dioxide. Carbon Dioxide and Carbon Monoxide are registered using Libelium devices, then read from IoT MQTT broker, and stored in the ESTABLISH database.

Based on the three values, an Air Quality index is computed.


*Figure 3. Air Quality exterior*

**Comfort condition interior (figure 4)**

This data is accessible to PATIENT and KINETO roles

The graphical components that represent averages of measurements of some sensors specific for weather conditions, installed in the interior of monitored building.

Temperature, Humidity and Pressure are registered using Libelium devices, then read from IoT MQTT broker, and stored in the ESTABLISH database.

Based on the three values, a comfort index is computed.

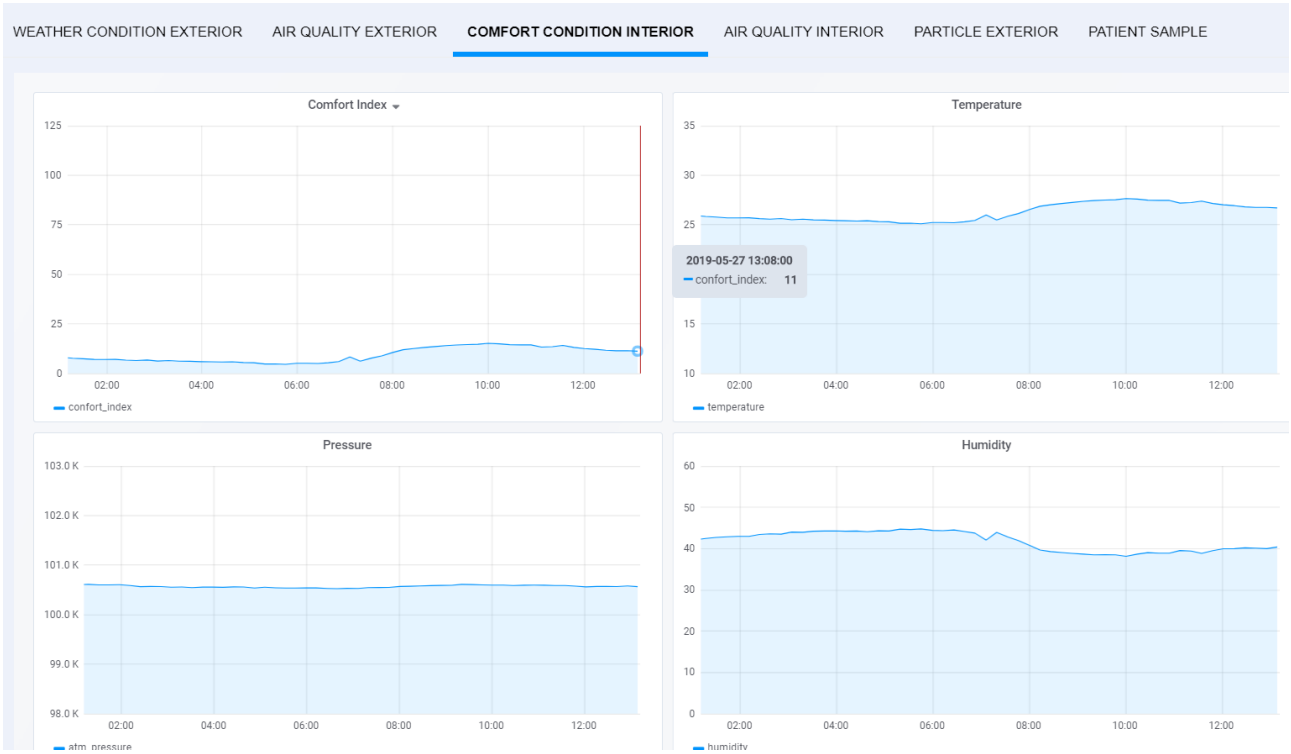


*Figure 4. Comfort condition interior*

**Air quality, interior (figure 5)**

This data is accessible to PATIENT and KINETO roles

The graphical components that represent averages of measurements of some sensors specific for air conditions, installed in the interior of monitored building.

Nitrogen Dioxide. Oxygen and Carbon Monoxide are registered using Libelium devices, then read from IoT MQTT broker, and stored in the ESTABLISH database.

Based on the three values, an Air Quality index is computed.


*Figure 5. Air quality interior*

**Particle, exterior (figure 6)**

This data is accessible to PATIENT and KINETO roles

The graphical components that represent averages of measurements of some sensors specific for particles measured in the exterior of the building.

Particle 1, 2.5 and 10 are registered using Libelium devices, then read from IoT MQTT broker, and stored in the ESTABLISH database.

Based on the three values, an Air Quality index is computed.

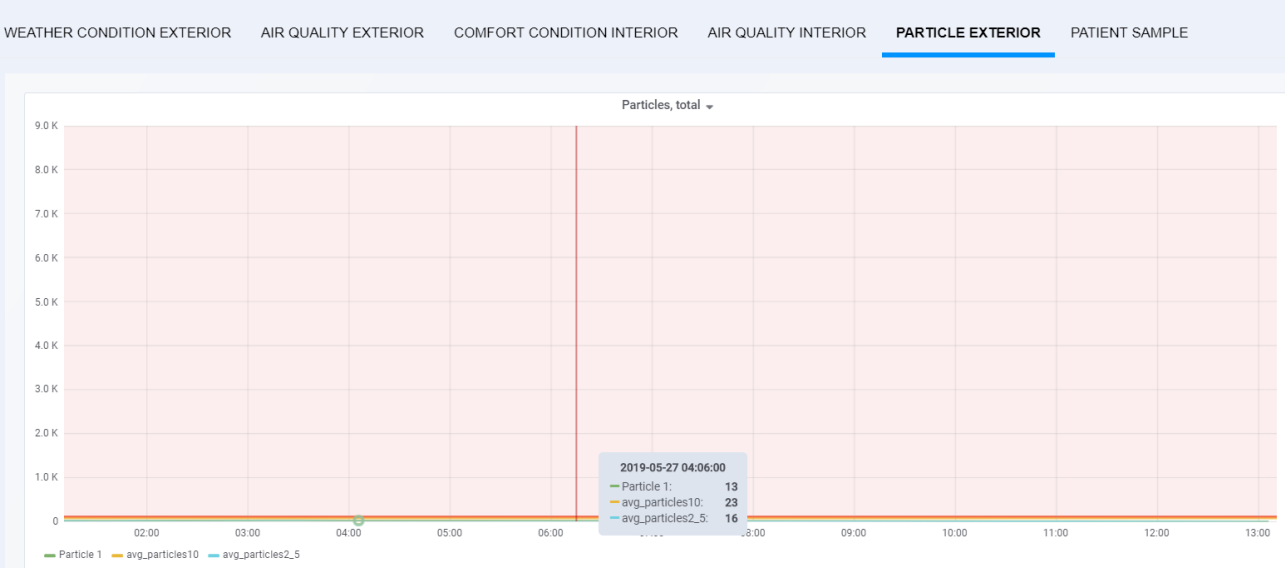


*Figure 6. Particles exterior*

## User Physiological data

This data is accessible to PATIENT and KINETO roles, with the following restrictions:

The PATIENT can access only data belonging to him

The KINETO can access only data belonging to patients assigned to him.

Physiological data is collected from the type of bracelet that the patient wears

These devices record the data according to the device type.

For ESTABLISH we are looking at devices that can record at least

- Cardiac rhythm
- Number of steps
- Sleep time

In the current implementation, we use FitBit Charge 2 devices.

Data is first recorded on the FitBit (cloud) site where it is then retrieved from the ESTABLISH kernel.

Once taken data is preprocessed, processed, stored and presented in system.

The component is interfaced with the activity management component where it provides input values for terms and conditions. It is also a data source for visualization, presentation, analysis.

Data viewing is displayed in graphs with time series or lists of values (reports)

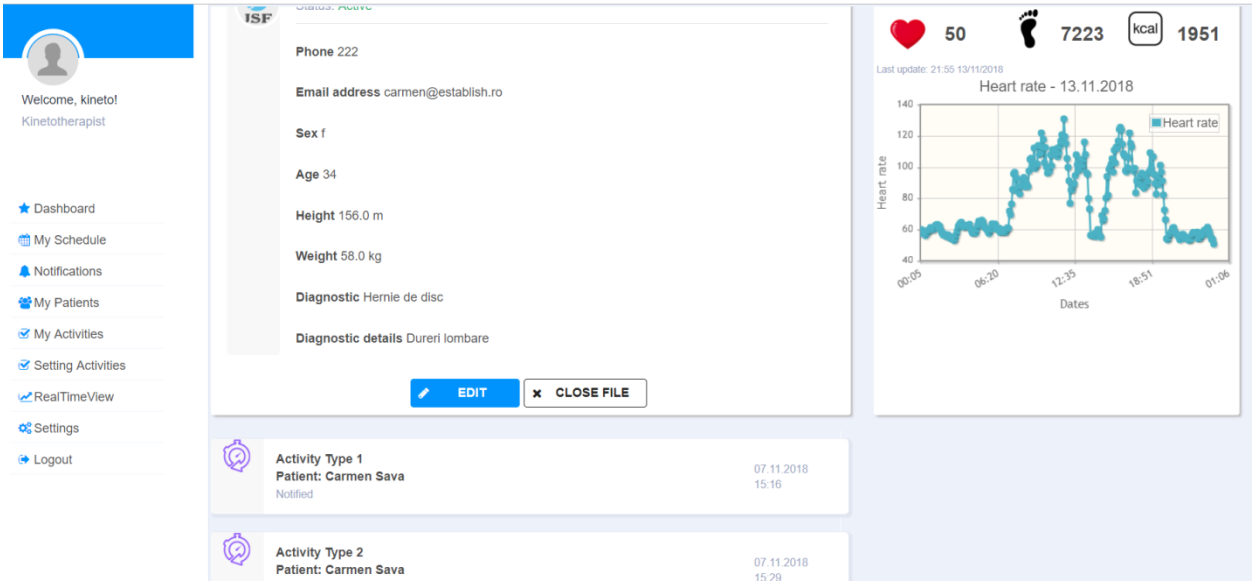


*Figure 7. User physiological data*

**User, health data analyzed (fugure 8)**

This data is accessible to PATIENT and KINETO roles, with the following restrictions:

The PATIENT can access only data belonging to him

The KINETO can access only data belonging to patients assigned to him.

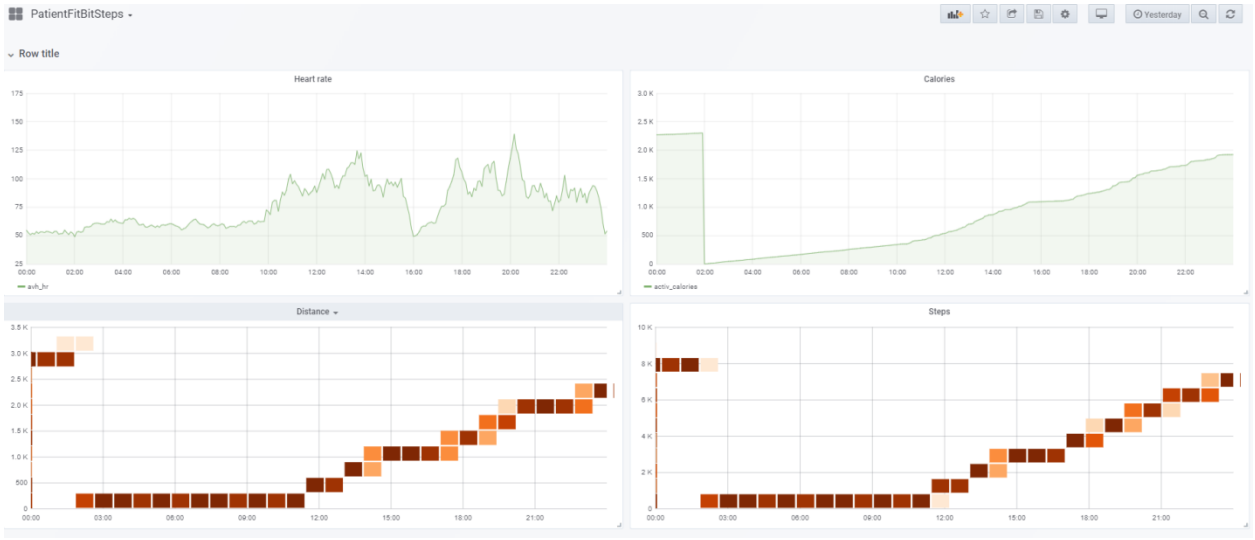


*Figure 8. User health*

## 2.2 Sensor Data Validation

The comparative analysis of measured pollutant concentrations with different sensors has been done to identify the variability of measurements between same type of equipment and between different equipment. The parameters tested are particulate matter (fractions $PM_{10}$ and $PM_{2.5}$), while the measuring techniques are represented by gravimetric and Optical Particle Counter (OPC).

Regarding the technologies that focus on PMs concentration measurement the main characteristics and recommendation are:

Gravimetric technology:

- **Advantages**: they offer very high precision; there are international standards for measurements using this technology;
- **Disadvantages**: intensive process in terms of human capital, investments (dedicated laboratory) and related costs (daily transport of the samples, materials); no real or near-real time concentration values; restriction regarding the location where can be installed; low-frequency values (24 h).
- **Recommendation**: for regulatory purposes (i.e. to demonstrate the compliance with Air Quality Directives).

OPC (Optical Particle Counter) technology:

- **Advantages**: provide real time values with very high frequency; low cost of investments and related costs; can be installed almost everywhere;
- **Disadvantages**: good or very good precision; no existing standard for measurements.

- **Recommendation**: used for the management of air pollutants sources to reduce emissions; to provide new insights relating the connection between exposure and effects of air pollutants; can be used to complement the measurements made by gravimetric network and raise awareness.

For gravimetric measurement of particle, a Bravo M Plus air pump was used to measure the $PM_{10}$ and $PM_{2.5}$ fraction of dust and atmospheric conditions (nebulosity, temperature, atmospheric pressure, humidity, speed, and wind direction). This equipment allows measurements according to the requirements of EN 12341 standard and represent the reference method for measurements used for regulatory purposes (including environmental authorities).

The equipment using OPC technique for measuring the $PM_{10}$ and $PM_{2.5}$ concentrations are Libelium Plug and Sense SCP[1](SCP1, SCP2, SCP3) and uRADMonitor Industrial[2]. The technical specification of the equipment used and the description of the scenarios are presented in Table 1 and Table 2.

*Table 1. Equipment and measurement technology*

| **Equipment** | **Technology** | **Frequency of measurement** | **Sensor** |
|---|---|---|---|
| Bravo M Plus Pump | Gravimetric | 24 hours | N/A |
| Libelium Plug and Sense SCP | OPC | User selected / 15 minutes | OPC-N2[3] |
| uRADMonitor Industrial | OPC | 1 minute | Winsen ZH03A[4] |

*Table 2. Scenario and type of equipment*

| **Scenario** | **Type of environment** | **Time period, days** | **Type of equipment** | | |
|---|---|---|---|---|---|
| | | | Bravo M Plus | uRADMonitor Industrial | Libelium Plug and Sense SCP |
| A - different technology | outdoor, industrial | 5 | Yes | Yes | Yes |
| B – same technology, different equipment | outdoor, urban | 10 | No | Yes | Yes |
| C - same technology and equipment | outdoor, urban | 10 | No | No | Yes |

### 2.2.1 Scenario A - Different technology

In order to compare the measured values with different technologies, it is necessary to consider the averaging time period of which equipment. Since gravimetric technology provide only daily values, data monitored with Libelium and uRAD were processed to obtain the same type of values (was calculated as average values of hourly average values).

The results of the monitored data processing (Figure 9) indicate that OPC equipment underestimates the values of concentrations between 18% (day 3, Libelium equipment) and 239% (day 5, uRAD equipment).

[1] Libelium, Waspmote Plug&Sense, Technical Guide, http://www.libelium.com/products/plug-sense/
[2] uRAD Monitor, Technical Documents https://www.uradmonitor.com/uradmonitor-industrial/
[3] http://www.alphasense.com/index.php/products/optical-particle-counter/
[4] https://www.winsensensor.com/d/files/PDF/Gas%20Sensor%20Module/PM2.5%20Detection%20Module/ZH03A%20Laser%20Dust%20Module%20V1.8.pdf

It should be noted that on the 5th day the concentration values were very high and could be correlated with special weather conditions (high speed wind).

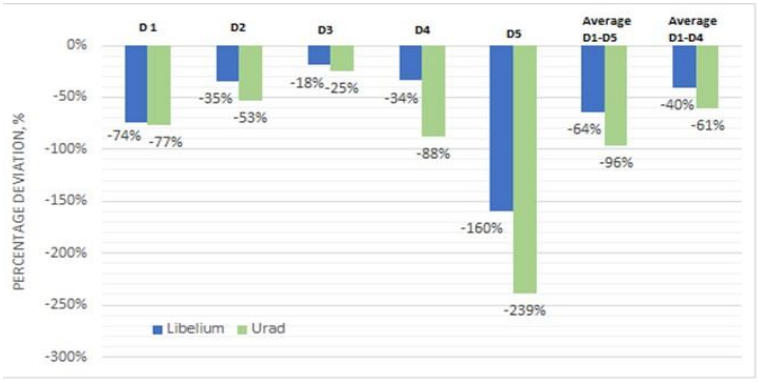


*Figure 9. Abatement percentage of OPC technology equipment compared with gravimetric equipment*

### 2.2.2 Scenario B - Same technology, different equipment

For the comparative analysis of PM concentrations, the data sets measured between 5 and 15 October 2018 were considered. From the measured data analysis of the concentrations $PM_{2.5}$ (Figure 10) and $PM_{10}$ (Figure 11), it is observed that they follow almost the same variation curve, with the concentrations measured by uRAD equipment being relatively lower than those measured by Libelium equipment (shown as mean value).

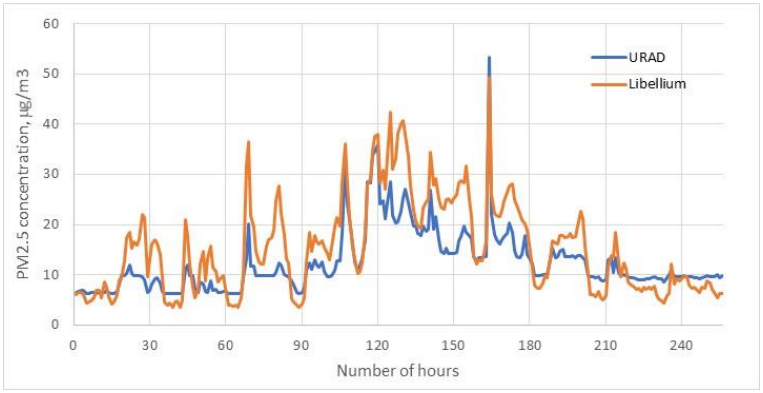


*Figure 10. $PM_{2.5}$ concentrations values registered in the monitoring period*

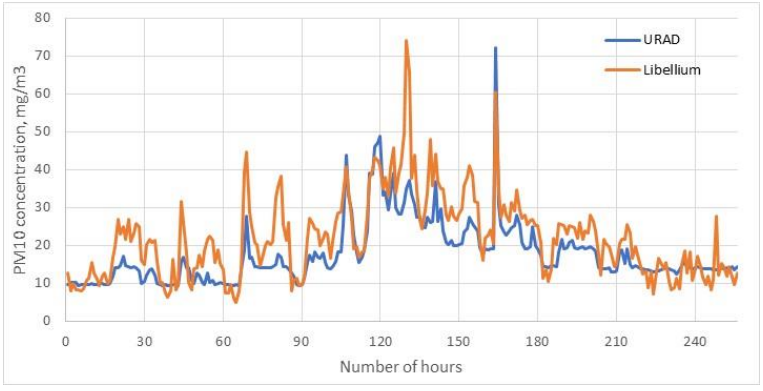


*Figure 11. $PM_{10}$ concentrations values registered in the monitoring period*

The values of the main statistical parameters (Table 3) for the uRAD data series and the mean value of the Libelium equipment are quite homogeneous (with the lower mean value for uRAD). The values of Pearson correlation coefficients are significant, with the smallest between Libelium and uRAD equipment (Table 3).

*Table 3. The values of the main statistical parameters for $PM_{2.5}$ and $PM_{10}$ measured concentrations*

| **Statistical parameter** | **$PM_{2.5}$** | | **$PM_{10}$** | |
|---|---|---|---|---|
| | uRADMonitor | Libelium | uRADMonitor | Libelium |
| Mean value | 12.43 | 15.28 | 17.68 | 21.93 |
| Median value | 9.91 | 13.18 | 14.33 | 20.99 |
| Standard deviation | 6.24 | 9.35 | 8.29 | 10.89 |
| Variation | 38.93 | 87.49 | 68.86 | 118.79 |
| Minimum value | 6.20 | 3.43 | 9.39 | 4.95 |
| Maximum value | 53.36 | 49.30 | 72.11 | 74.23 |

### 2.2.3 Scenario C - Same technology and equipment

In this scenario beside $PM_{2.5}$ and $PM_{10}$ concentrations were compared the following parameters: temperature, atmospheric pressure and relative humidity. The data obtained in the Scenario C show little variation between $PM_{2.5}$ and $PM_{10}$ concentrations (Figure 12 and 13) measured with the same type of sensors.

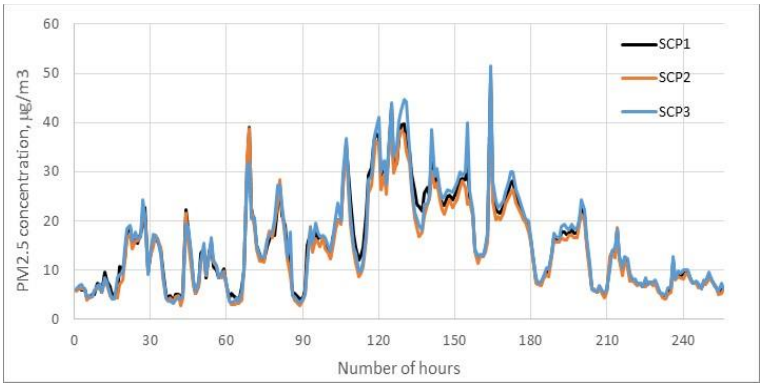


*Figure 12. PM2.5 concentrations values registered in the monitoring period*

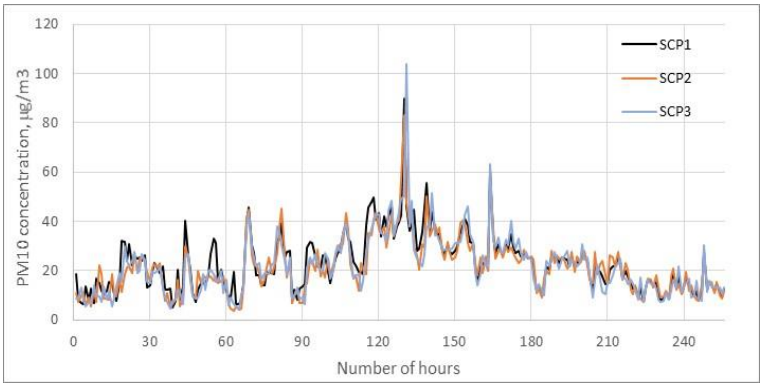


*Figure 13. $PM_{10}$ concentrations values registered in the monitoring period*

The values of Pearson correlation coefficients are significant, with the smallest between Libelium and uRAD equipment (Table 4).

*Table 4. Pearson's correlation coefficient values for $PM_{2.5}$, $PM_{10}$*

| | $PM_{2.5}$ | | | | | $PM_{10}$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | SCP1 | SCP2 | SCP3 | URAD | | SCP1 | SCP2 | SCP3 | URAD |
| SCP1 | 1 | | | | SCP1 | 1 | | | |
| SCP2 | 0.989 | 1 | | | SCP2 | 0.905 | 1 | | |
| SCP3 | 0.985 | 0.984 | 1 | | SCP4 | 0.848 | 0.850 | 1 | |
| URAD | 0.879 | 0.857 | 0.864 | 1 | URAD | 0.804 | 0.791 | 0.803 | 1 |

The ANOVA method was used to evaluate the differences between the four datasets. The value of the statistical test F ($F_{PM2.5}$ = 7.9396, $F_{PM10}$ = 10.8123) for the series analysed is higher than the critical value ($F_{crt}$ = 2.6136) which indicates that the differences between the four sets of data are statistically significant. The result of the test indicates statistically significant differences between the measurements performed with the uRAD and Libelium equipment.

From the analysis of the measured data (Figure 14), it is observed it follows the same variation curve, indicating that the systems for measuring the temperature provide similar values.

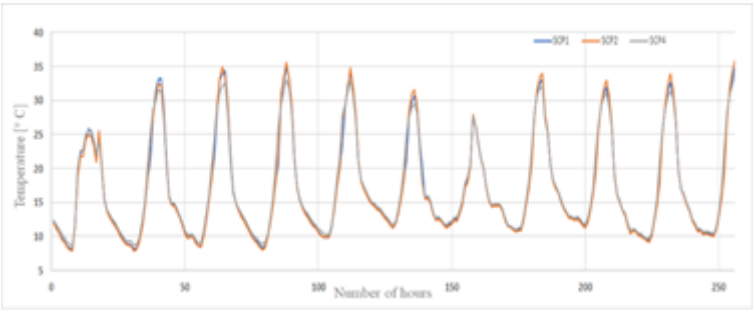


*Figure 14. The variation of temperature values during the monitoring period*

The values of the main statistical parameters (Table 5) for the three data sets are quite homogeneous, and the values of correlation coefficients are significant (Table 6).

*Table 5. The values of the main statistical parameters for temperature*

| Statistical parameter | Libelium equipment | | |
|---|---|---|---|
| | SCP1 | SCP2 | SCP4 |
| Mean value | 17.16 | 17.10 | 17.35 |
| Median value | 14.57 | 14.40 | 14.73 |
| Standard deviation | 7.45 | 7.64 | 7.11 |
| Variation | 55.52 | 58.40 | 50.68 |

| | | | |
|---|---|---|---|
| Minimum value | 8.06 | 7.83 | 8.53 |
| Maximum value | 34.94 | 35.93 | 33.96 |
| Count | 256 | 256 | 256 |
| Confidence level (95%) | ± 0.91 | ± 0.94 | ± 0.87 |

*Table 6. The values of correlation coefficients for temperature*

| | SCP1 | SCP2 | SCP4 |
|---|---|---|---|
| SCP1 | 1 | | |
| SCP2 | 0.997 | 1 | |
| SCP4 | 0.994 | 0.994 | 1 |

The ANOVA method was used to assess the differences between the three sets of data. The value of the statistical test F (F = 0.0783) for the series analysed is lower than the critical value ($F_{crt}$ = 3.0074), indicating that the differences between the three data sets are not statistically significant and the variations registered are explain by the influence of random factors.

From the analysis of the measured data (Figure 15), it is observed it follows the same variation curve, indicating that the systems for measuring the atmospheric pressure provide similar values.


*Figure 15. The variation of atmospheric pressure values during the monitoring period*

The values of the main statistical parameters (Table 7) for the three data sets are quite homogeneous, and the values of correlation coefficients are significant (Table 8).

*Table 7. The values of the main statistical parameters for atmospheric pressure*

| Statistical parameter | Libelium equipment | | |
|---|---|---|---|
| | SCP1 | SCP2 | SCP4 |
| Mean value | 101626.3 | 101567.5 | 101678.6 |
| Median value | 101712.4 | 101653.1 | 101775.5 |
| Standard deviation | 368.9 | 368.7 | 369.2 |
| Variation | 136100.7 | 135986.4 | 136350.2 |
| Minimum value | 100739.8 | 100683.2 | 100771.7 |
| Maximum value | 102321.8 | 102272.3 | 102361.5 |
| Count | 256 | 256 | 256 |
| Confidence level (95%) | ± 45.40 | ± 45.38 | ± 45.44 |

*Table 8. The values of correlation coefficients for atmospheric pressure*

| | SCP1 | SCP2 | SCP4 |
|---|---|---|---|
| SCP1 | 1 | | |
| SCP2 | 0.999 | 1 | |
| SCP4 | 0.999 | 0.998 | 1 |

The ANOVA method was used to assess the differences between the three sets of data. The value of the statistical test F (F = 5.8143) for the series analysed is higher than the critical value (Fcrt = 3.0074), indicating that the differences between the three data sets are statistically significant.

In order to identify which of the three equipment that provide mean value which is statistically significantly different from the other two, the Student test (test t) was applied. The resulting values (table 9) indicate with high probability that the mean value for the values measured by SCP1 is statistically different from those measured by the SCP2 stations, while between the values measured by the SCP1 and SCP4 stations, respectively SCP2 and SCP4 are no statistically significant differences.

*Table 9. Student test values for the data sets analysed*

| Statistical parameters | SCP1vs. SCP4 | SCP1vs. SCP2 | SCP2 vs. SCP4 |
|---|---|---|---|
| Test t value | -57.0163 | 79.5280 | -78.3969 |
| P value | $1.9 \times 10^{-147}$ | $2.6 \times 10^{-182}$ | $8.8 \times 10^{-181}$ |
| Critical value ( Test t) | 1.6508 | 1.6508 | 1.6508 |

From the analysis of the measured data (Figure 16), it is observed it follows the same variation curve, indicating that the systems for measuring the relative humidity provide similar values.

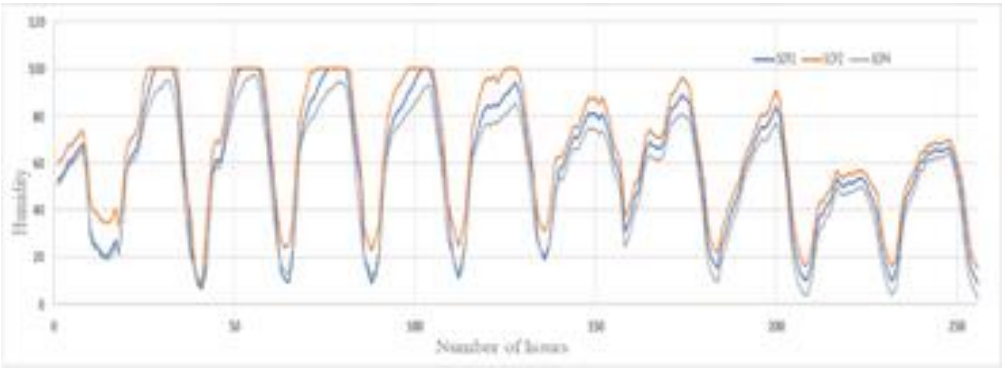

*Figure 16. The variation of relative humidity values during the monitoring period*

The values of the main statistical parameters (Table 10) for the three data sets are quite homogeneous, and the values of correlation coefficients are significant (Table 11).

*Table 10. The values of the main statistical parameters for relative humidity*

| Statistical parameter | Libelium equipment | | |
|---|---|---|---|
| | SCP1 | SCP2 | SCP4 |
| Mean value | 58.62 | 65.55 | 53.55 |
| Median value | 61.39 | 68.09 | 57.95 |
| Standard deviation | 27.21 | 25.88 | 25.66 |
| Variation | 740.85 | 669.96 | 658.54 |
| Minimum value | 6.81 | 8.50 | 2.02 |
| Maximum value | 100 | 100 | 97.65 |
| Count | 256 | 256 | 256 |
| Confidence level (95%) | ± 3.35 | ± 3.18 | ± 3.15 |

*Table 11. The values of correlation coefficients for relative humidity*

| | SCP1 | SCP2 | SCP4 |
|---|---|---|---|
| SCP1 | 1 | | |
| SCP2 | 0.999 | 1 | |
| SCP4 | 0.992 | 0.987 | 1 |

The ANOVA method was used to assess the differences between the three sets of data. The value of the statistical test F (F = 13.4485) for the series analysed is higher than the critical value (Fcrt = 3.0074), indicating that the differences between the three data sets are statistically significant. In order to identify which of the three equipment provide mean value which is statistically significantly different from the other two, the Student test (test t) was applied. The resulting values (table 12) indicate with high probability that all three mean values are statistically different from each other.

*Table 12. Student test values for the data sets analysed*

| Statistical parameters | SCP1vs. SCP2 | SCP1vs. SCP4 | SCP2 vs. SCP4 |
|---|---|---|---|
| Test t value | 28.37 | 22.36 | 47.31 |
| P value | $3.6 \times 10^{-81}$ | $2.3 \times 10^{-62}$ | $1.4 \times 10^{-128}$ |
| Critical value ( Test t) | 1.6508 | 1.6508 | 1.6508 |

### 2.2.4 Conclusion

The comparison of the measured values of $PM_{10}$ concentrations measured with equipment with different technologies (gravimetric and laser equipment) indicated that both laser equipment underestimated the concentration values. This is due to the lower reading frequency (15 minutes) for Libelium and for the fact that laser technology does not allow the elimination of the influence of humidity on the particle size measured.

Conversion factors between $PM_{10}$ concentrations measured with gravimetric equipment and those measured with laser technology are 1.64 for Libelium and 1.96 for uRAD. For Libelium equipment it is possible to improve this factor by increasing the measurement frequency.

The results of the comparative analysis between the measured values of the same type of equipment (Libelium) for $PM_{10}$ and $PM_{2.5}$ concentrations indicate a high degree of homogeneity and reproducibility of the measurements made (differences between data sets are not statistic significant). Concerning atmospheric pressure and relative humidity parameters, the results of statistical tests (ANOVA analysis) show statistically significant differences.

The results of the comparative analysis between $PM_{10}$ and $PM_{2.5}$ concentrations measured with different types of equipment (Libelium and uRAD) with the same type of technology (laser) indicate that there are statistic significant differences between them, although the coefficients correlations are significant. For all parameters included in the analysis, the values measured with uRAD are lower than those measured with Libelium.

## 2.3 Data Sharing

The LoRaWAN networks laid out in a star-of-stars topology have base stations relaying the data between the sensor nodes and the network server.

Communication between the sensor nodes and the base stations goes over the wireless channel utilizing the LoRa physical layer, whilst the connection between the gateways and the central server are handled over a backbone IP-based network.

End Nodes transmit directly to all gateways within range, using LoRa.

Gateways relay messages between end-devices and a central network server using IP.

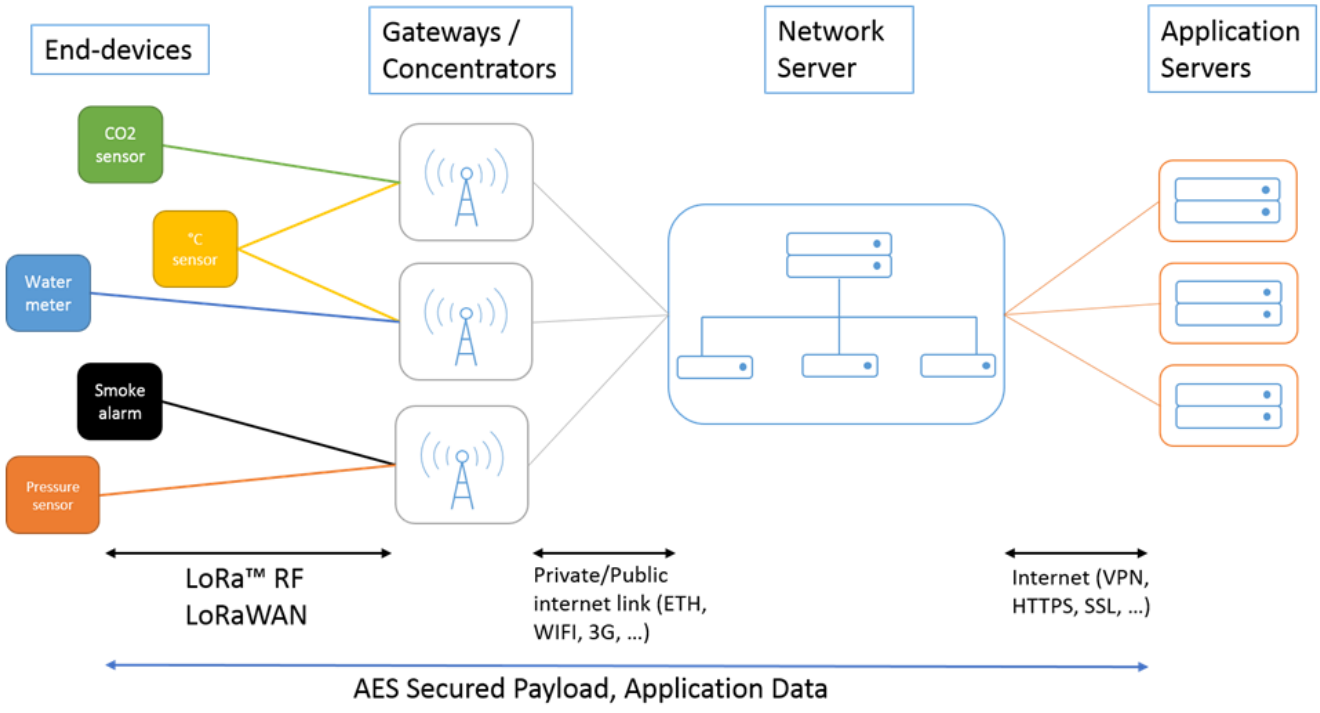


*Figure 17. Architecture for Data Sharing*

**End-devices**

The end devices are LoRa embedded sensors (used to detect the changing parameter eg. temperature, humidity, accelerometer, gps).

The sensors may connect to the LoRa transponder chip, or the sensor may be an integrated unit with the LoRa transponder chip embedded.

The LoRaWAN sensors typically use Low Power and are battery powered (Class A and Class B). LoRa embedded sensors that run on batteries that can typically last from 2-5 years. The LoRa sensors can transmit signals over distances from 1km - 10km.

**Gateways**

The LoRa sensors transmit data to the LoRa gateways. The LoRa gateways connect to the internet via the standard IP protocol and transmit the data received from the LoRa embedded sensors to the Internet i.e. a network, server or cloud.

The Gateways devices are always connected to a power source. The Gateways connect to the network server via standard IP connections and act as a transparent bridge, simply converting RF packets to IP packets and vice versa.

**Network Servers**

The Network servers can be cloud based platform solutions like Ceske Radiokomunikace company (CRA). or LoRIOT. The network servers connect to the gateways and de-dup data packets, and then routes it to the relevant application. The network servers can be used for both uplink (i.e. sensor to application) or downlink (i.e. application to sensor) communication.

**Application Server**

IoT LoRaWAN web application server to monitor the state of the environment using a LoRa wireless network.

**Data sharing format**

We use the JSON format for data sharing. JSON (JavaScript Object Notation) is a lightweight format for sharing data. Although it's derived from JavaScript—it may be used with many programming languages. You don't need to invent your own file format. You can use standardized infrastructure (serializers, libraries).

First of all, we need to login to get an authorization token.
**POST** https://iotlorawan.azurewebsites.net/api/accounts/login
**Headers**:
**Email**: john.doe@ima.cz
**Password**: johndoe1

**Status: 200 OK**

{
"accessToken": "HFav1KKmrjoo3Pitii20tvqZMPErF8HNYrF2JJMvz0JKq5ioH8CZ9EM9faleuznx9sRw8a",
"expires": "2020-05-30T23:59:59+00:00"
}

If the user does not exist, the wrong password or email, returns
**Status**: **401 Unauthorized**
{

" Message ": "Login failed. Incorrect email address or password!"
}

Then we can get information about of established and active sensors

**GET** https://iotlorawan.azurewebsites.net/api/sensors
**Headers:**
**Authorization:** "HFav1KKmrjoo3Pitii20tvqZMPErF8HNYrF2JJMvz0JKq5ioH8CZ9EM9faleuznx9sRw8a"

**Status: 200 OK**

[
  {
    "eui": "0E7E3464333100B6",
    "acronym": "ASC-B6",
    "name": "Ascoel CO868LR",
    "description": "Ascoel CO2 for Establish",
    ...
     "status": 1
  },
  {
    "eui": "8CF9574000000231",
    "acronym": "RHF231",
    "name": " Rising RHF1S001",
    "description": " Rising TH for Establish",
    ...
    "status": 1
  }
]

- The authorization token is not valid at the current time, returns
**Status: 401 Unauthorized**

- No data found to display, returns
**Status**: **204 No Content**

Then we can get the decoded data from the sensor without any aggregations.

**GET** https://iotlorawan.azurewebsites.net/api/sensordata/?eui=0E7E3464333100B6&from=2019-05-30T00:00:00&limit=100

**Parameters:**

**eui –** The unique identifier for the device in this app.

**from –** Returns data, their date and time greater than “from”

**limit -** The number of measurements returned (for example, the limit = 100 returns only the last 100 measurements that are greater than “from”)

**Headers:**
**Authorization:** "HFav1KKmrjoo3Pitii20tvqZMPErF8HNYrF2JJMvz0JKq5ioH8CZ9EM9faleuznx9sRw8a"

**Status: 200 OK**

{
  "eui": "0E7E3464333100B6",

```
  "name": "Ascoel CO868LR",
  "data": [
    {
      "time": "2019-05-30T13:39:15+02:00",
      "temperature": 23.4,
      "humidity": 44.06,
      "co2": 400,
      "rssi": -122,
      "snr": -17.5,
      "batteryLevel": 3.6
    },
    {
      "time": "2019-05-30T14:10:12+02:00",
      "temperature": 22.5,
      "humidity": 48.27,
      "co2": 402,
      "rssi": -120,
      "snr": -20.2,
      "batteryLevel": 3.6
    }

   ...
  ]
}
```


- The authorization token is not valid at the current time, returns
**Status: 401 Unauthorized**

- No data found to display, returns
**Status**: **204 No Content**

The data sharing refers to:

- Data sharing between ESTABLISH modules. It considers data from IoT (sensors), wearable (bracelets), input text data, analysis output
- Data presentation to external systems by the help of API

The Data Management process specific to the Romanian use case consists in:

- The sensors nodes transmit the sensors measurements through 4G or WiFi connectivity to the Meshlium Gateway. Once in Meshlium, data is stored in a MySQL or Postgresql database that ensures local persistence of the sensors data. Regarding the Romanian platform, data forwarding to Cloud is done through a software component that serializes data to an MQTT broker. Similarly, uRADMonitor connects (via Wi-Fi) to a Cloud platform that allows data to be accessed via a REST interface.
    - The integration with the common data platform can be achieved with the Azure Iot Hub service. This process can be performed by following the next steps:
    - Registering the Meshlium in Azure Portal, by annotating the connection string generated in Azure Iot Hub in the Meshlium configuration; basically, the obtained “connection string” from the Azure portal will certificate the Meshlium device as a valid sender of messages.

- After validating the Meshlium Gateway, there are required a few configurations in the Configuration panel:
- Number Requests: Number of requests to send per iteration.
- Sync Interval: Time duration in seconds between synchronizing data batches.
- Protocol: Choose the protocol to communicate with Azure IoT Hub. Valid protocols are MQTT (by default), AMQPS and HTTPS.
- Log Level: Generate log messages. From fewer to more details, the levels are OFF, ERROR, INFO, DEBUG, REPORT. The default is OFF.

### 2.3.1 IoT Gateway Logic

Part of the analytics and logic can be shifted from the knowledge layer onto the gateway layer. In this type of computing, there's a processing unit that the data passes through on the way from the sensors to the database. The processing unit can then for example filter useless or noisy information out from the data using various algorithms. This processing can reduce the size of transfers that are made between the sensors and the gateway.

In the Finnish Indoor Air Quality Improvement at School pilot, Azure Stream Analytics is a part of the gateway logic as the data is pre-processed before it is stored for the first time in the database. This way of handling data management is becoming more and more popular due to advancements in hardware technologies.

As for the Romanian Use Case, a Java bases developed module will handle the Gateway Logic tasks; namely pre-process the sets of data in order to advance to the next phase - data visualization API

APIs are exposed to allow external users to connect to the system and obtain important information about decision support. APIs are RESTFUL services. Below is the presentation of the main API classes exposed.

The whole description and documentation of APIs is part of deliverable: D3.1 Architectural and design specification


*Figure 18. API-Service RS*

### 2.3.2 Data Structure

In the figures bellow we are presented only the structures specific to data captured from sensors or wearable.

In the first structure is a detailed view of these structures.

The next figure (19) emphasizes (in red) the place of this data in the whole ERD of the application.

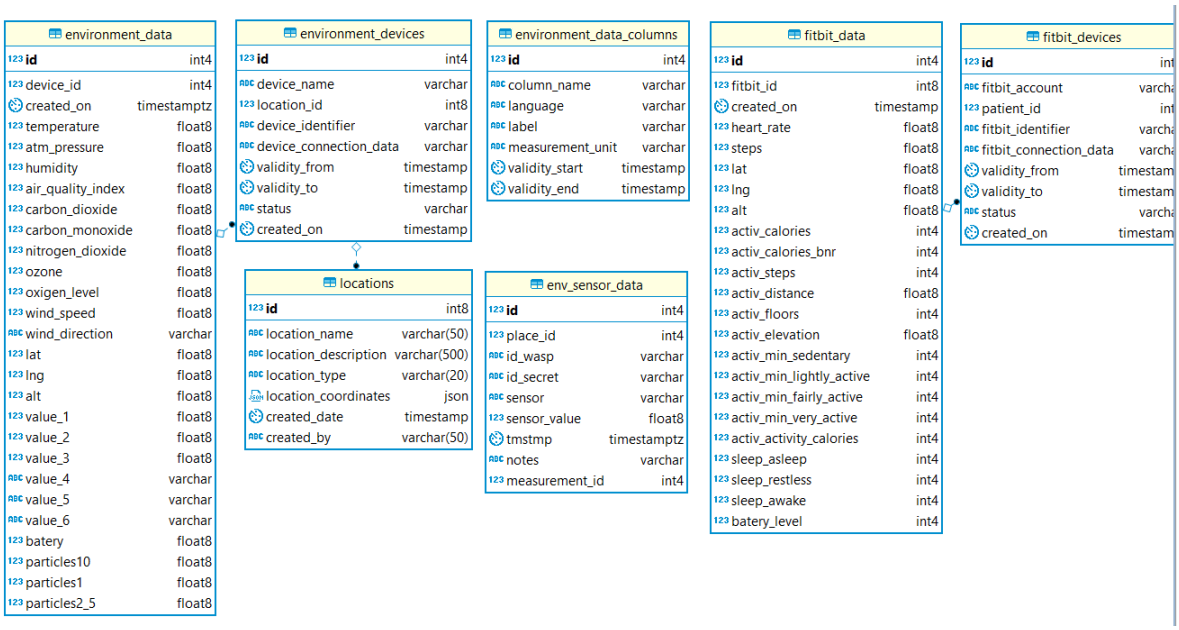


*Figure 19. Detailed data structures*

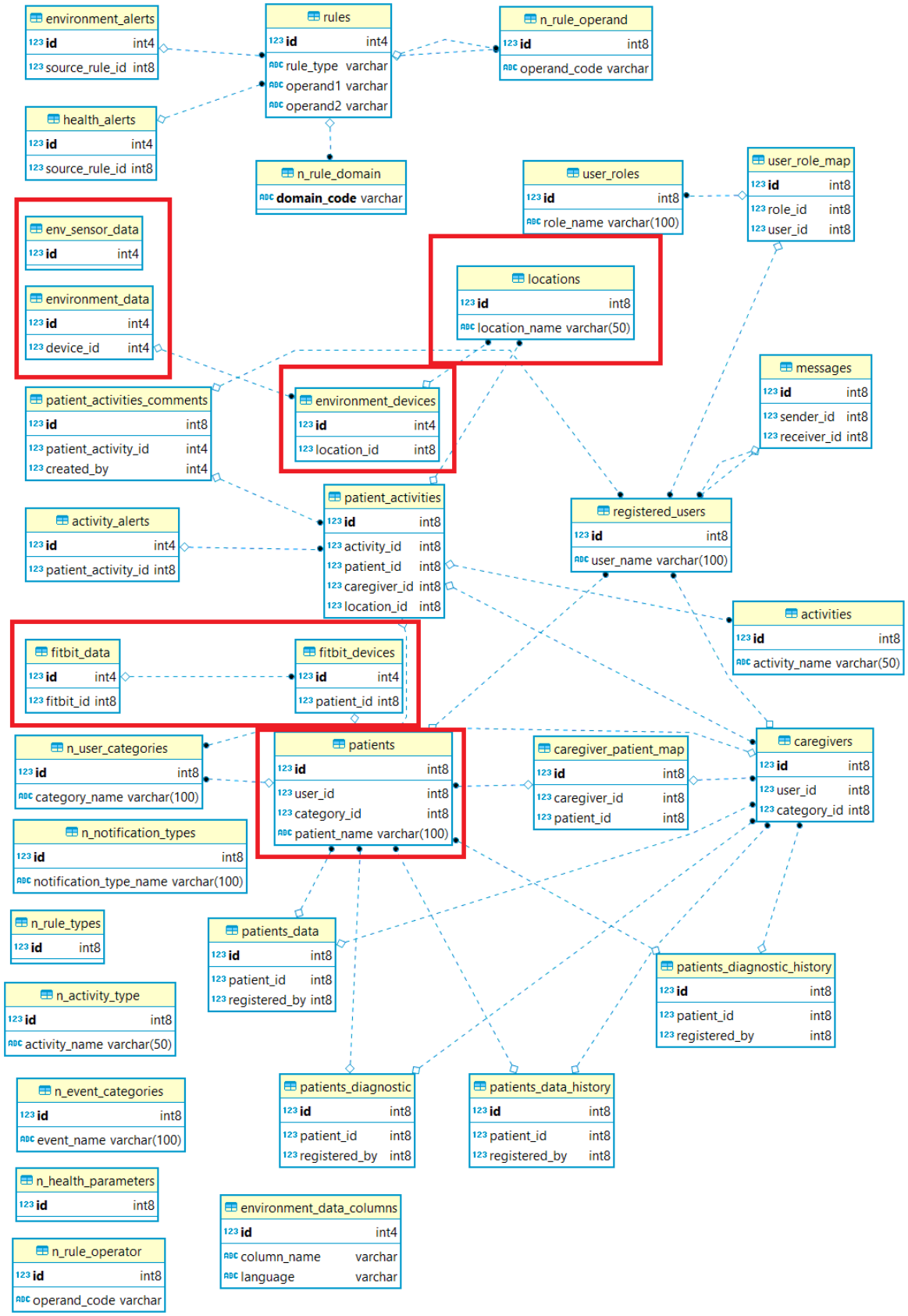


*Figure 20. Main ERD*

### 2.3.3 IAQ Data Gateway for Sharing Indoor Air Quality Data

The IAQ devices include WiFi communication to transmit their measured sensor data to the data gateway. The data gateway collects data from the IAQ devices installed in an indoor place via Transmission Control Protocol (TCP). It also sends the collected data to other target servers via TCP with Json text format or writes the data into the local files. The following figure shows the overall system architecture of the data gateway.

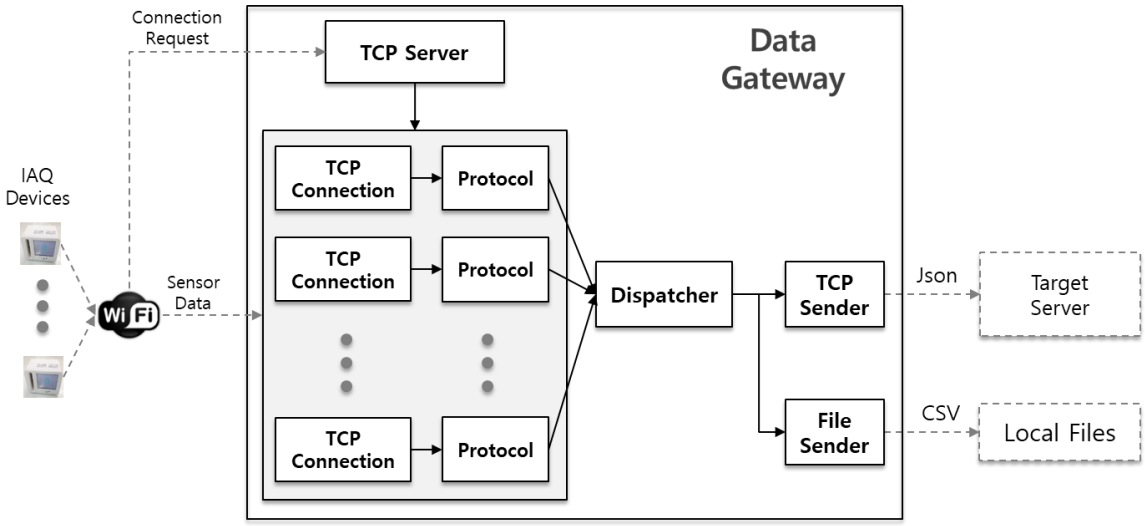


*Figure 21. Data Gateway Architecture*

1) System configuration

The system configuration file contains all the system configuration information of the data gateway. Some portion of the configuration should be changed for the execution of the data gateway. The followings describe the configuration parts that should be changed. You don't have to change the other parts of the configuration.

- TCP server setting

The IAQ devices send sensor data to the data gateway via TCP. The date gateway should prepare TCP server to receive sensor data from several IAQ devices. The following shows an example of the ip and port setting of the TCP server.

```
<networks>
    <network>
        <id>CowayServer</id>
        <io>tcp.server</io>
        <ip>192.268.1.10</ip>   <!-- ip address of the TCP server -->
        <port>port</port>        <!-- port number of the TCP server -->
        <protocol>CowayProtocol</protocol>
    </network>
</networks>
```

- Sender setting

The dispatcher sends sensor data via TCP to other target servers or saves to local files.

The setting about a target server for sending data via TCP is as follows.

```
<sender>
  <type>tcp</type>          <!-- type of sender should be “tcp” -->
  <ip>192.168.1.100</ip> <!-- ip address of the target server -->
  <port>8140</port>         <!-- port number -->
  <format>json</format>  <!-- data format : “json” -->
</sender>
```

The setting for saving data to local files is as follows.

```
<sender>
  <type>file</type>         <!-- type of sender should be “file” -->
  <unit>day</unit>          <!-- new file creation period: day or hour -->
  <path>../data</path>    <!-- directory in which files are created -->
  <format>csv</format>  <!-- data format : “csv” -->
</sender>
```

In case of saving sensor data to local files, a new file is created by daily or hourly depending on the value of “unit” tag. Therefore, a new file is created per each day or each hour when the value of the unit tag is “day” or “hour”, repectively.

2) Data format

Each IAQ sensor device for indoor air quality measurements includes temperature, humidity, CO2, illuminance, noise, VOC (volatile organic compounds), Formaldehyde and PM10, PM2.5 sensors. The following figure shows indoor the specification of the air quality sensors:

| Type | $CO_2$ | MIC | IR LED | **Illuminance** | VOC | **Formaldehyde** | PM10 / PM2.5 | Temperature / Humidity |
|---|---|---|---|---|---|---|---|---|
| Product Picture |  |  |  |  |  |  |  |  |
| Measurement Range | 0~2000 Ppm | 0~100dB | 940nm | 10~1000 lux | 1~10ppm | 0~500ppb | 0~500ug/$m^3$ | -40~125℃ 0~100%RH |
| Accuracy | <±50ppm +2% | | | | Coway's Algorithm | <±30%@Full Range | ±15% | ±0.3℃ ±2% |
| Interface | I2C | Analogue | Digital Output | Analogue | Analogue | I2C | UART | I2C |

*Figure 22. IAQ Sensors*

The dispatcher sends sensor data via TCP with Json format to other target servers or saves to local files with CSV format. The following shows an example of the Json transferred via TCP.

```
{
  "id":"EUREKA_IAQ_0000003",
```

```
"time":"2019-04-02 15:47:14",
"sensors":[
  {
    "name":"PM25",
    "unit":"ug/m3",
    "value":"7",
    "id":1
  },
  {
    "name":"PM10",
    "unit":"ug/m3",
    "value":"8",
    "id":2
  },
 {
    "name":"VOC",
    "unit":"V",
    "value":"0.19",
    "id":4
  },
  {
    "name":"Humidity",
    "unit":"RH",
    "value":"22",
    "id":5
  },
  {
    "name":"Temperature",
    "unit":"C",
    "value":"27.0",
    "id":6
  },
  {
    "name":"Light",
    "unit":"lux",
    "value":"500",
    "id":7
  },
  {
    "name":"CO2",
    "unit":"ppm",
    "value":"747",
    "id":9
  },
  {
    "name":"Formaldehyde",
    "unit":"ppb",
    "value":"0",
    "id":11
```

```
    },
    {
      "name":"Noise",
      "unit":"dB",
      "value":"50",
      "id":25
    },
    {
      "name":"BloothSpeaker",
      "unit":"V",
      "value":"0",
      "id":39
    }
  ]
}
```


When a json text is sent, it is contained in one line, that is, there is no new line in it except the end of it. Therefore, the target servers should read line by line from TCP connection in which one line contains one json text. The above json example is reformatted for making it more readable.

The meanings of tags in the json are as follows

- id: the unique device identifier of the corresponding installed IAQ device
- time: the sensor data creation time
- sensors: list of the sensor data in the IAQ device
  . name: the name of the sensor
  . unit: the unit of the sensor data unit
  . value: the value of the sensor data
  . id: the identifier of the sensor

The following shows a csv example stored to local files.


```
time,PM25,PM10,VOC,Humidity,Temperature,Light,CO2,Formaldehyde,Noise,BloothSpeaker
2019-04-02 15:52:14,7,7,0.19,22,27.0,500,839,0,50,0
2019-04-02 15:53:15,8,8,0.25,22,27.0,500,829,0,50,0
```


The first line of the csv file is the header of the csv that includes the name of each sensor. Each line except the first line includes the value of each sensor data separated by comma with the sensor data creation time in the first column.

### 2.3.4 Access to Indoor Air Quality Data

Indoor air quality data from VTT offices consists of temperature, humidity, air pressure, $CO_2$, presence (PIR) information and door open/closed latch information. The data is reported every 15 minutes per sensor node. Each node contains several sensors, and each office room has several sensors. Access to data from each room is via a room-specific URL. Data is by default available as XML. JSON is available by adding an '`accept:application/json`' header to the request.
The base URL for room 1 is as follows:

```
https://vtttepipilot.table.core.windows.net/VttNodeIAQData101?st=2018-12-31T22:01:00Z&se=2019-12-31T20:55:00Z&sp=r&sv=2018-03-28&tn=vttnodeiaqdata101&sig=YteQb2zcnGeZNu8KBlvKoL9ujnW5kUMyn6vfzDjtP7c=
```

This URL will return all data for the room. All above query parameters are mandatory. To limit the data, filters can be applied with a `$filter` parameter. The parameter can contain several limitations, e.g.,

```
$filter=PartitionKey eq '20190114' and nodeId eq 597 and co2 ge 530
```

will return data from the specified day and sensor (node) where CO2 levels are above 530ppm. Specifying a partition key or partition key range is recommended to reduce the query result set.

Additionally, a `$select` parameter can be added to limit the returned fields, e.g. `$select=co2,humidity`.

Data for another room is available via the base url

```
https://vtttepipilot.table.core.windows.net/VttNodeIAQData104?st=2018-12-31T22:01:00Z&se=2019-12-31T20:59:00Z&sp=r&sv=2018-03-28&tn=vttnodeiaqdata104&sig=ugP3+oS4ITJxzR/5KQdgUDbAABpQ+8bOcy6UpOejzoM=
```

The resulting data format is as follows (JSON). Note that not all sensors will return all fields.


```
{
   "odata.metadata": "https://vtttepipilot.table.core.windows.net/$metadata#VttNodeIAQData101",
   "value": [
      {
         "odata.etag": "W/\"datetime'2019-01-04T12%3A16%3A27.4264943Z'\"",
         "PartitionKey": "20190104",
         "RowKey": "1546604174",
         "Timestamp": "2019-01-04T12:16:27.4264943Z",
         "batteryVoltage": 3.25,
         "deviceid": "tepi_vtt_gw_101",
         "devicetimestamp": 1546604174,
         "humidity": 14.86,
         "nodeId": 610,
         "pir_cnt": 0,
         "pressure": 1007.8,
         "rssi": 76.9,
         "temperature": 21.43,
         "state": 0,
         "state_cnt": 881,
         "co2": 567
      {
         …
      },
   ]
}
```

The same result in XML:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xml:base="https://vtttepipilot.table.core.windows.net/" xmlns="http://www.w3.org/2005/Atom"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:georss="http://www.georss.org/georss" xmlns:gml="http://www.opengis.net/gml">
   <id>https://vtttepipilot.table.core.windows.net/VttNodeIAQData101</id>
   <title type="text">VttNodeIAQData101</title>
   <updated>2019-06-06T11:47:23Z</updated>
   <link rel="self" title="VttNodeIAQData101" href="VttNodeIAQData101" />
   <entry m:etag="W/&amp;quot;datetime'2019-01-04T12%3A16%3A27.4264943Z'&amp;quot;">

<id>https://vtttepipilot.table.core.windows.net/VttNodeIAQData101(PartitionKey='20190104',RowKey='1
546604174')</id>
      <category term="vtttepipilot.VttNodeIAQData101"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
      <link rel="edit" title="VttNodeIAQData101"
href="VttNodeIAQData101(PartitionKey='20190104',RowKey='1546604174')" />
      <title />
      <updated>2019-06-06T11:47:23Z</updated>
      <author>
         <name />
      </author>
      <content type="application/xml">
         <m:properties>
            <d:PartitionKey>20190104</d:PartitionKey>
            <d:RowKey>1546604174</d:RowKey>
            <d:Timestamp m:type="Edm.DateTime">2019-01-04T12:16:27.4264943Z</d:Timestamp>
            <d:batteryVoltage m:type="Edm.Double">3.25</d:batteryVoltage>
            <d:deviceid>tepi_vtt_gw_101</d:deviceid>
            <d:devicetimestamp m:type="Edm.Int32">1546604174</d:devicetimestamp>
            <d:humidity m:type="Edm.Double">14.86</d:humidity>
            <d:nodeId m:type="Edm.Int32">610</d:nodeId>
            <d:pir_cnt m:type="Edm.Int32">0</d:pir_cnt>
            <d:pressure m:type="Edm.Double">1007.8</d:pressure>
            <d:rssi m:type="Edm.Double">76.9</d:rssi>
            <d:temperature m:type="Edm.Double">21.43</d:temperature>
            <d:state m:type="Edm.Int32">0</d:state>
            <d:state_cnt m:type="Edm.Int32">881</d:state_cnt>
            <d:co2 m:type="Edm.Int32">567</d:co2>
         </m:properties>
      </content>
   </entry>
…
</feed>
```

### 2.3.5 Data exposed by Establish Coordinator

The Establish coordinator is exposing data via API (REST Services)
Data is exposed in XML or JSON format (as the clients wants)
The data structures presented below are using XML format and XML schema for their description

Access to data is via rest services
The generic URL is:
**Error! Hyperlink reference not valid.**
where {dataRoot} depends on the type of data requested, and is fully documented in the description of services.

For visualization purposes, the structures widely used are:
**fitbitData (with reference to fitbitDevices) and URL**
http://{{server}}:{{port}}/EstablishCoordinator/webresources/ro.establishrs.patientsdata/
**environment (with reference to locations and environmentDevices) and URL:**
http://{{server}}:{{port}}/EstablishCoordinator/webresources/ro.establishrs.envsensordata/
**patientsDataHistory and URL:**
http://{{server}}:{{port}}/EstablishCoordinator/webresources/ro.establishrs.patientsdatahistory

The other structures give information about patients, caregivers, rules, etc.

Bellow is a description of all structures exposed.

**fitbitData** is a structure where data registered from a wearable is placed:

```
<xs:complexType name="fitbitData">
<xs:sequence>
<xs:element minOccurs="0" name="activActivityCalories" type="xs:int"/>
<xs:element minOccurs="0" name="activCalories" type="xs:int"/>
<xs:element minOccurs="0" name="activCaloriesBnr" type="xs:int"/>
<xs:element minOccurs="0" name="activDistance" type="xs:double"/>
<xs:element minOccurs="0" name="activElevation" type="xs:double"/>
<xs:element minOccurs="0" name="activFloors" type="xs:int"/>
<xs:element minOccurs="0" name="activMinFairlyActive" type="xs:int"/>
<xs:element minOccurs="0" name="activMinLightlyActive" type="xs:int"/>
<xs:element minOccurs="0" name="activMinSedentary" type="xs:int"/>
<xs:element minOccurs="0" name="activMinVeryActive" type="xs:int"/>
<xs:element minOccurs="0" name="activSteps" type="xs:int"/>
<xs:element minOccurs="0" name="alt" type="xs:double"/>
<xs:element minOccurs="0" name="bateryLevel" type="xs:int"/>
<xs:element minOccurs="0" name="createdOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="fitbitId" type="fitbitDevices"/>
<xs:element minOccurs="0" name="heartRate" type="xs:double"/>
<xs:element minOccurs="0" name="id" type="xs:int"/>
<xs:element minOccurs="0" name="lat" type="xs:double"/>
```

<xs:element minOccurs="0" name="lng" type="xs:double"/>
<xs:element minOccurs="0" name="sleepAsleep" type="xs:int"/>
<xs:element minOccurs="0" name="sleepAwake" type="xs:int"/>
<xs:element minOccurs="0" name="sleepRestless" type="xs:int"/>
<xs:element minOccurs="0" name="steps" type="xs:double"/>
</xs:sequence>
</xs:complexType>

**fitbitDevices** is a structure used to describe the devices (wearable)

<xs:complexType name="fitbitDevices">
<xs:sequence>
<xs:element minOccurs="0" name="createdOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="fitbitAccount" type="xs:string"/>
<xs:element minOccurs="0" name="fitbitConnectionData" type="xs:string"/>
<xs:element minOccurs="0" name="fitbitIdentifier" type="xs:string"/>
<xs:element minOccurs="0" name="id" type="xs:int"/>
<xs:element minOccurs="0" name="patientId" type="patients"/>
<xs:element minOccurs="0" name="status" type="xs:string"/>
<xs:element minOccurs="0" name="validityFrom" type="xs:dateTime"/>
<xs:element minOccurs="0" name="validityTo" type="xs:dateTime"/>
</xs:sequence>
</xs:complexType>

**patients** is a structure used to describe patients

<xs:complexType name="patients">
<xs:sequence>
<xs:element minOccurs="0" name="batery" type="xs:string"/>
<xs:element minOccurs="0" name="categoryId" type="nUserCategories"/>
<xs:element minOccurs="0" name="cel" type="xs:string"/>
<xs:element minOccurs="0" name="city" type="xs:string"/>
<xs:element minOccurs="0" name="dob" type="xs:dateTime"/>
<xs:element minOccurs="0" name="email" type="xs:string"/>
<xs:element minOccurs="0" name="emergencynr" type="xs:string"/>
<xs:element minOccurs="0" name="firstName" type="xs:string"/>
<xs:element minOccurs="0" name="fullAddress" type="xs:string"/>
<xs:element minOccurs="0" name="gender" type="xs:string"/>
<xs:element minOccurs="0" name="id" type="xs:long"/>
<xs:element minOccurs="0" name="idValue" type="xs:string"/>
<xs:element minOccurs="0" name="LPict" type="xs:string"/>
<xs:element minOccurs="0" name="lastName" type="xs:string"/>
<xs:element minOccurs="0" name="lastUpdatedOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="lotnr" type="xs:string"/>
<xs:element minOccurs="0" name="MPict" type="xs:string"/>
<xs:element minOccurs="0" name="macAddress" type="xs:string"/>
<xs:element minOccurs="0" name="mapLat" type="xs:decimal"/>
<xs:element minOccurs="0" name="mapLng" type="xs:decimal"/>
<xs:element minOccurs="0" name="nat" type="xs:string"/>

<xs:element minOccurs="0" name="patientName" type="xs:string"/>
<xs:element minOccurs="0" name="patientType" type="xs:string"/>
<xs:element minOccurs="0" name="patientscol" type="xs:string"/>
<xs:element minOccurs="0" name="phone" type="xs:string"/>
<xs:element minOccurs="0" name="registrationDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="removalDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="SPict" type="xs:string"/>
<xs:element minOccurs="0" name="state" type="xs:string"/>
<xs:element minOccurs="0" name="status" type="xs:string"/>
<xs:element minOccurs="0" name="title" type="xs:string"/>
<xs:element minOccurs="0" name="unit" type="xs:string"/>
<xs:element minOccurs="0" name="userId" type="registeredUsers"/>
</xs:sequence>
</xs:complexType>

**registeredUser** structure represents data from the profiles of the users

<xs:complexType name="registeredUsers">
<xs:sequence>
<xs:element minOccurs="0" name="accessArea" type="xs:string"/>
<xs:element minOccurs="0" name="active" type="xs:boolean"/>
<xs:element minOccurs="0" name="email" type="xs:string"/>
<xs:element minOccurs="0" name="enabled" type="xs:boolean"/>
<xs:element minOccurs="0" name="id" type="xs:long"/>
<xs:element minOccurs="0" name="lastLogin" type="xs:dateTime"/>
<xs:element minOccurs="0" name="lastUpdatedOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="pwd" type="xs:string"/>
<xs:element minOccurs="0" name="userFullName" type="xs:string"/>
<xs:element minOccurs="0" name="userName" type="xs:string"/>
<xs:element minOccurs="0" name="userPin" type="xs:string"/>
<xs:element minOccurs="0" name="validityEnd" type="xs:dateTime"/>
<xs:element minOccurs="0" name="validityStart" type="xs:dateTime"/>
</xs:sequence>
</xs:complexType>

**patientActivities** contains the full description of activities associated to patients

<xs:complexType name="patientActivities">
<xs:sequence>
<xs:element minOccurs="0" name="activityId" type="activities"/>
<xs:element minOccurs="0" name="activityStatus" type="xs:string"/>
<xs:element minOccurs="0" name="cancelationDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="caregiverId" type="caregivers"/>
<xs:element minOccurs="0" name="completionMark" type="xs:int"/>
<xs:element minOccurs="0" name="createdBy" type="xs:string"/>
<xs:element minOccurs="0" name="createdDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="finishedDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="id" type="xs:long"/>
<xs:element minOccurs="0" name="locationId" type="locations"/>

<xs:element minOccurs="0" name="patientId" type="patients"/>
<xs:element minOccurs="0" name="scheduledOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="startedDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="validationDate" type="xs:dateTime"/>
</xs:sequence>
</xs:complexType>

**activities** contains the generic activities, which eventually are linked to patiens

<xs:complexType name="activities">
<xs:sequence>
<xs:element minOccurs="0" name="activityDescription" type="xs:string"/>
<xs:element minOccurs="0" name="activityDificulty" type="xs:string"/>
<xs:element minOccurs="0" name="activityLenght" type="xs:integer"/>
<xs:element minOccurs="0" name="activityName" type="xs:string"/>
<xs:element minOccurs="0" name="contraindications" type="xs:string"/>
<xs:element minOccurs="0" name="createdBy" type="xs:string"/>
<xs:element minOccurs="0" name="createdDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="every" type="xs:int"/>
<xs:element minOccurs="0" name="id" type="xs:long"/>
<xs:element minOccurs="0" name="recommendations" type="xs:string"/>
<xs:element minOccurs="0" name="recoveryProgram" type="xs:string"/>
<xs:element minOccurs="0" name="recurenceType" type="xs:string"/>
<xs:element minOccurs="0" name="validityEnd" type="xs:dateTime"/>
<xs:element minOccurs="0" name="validityStart" type="xs:dateTime"/>
<xs:element minOccurs="0" name="weekDays" type="xs:string"/>
</xs:sequence>
</xs:complexType>

**caregivers** is a structure where data about the caregives is stored

<xs:complexType name="caregivers">
<xs:sequence>
<xs:element minOccurs="0" name="caregiverLanguage" type="xs:string"/>
<xs:element minOccurs="0" name="caregiverName" type="xs:string"/>
<xs:element minOccurs="0" name="caregiverType" type="xs:string"/>
<xs:element minOccurs="0" name="categoryId" type="nUserCategories"/>
<xs:element minOccurs="0" name="cell" type="xs:string"/>
<xs:element minOccurs="0" name="city" type="xs:string"/>
<xs:element minOccurs="0" name="email" type="xs:string"/>
<xs:element minOccurs="0" name="firstName" type="xs:string"/>
<xs:element minOccurs="0" name="fullAddress" type="xs:string"/>
<xs:element minOccurs="0" name="id" type="xs:long"/>
<xs:element minOccurs="0" name="idValue" type="xs:string"/>
<xs:element minOccurs="0" name="LPict" type="xs:string"/>
<xs:element minOccurs="0" name="lastName" type="xs:string"/>
<xs:element minOccurs="0" name="lastUpdatedOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="MPict" type="xs:string"/>
<xs:element minOccurs="0" name="macAddress" type="xs:string"/>

<xs:element minOccurs="0" name="nat" type="xs:string"/>
<xs:element minOccurs="0" name="phone" type="xs:string"/>
<xs:element minOccurs="0" name="registrationDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="removalDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="SPict" type="xs:string"/>
<xs:element minOccurs="0" name="state" type="xs:string"/>
<xs:element minOccurs="0" name="title" type="xs:string"/>
<xs:element minOccurs="0" name="userId" type="registeredUsers"/>
</xs:sequence>
</xs:complexType>

**locations** is a structure which indicat6es the locations where sensors are placed

<xs:complexType name="locations">
<xs:sequence>
<xs:element minOccurs="0" name="createdBy" type="xs:string"/>
<xs:element minOccurs="0" name="createdDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="id" type="xs:long"/>
<xs:element minOccurs="0" name="locationCoordinates" type="xs:anyType"/>
<xs:element minOccurs="0" name="locationDescription" type="xs:string"/>
<xs:element minOccurs="0" name="locationName" type="xs:string"/>
<xs:element minOccurs="0" name="locationType" type="xs:string"/>
</xs:sequence>
</xs:complexType>

**environment** data is a structure where data about environment (temperature, pressure, etc) is stored.

<xs:complexType name="environmentData">
<xs:sequence>
<xs:element minOccurs="0" name="airQualityIndex" type="xs:double"/>
<xs:element minOccurs="0" name="alt" type="xs:double"/>
<xs:element minOccurs="0" name="atmPressure" type="xs:double"/>
<xs:element minOccurs="0" name="batery" type="xs:double"/>
<xs:element minOccurs="0" name="carbonDioxide" type="xs:double"/>
<xs:element minOccurs="0" name="carbonMonoxide" type="xs:double"/>
<xs:element minOccurs="0" name="createdOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="deviceId" type="environmentDevices"/>
<xs:element minOccurs="0" name="humidity" type="xs:double"/>
<xs:element minOccurs="0" name="id" type="xs:int"/>
<xs:element minOccurs="0" name="lat" type="xs:double"/>
<xs:element minOccurs="0" name="lng" type="xs:double"/>
<xs:element minOccurs="0" name="nitrogenDioxide" type="xs:double"/>
<xs:element minOccurs="0" name="oxigenLevel" type="xs:double"/>
<xs:element minOccurs="0" name="ozone" type="xs:double"/>
<xs:element minOccurs="0" name="particles1" type="xs:double"/>
<xs:element minOccurs="0" name="particles10" type="xs:double"/>
<xs:element minOccurs="0" name="particles25" type="xs:double"/>
<xs:element minOccurs="0" name="temperature" type="xs:double"/>
<xs:element minOccurs="0" name="value1" type="xs:double"/>

<xs:element minOccurs="0" name="value2" type="xs:double"/>
<xs:element minOccurs="0" name="value3" type="xs:double"/>
<xs:element minOccurs="0" name="value4" type="xs:string"/>
<xs:element minOccurs="0" name="value5" type="xs:string"/>
<xs:element minOccurs="0" name="value6" type="xs:string"/>
<xs:element minOccurs="0" name="windDirection" type="xs:string"/>
<xs:element minOccurs="0" name="windSpeed" type="xs:double"/>
</xs:sequence>
</xs:complexType>

**environmentDevices** contains data describing the sensors used to registere data about environment (temperature, pressure, etc)

<xs:complexType name="environmentDevices">
<xs:sequence>
<xs:element minOccurs="0" name="createdOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="deviceConnectionData" type="xs:string"/>
<xs:element minOccurs="0" name="deviceIdentifier" type="xs:string"/>
<xs:element minOccurs="0" name="deviceName" type="xs:string"/>
<xs:element minOccurs="0" name="id" type="xs:int"/>
<xs:element minOccurs="0" name="locationId" type="locations"/>
<xs:element minOccurs="0" name="status" type="xs:string"/>
<xs:element minOccurs="0" name="validityFrom" type="xs:dateTime"/>
<xs:element minOccurs="0" name="validityTo" type="xs:dateTime"/>
</xs:sequence>
</xs:complexType>

**ealthAlerts** contains the data describing the alerts computed by the system when some conditions are met

<xs:complexType name="healthAlerts">
<xs:sequence>
<xs:element minOccurs="0" name="alertMessage" type="xs:string"/>
<xs:element minOccurs="0" name="alertStatus" type="xs:string"/>
<xs:element minOccurs="0" name="createdBy" type="xs:string"/>
<xs:element minOccurs="0" name="createdOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="id" type="xs:int"/>
<xs:element minOccurs="0" name="readOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="severity" type="xs:int"/>
<xs:element minOccurs="0" name="sourceRuleId" type="rules"/>
</xs:sequence>
</xs:complexType>

**messages** structure contains data which was sent as messages to users.

<xs:complexType name="messages">
<xs:sequence>
<xs:element minOccurs="0" name="createdOn" type="xs:dateTime"/>
<xs:element minOccurs="0" name="id" type="xs:long"/>

<xs:element minOccurs="0" name="messageContent" type="xs:string"/>
<xs:element minOccurs="0" name="receiverId" type="registeredUsers"/>
<xs:element minOccurs="0" name="senderId" type="registeredUsers"/>
<xs:element minOccurs="0" name="viewOn" type="xs:dateTime"/>
</xs:sequence>
</xs:complexType>

**patientsDataHistory** contains data describing the full history of data registered for a patient

<xs:complexType name="patientsDataHistory">
<xs:sequence>
<xs:element minOccurs="0" name="age" type="xs:integer"/>
<xs:element minOccurs="0" name="bloodPressure" type="xs:string"/>
<xs:element minOccurs="0" name="heartRate" type="xs:int"/>
<xs:element minOccurs="0" name="height" type="xs:double"/>
<xs:element minOccurs="0" name="id" type="xs:long"/>
<xs:element name="idPatientData" type="xs:long"/>
<xs:element minOccurs="0" name="operationTime" type="xs:dateTime"/>
<xs:element minOccurs="0" name="operationType" type="xs:string"/>
<xs:element minOccurs="0" name="otherInfo" type="xs:string"/>
<xs:element minOccurs="0" name="patientId" type="patients"/>
<xs:element minOccurs="0" name="registeredBy" type="caregivers"/>
<xs:element minOccurs="0" name="registrationDate" type="xs:dateTime"/>
<xs:element minOccurs="0" name="weight" type="xs:double"/>
</xs:sequence>
</xs:complexType>

## 2.4 EViF integrations

ESTABLISH Visualization Framework (EVIF) represents one of the main general purpose components of the ESTABLISH project. EVIF targets highly customizable visualizations and customized reports. EVIF features visualization widgets (e.g., line chart, bar-chart, legend, 3D building browser), that can be composed to create complex visualization. The creation of the visualization is performed via a web-based administrative interface that EVIF provides.

Figure 23 shows the overall architecture of EVIF. In detail, the architecture and end-user perspective of EFIV have been already described in the *D3.1 High Level System Architecture* deliverable. Here, we focus only on the data sources integration.

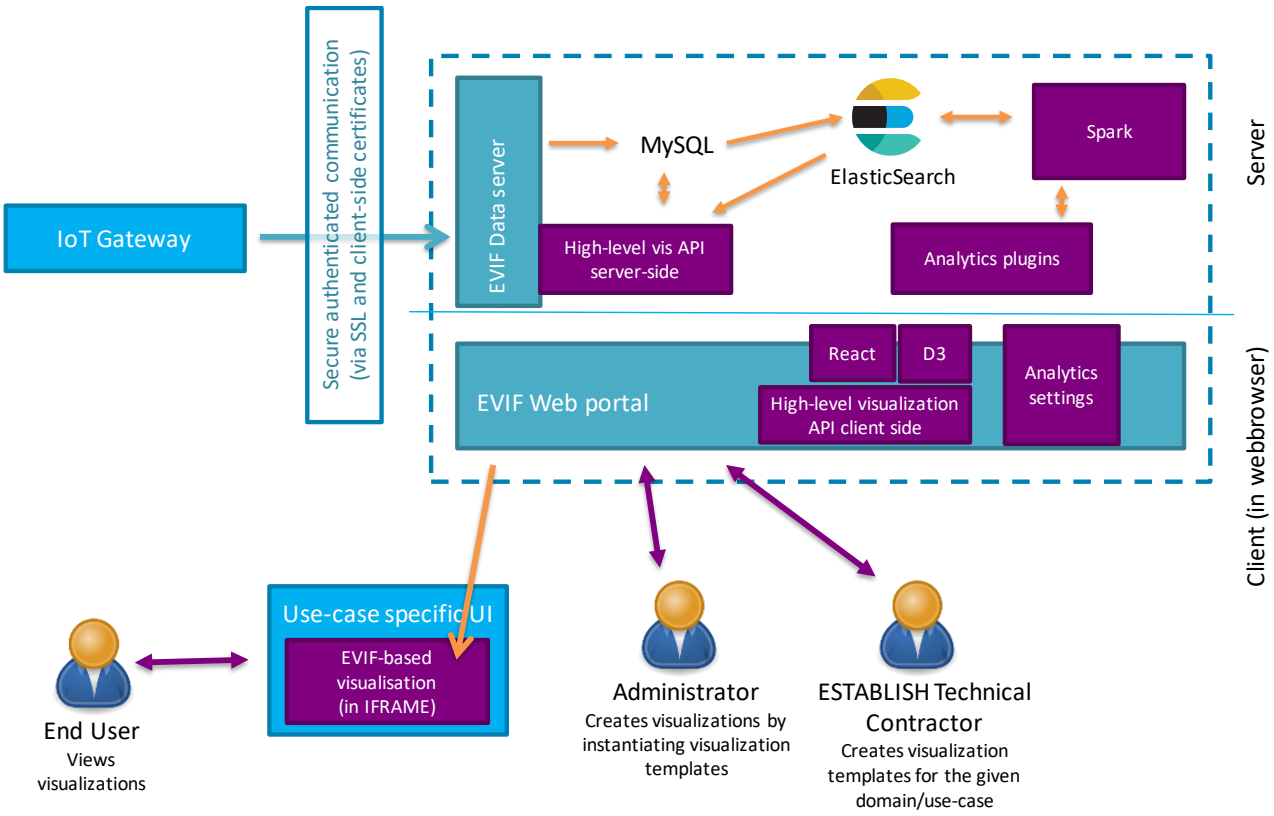


*Figure 23. EVIF architecture*

Data from sensors (or any other source) are in EVIF managed by its backend part (the server part in the figure), namely the EVIF Data Server component. Data are permanently stored in a MySQL database (and temporarily in Elastic search for fast indexing, searching, etc.) in an EVIF specific format. To the database, the data are a stored via *data intake connectors*, which need to be specifically prepared for each data source and/or data format, i.e., the perform transformations from a particular format to the EVIF format.

Currently, there are a number of connectors being developed for data from the individual partners within the project. Namely, these are:

- A connector for IMA and DEKPROJEKT demonstrator data (both of them share the same data format)
- A connector to BEIA/SIVECO demonstrator data
- A connector to data from VTT

The connectors use the APIs of the respective partners as described in the sections above.
EVIF pulls the data periodically using these connectors and stores them for further visualization in its database. This process typically takes two steps:

1) EVIF queries the list of available sensors and their allocation to end-users. It mirrors this structure in its own database. In particular, a sensor (potentially containing multiple attributes) is mapped to EVIF sensor. An end-user is mapped to EVIF user and associated namespace. The namespace serves as a container for all the sensors owned by the end-user. EVIF automatically creates corresponding sensors, users and namespaces. It configures these automatically created users in such a way that they have only access to sensors in their corresponding namespace.

2) EVIF pulls data for the detected sensors. The request for pulling the data specifies the last timestamp that EVIF registers for the particular sensor. Thus, only new data are pulled.

The data that are thus obtained can then be made available through various visualizations. A visualization is setup using an administrative interface of EVIF and made available to end-user as an EVIF panel. The panel is typically served as an IFRAME allowing thus seamless integration into existing UIs.

# 3 Smart city platform

In this pilot, all the data used comes from open data platforms or third-party web services. No sensors are provided by Establish partners.

The pilot is being developed in Valencia (Spain) and the data comes from the Valencia smart City Platform (VLCi) which is the name of its smart city platform.

In addition to the data obtained from the open data, the use case will make use of public weather forecast systems; this information is relevant to make predictions of pollution levels.

The data sources used in the pilot are:

- Air pollution stations.
- Measuring Stations for Pollen
- Bike lines
- Google Transit for public transport
- Parkings
- Real-time traffic status
- Sense of circulation
- Intensity of bicycle
- Intensity of traffic

The VLCi platform is based on Fiware, which is an open standard recommended by the European Commission for Smart Cities to ensure adaptation to the Internet of Things. The Main formats used by VLCi Open Data to expose the data are SHP, GML, WFS, WMS, KML, KMZ, CSV, JSON, JSON-LD, RDF XML/TURTLE /N3.

In addition to the data obtained from the open data, the use case will make use of weather forecast systems; this information is relevant to make predictions of pollution levels.

To support the large amount of data required by the use case, the data will be stored in a non-SQL scalable database. It will provide a Restful API to manage data

The selected data sets for the Spanish pilot are listed in the following table 13.

*Table 13. Selected Data sets of the Spanish Pilot*

| Domain | Type | Data set | Description | Update frequency | Format |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| Environment | Air pollution station | Air pollution station of the Universidad Politécnica (1A) | Daily data of the air pollution (PM2.5, PM1, SO2, NO, NO2, PM10, NOx, Ozone,) | Daily | CSV |
| Environment | Air pollution station | Air pollution station of the Molí del Sol (3A) | Daily data of the air pollution (PM2.5, PM1, SO2, NO, NO2, PM10, NOx, Ozone,) | Daily | CSV |
| Environment | Air pollution station | Air pollution station of the Pista de Silla (4A) | Daily data of the air pollution (PM2,5, PM1, Xylene, SO2, CO, NO, NO2, PM10, NOx, Ozone, Toluene, Benzene, Noise) | Daily | CSV |
| Environment | Air pollution station | Air pollution station of the Viveros (5A) | Daily data of the air pollution (PM2.5, SO2, NO, NO2, PM10, Ni, NOx, Ozone, As, Pb, Cd) | Daily | CSV |
| Environment | Air pollution station | Air pollution station of the Avinguda Francia (6A) | Daily data of the air pollution (SO2, CO, NO, NO2, NOx, Ozone, Speed) | Daily | CSV |
| Environment | Air pollution station | Air pollution station of the Boulevar Sur (7A) | Daily data of the air pollution (SO2, NO, NO2, PM10, Ni, NOx, Ozone, As, Pb, BaP, Cd) | Daily | CSV |
| Environment | Air pollution station | Air pollution monitoring network | Stations of automatic measurement of atmospheric pollution. | Daily | WFS, CSV, GEOJSON, SHAPE, GML, WMS, KML, KMZ |
| Environment | Pollen | Pollen Map Casuarina | Pollen map grouped by several tree species (Casuarina). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE GML KML KMZ |
| Environment | Pollen | Pollen Map Ulmus | Pollen map grouped by several tree species (Ulmus). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE WMS GML KML KMZ |

| | | | | | |
|---|---|---|---|---|---|
| Environment | Pollen | Pollen Map Ligustrum | Pollen map grouped by several tree species (Ligustrum). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE WMS GML KML KMZ |
| Environment | Pollen | Pollen Map Fraxinus | Pollen map grouped by several tree species (Fraxinus). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE GML KML KMZ |
| Environment | Pollen | Pollen Map Cupressus | Pollen map grouped by several tree species (Cupressus). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE GML KML KMZ |
| Environment | Pollen | Pollen Map Quercus | Pollen map grouped by several tree species (Quercus). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE GML KML KMZ |
| Environment | Pollen | Pollen Map Morus | Pollen map grouped by several tree species (Morus). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE GML KML KMZ |
| Environment | Pollen | Pollen Map Olea | Pollen map grouped by several tree species (Olea). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE GML KML KMZ |
| Environment | Pollen | Pollen Map Populus | Pollen map grouped by several tree species (Populus). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE GML KML KMZ |
| Environment | Pollen | Pollen Map Pinus | Pollen map grouped by several tree species (Pinus). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE GML KML KMZ |
| Environment | Pollen | Pollen Map Platanus | Pollen map grouped by several tree species (Platanus). Density: pollen density per zone. | 2 months | WFS GEOJSON SHAPE GML KML KMZ |
| Transport | Bike | Bike line | Cycling routes of the city of Valencia | Quarterly | WFS CSV GEOJSON SHAPE GML WMS KML KMZ |

| | | | | | |
|---|---|---|---|---|---|
| Transport | Bus | Google Transit, Lines, stops bus schedules | File Google Transit bus lines | 2 meses | ZIP |
| Transport | Car | Night parking in the bus lane | Night parking in the bus lane | annual | WFS CSV JSON SHAPE GML WMS KML KMZ |
| Transport | Car | Real-time traffic status | Traffic status data is updated every 3 minutes | 3 min | WFS, GEOJSON, SHAPE, GML, WMS, KML, KMZ, RDF, HTML, JSON-F, N3, XML, TURTLE, CSV, ATOM, JSONLD-G |
| Transport | | Circulation directions | Traffic circulation directions. Point element. | Semi-annual | WFS CSV GEOJSON SHAPE GML WMS KML KMZ |
| Transport | Car | Reduced Mobility Parking | Location of car parking for people with reduced mobility | Semi-annual | WFS, CSV, GEOJSON, SHAPE, GML, WMS, KML, KMZ |
| Transport | Bike | Intensity measuring point Bicycle (electromagnetic coils) | Geodata of measurement points bikes (grouping of electromagnetic coils) and its time intensity | Real time | GML, GEOJSON KML KMZ SHAPE WFS WMS RDF HTML JSON-G N3 XML TURTLE CSV ATOM JSONLD-G |
| Transport | Car | Intensity of traffic by sections | The intensity data is updated every 15 minutes, the unit of measure is vehicles / hour. The data shown are collected by the electromagnetic loops. | Real time | WFS GEOJSON SHAPE GML WMS KML KMZ RDF HTML JSON-G N3 XML TURTLE CSV |
| Transport | Car | Permanent no parking places | Geographical data on the location of the permanent no parking places | Annual | CSV GML GEOJSON KML KMZ SHAPE WFS WMS |
| Transport | Car | Unregulated parking | Geographical data on the location of unregulated parking spaces | Quarterly | CSV GML JSON KML KMZ SHAPE WFS WMS |
| Transport | Motorcycles | Parking for motorcycles | Geographical information on the location of motorcycle parking | Quarterly | CSV GML GEOJSON KML |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | KMZ SHAPE WFS WMS |
| Transport | Car | Intensity of Traffic Measurement Points | Geographical data of traffic measurement points (grouping of electromagnetic turns) and their hourly intensity | Real time | CSV WFS GEOJSON SHAPE GML WMS KML KMZ RDF HTML JSON-G N3 XML TURTLE ATOM JSONLD-G |
| Transport | Bicycle | Bicycle parking | Public bicycle parking: | Semi-annual | WFS CSV GEOJSON SHAPE GML WMS KML KMZ |
| Transport | Car | Car parking | Public and private parking of the city | Quarterly | WFS CSV GEOJSON SHAPE GML WMS KML KMZ |
| Transport | Car | ORA Expenders | Location of the ORA Expenders | Quarterly | WFS CSV GEOJSON SHAPE GML WMS KML KMZ |
| Transport | Car | ORA parking | Location of the ORA Parking | Quarterly | WFS CSV GEOJSON SHAPE GML WMS KML KMZ |
| Transport | Taxi | Taxi stops | Location of taxi stops | Semi-annual | WFS GEOJSON SHAPE GML KML KMZ |
| Transport | Bus | EMT Stops | Location of the bus stops | Daily | WFS CSV GEOJSON SHAPE GML WMS KML KMZ |
| Transport | Bicycle | Valenbisi stations | Information about the Valenbici stations (location, available bikes) | Real time | WFS CSV GEOJSON SHAPE GML WMS KML KMZ |
| Infrastructure | Street | Street works executed | Data on urbanization works carried out | Daily | CSV GML JSON KML KMZ SHAPE WFS WMS |
| Infrastructure | Street | Axes of the Streets works executed | Scope that covers the street in question. | Daily | WFS KLM KMZ GML SHAPE CSV GEOJSON WFS |
| Infrastructure | Street | Texts of street portals | Identification of the numbers of the portals | Daily | WFS CSV GEOJSON |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | SHAPE GML WMS KML KMZ |
| Infrastructure | Street | Street list | List of the streets of the city | Daily | CSV |
| Infrastructure | Street | Municipal districts | Territorial Division integrated by Municipal Districts according to the amount of population. | Weekly | WFS CSV GEOJSON SHAPE GML WMS KML KMZ |

**Data Storage**

ETL is a type of data integration that refers to the three steps (extract, transform, load) used to blend data from multiple sources. Data extraction is where data is extracted from homogeneous or heterogeneous data sources; data transformation is where the data is transformed for storing in the proper format or structure for the purposes of querying and analysis; data loading where the data is loaded into the final target database, more specifically, an operational data store, data mart, or data warehouse.

The ETL architecture of the Optimized city mobility planning is presented in the following figure 24.

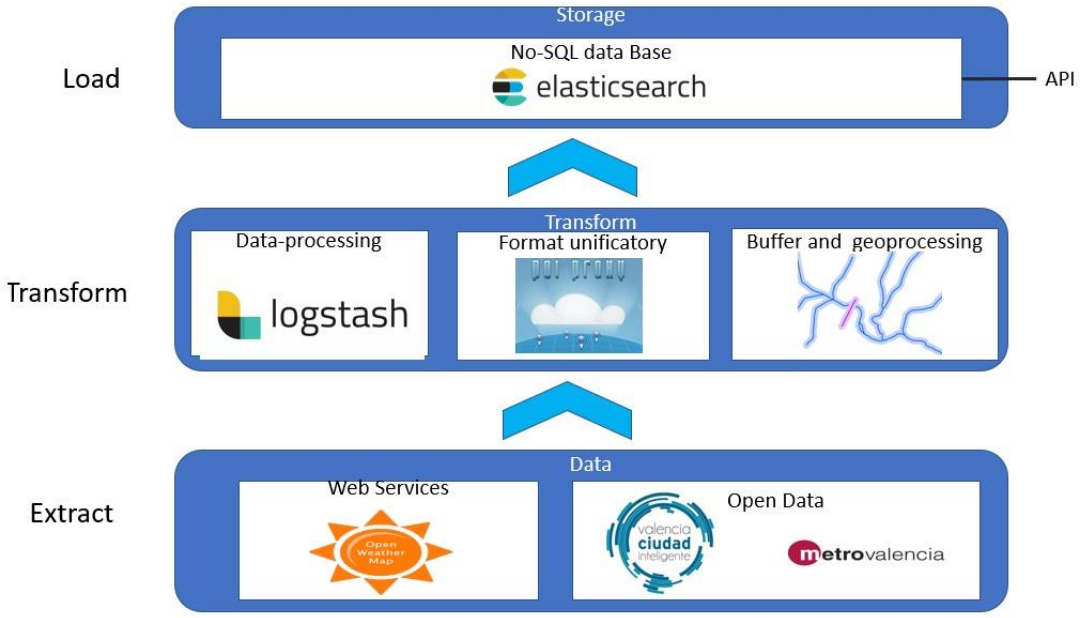


*Figure 24. Spanish pilot ETL*

The data base used to store the information needed for the use case is ElasticSearch.

Elasticsearch is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elasticsearch is developed alongside a data-collection and log-parsing engine called Logstash, and an analytics and visualisation

platform called Kibana. The three products are designed for use as an integrated solution, referred to as the "Elastic Stack" (formerly the "ELK stack").

Elasticsearch can be used to search all kinds of documents. It provides scalable search, has near real-time search, and supports multitenancy. Elasticsearch makes all its features available through the JSON and Java API.

### Data sources

In order to be able to perform data analysis, we first need to retrieve the data to be analyzed. Currently, data has been recovered from the VLCi platform, its API information can be found at the following URL http://gobiernoabierto.valencia.es/va/info-api/. The identifier of all available datasets (data-st/package) can be obtained from http://apigobiernoabiertocatalog.valencia.es/api/3/action/package_list.

As an example, if we wanted to recover the data from the Valenbisi stations, first we should recover information about the Valenbisi data set, for this we would call the following URL, http://apigobiernoabiertocatalog.valencia.es/api/3/action/package_show?id=estaciones-valenbisi , this query returns the URLs of all the formats in which the data are available, since the VLCi platform uses an Open Data product, which provides the information in several formats.


```
{
  ,
  success: true,
  result:
  {
        relationships_as_object: [ ],
        private: false,
        maintainer_email: "",
        num_tags: 1,
        id: "ee57f2d7-e1a4-48f4-aa78-b1476e2412fc",
        state: "active",
        valid: "",
        creator_user_id: "fb60b3c1-6708-4400- b8c9-5090d5bf4c93",
        type: "dataset",
```

**resources**:

[ ...

{

- **cache_last_updated**: null,
- **package_id**: "ee57f2d7-e1a4-48f4-aa78-b1476e2412fc",
- **webstore_last_updated**: null,
- **id**: "adadea80-37be-4f74-9392-0bdc95bd32a8",
- **size**: null,
- **name_va**: "Estacions de Valenbisi",
- **state**: "active",
- **hash**: "",
- **description**: "Localización de las estaciones de valenbisi.",
- **format**: "GeoJSON",
- **last_modified**: null,
- **description_va**: "Localització de les estacions de valenbisi.",
- **url_type**: null,
- **mimetype**: null,
- **cache_url**: null,
- **name**: "Estaciones de Valenbisi",
- **created**: "2014-10-28T12:13:33.581386",
- **url**: "http://mapas.valencia.es/ lanzadera/opendata/Valenbisi/JSON",
- **webstore_url**: null,
- **mimetype_inner**: null,
- **position**: 2,
- **revision_id**: "47cca846-c293-48c3-92f3-36b5148baaff",
- **resource_type**: null

},

...

This data set provides real-time data in several format, the data showed above is in GeoJSON format, as it has the coordinates of the stations, although they have to be transformed later.

Finally, to recover the data in Geojson format, the following URL is used: http://mapas.valencia.es/ lanzadera/opendata/Valenbisi/JSON", an extract of the data provided can be seen in the following figure 25.

{
"type": "FeatureCollection",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::25830" } },

"features": [
{ "type": "Feature", "properties": { "name": "012_CALLE_MINYANA", "number": "12", "address": "Calle Salvá - Calle Poeta Querol", "open": "T", "available": "6", "free": "14", "total": "20", "ticket": "T", "updated_at": "25\/08\/2017 10:46:53" }, "geometry": { "type": "Point", "coordinates": [ 725886.009, 4372445.348 ] } },

{ "type": "Feature", "properties": { "name": "013_PZA. ALFONSO MAGNANIMO_CON_CALLE_LA_NAVE", "number": "13", "address": "Alfonso el Magnánimo - Nave", "open": "T", "available": "24", "free": "0", "total": "24", "ticket": "T", "updated_at": "25\/08\/2017 10:46:53" }, "geometry": { "type": "Point", "coordinates": [ 726155.487, 4372463.775 ] } },

*Figure 25. Valenbisi Dataset in JSON format*

The geographical coordinates that the dataset returns, are in the reference system EPSG::25830, this data will be transformed to be able to be visualized in the map.

### Data acquisition

A set of processes were programmed to make periodic calls to the different datasets, the periodicity of each call depends on the frequency of refreshing them.

Logstash is used to program the processes. Logstash is a tool developed by Elastic that works under the Java JVM. It allows us to manage the logs of our applications, so we can use it to collect, parse and save the logs for later searches. This tool is based on the integration of inputs, codecs, filters and outputs. Inputs are the data sources that will be used later; codecs convert one input format into another that Logstash accepts, and the latter into an output format. Codecs are (usually) used when data is not plain text.

Filters are actions used to process events by modifying or deleting them. Finally, the outputs are the destinations where the processed data will be sent.

An example of how data related to bus stops would be imported can be seen in the figure below. The processes have three different parts:

- Inputs: defines the data source, format and refresh rate.
- Filter: defines the mapping between the data source and the destination, as well as the corresponding transformations.
- Output: defines the output of the data, with the index used to store the information.

```
input {
  http_poller {
    urls => {
      test1 => " http://geoserver:8080/geoserver/Establish/wfs?
service=wfs&typename=Establish:vlci_Emt_paradas&VERSION=1.1.0
&REQUEST=GETFEATURE&srsName=EPSG:4326&outputFormat=json"
    }
    request_timeout => 60
    codec => "json"
    interval => 100000000000
    metadata_target => http_poller_metadata_paradas_emt
  }
}

## Add your filters / logstash plugins configuration here
filter {
   if [http_poller_metadata_paradas_emt] {
    split { field => "[features]" }

    mutate {
      remove_field => ["http_poller_metadata_paradas_emt","[crs][properties][name]", "[features][type]",
"[crs][type]"]
      add_field => {
           "[@metadata][type]" => "emt_stops"
      }
    }
  }

}

output {
  if [@metadata][type] == "emt_stops"{
```

```
        elasticsearch {
                hosts => "elasticsearch:9200"
                index => "vlci_emt_stops"
                sniffing => false
    document_type => "measurement"
    document_id => "%{[features][properties][id_parada]}"
        }
  }
}
```

*Figure 26. logstash file for bus stops*

Some of these processes are very demanding, since hundreds of records are recovered for each call and an indexation has to be done that is very costly because it uses geometries.

Due to the high computational cost of processing some data sets, such as the location of the parkings and the intensity of the datasets, two processes have been carried out for each dataset. The first data process performs the initial load (very costly) and the second process only stores information that has been updated since the last time. In this way, the information load is lightened, as the costly process is only carried out once. This strategy has been followed for the processing of car parks and the intensity of traffic in the city.

# 4 Tracking of Athletes with Wearable Sensors

ESTABLISH will be implemented as a PaaS (Platform as a Service). The platform will have the ability to receive and store data from any device that has the capability of communicating and will have the ability to make suggestions based on the prescription (Rule sets) that will be created by users based on their profile. Platform will include content-based recommender system for recommendation. Thus, it will be a platform that can be used by amateur or professional individual and team athletes who want to record, report and manage their sports activities, as well as individuals who use lifestyle facilitating devices. The goal of Turkish consortium will implement necessary functions and algorithms that include coaching for the amateur and professional athletes, training management, and recommendation system that works with artificial intelligence.

The main components of the project are:

1. Extracting meaningful information by analyzing collected data from mobile applications and IOT devices.
2. Integration of semantic data using Big Data platform,
3. Content-aware adaptation and automation of the IOT infrastructure,
4. Development of a suggestion system based on the results of data analysis.
5. Development of mobile and web applications software for tracking data, accessing analysis results and tracking recommendations.

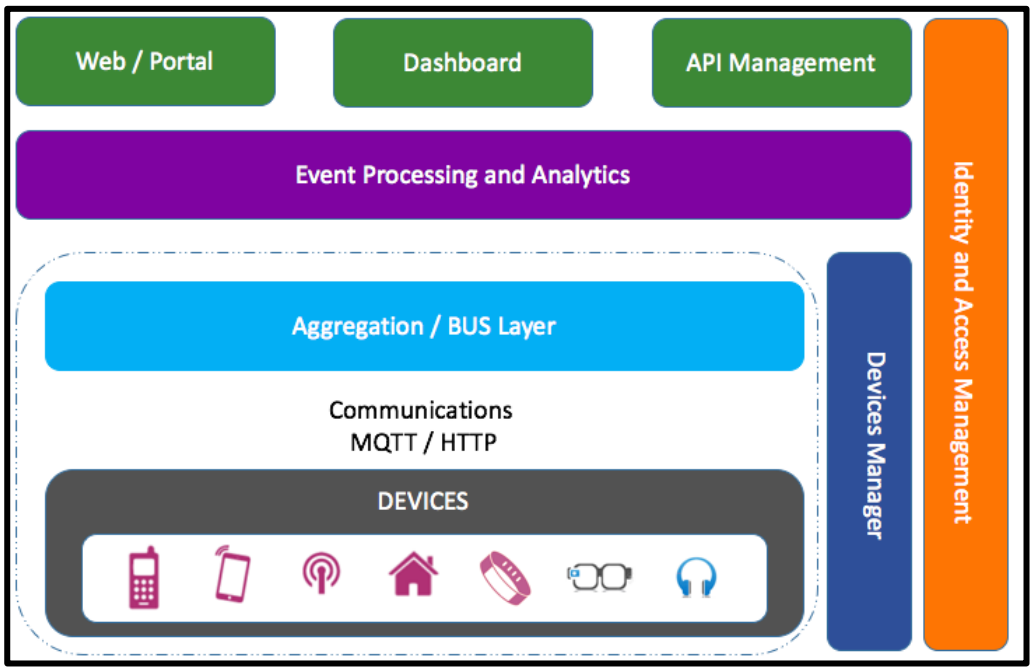


*Figure 27. Logical Layers of Establish Project from implementation point of view*

The logical layers of the project shown above says that the green boxes are integration points to Establish platform that is implemented to align all use cases. Turkgen and Semantik organizations are implementing the Dashboard and APIs for communicating the global Establish platform. These components provide the data transfer between global Establish platform and Establish platform for the Turkish use case.

# 5 Conclusion

Implementation and integration of adapters has to do with the loading and integration of sensor collected data. ESTABLISH partners have thrived to work together during this work package to enable synergies, this is why a clear distinction of different pilots in not evident in this deliverable.

The integration and implementation of data acquisition adapters concludes work package 4 of the ESTABLISH project. The work package 5 Data analytics and adaptive control will follow the work package 4 with utilizing collected and integrated data by analyzing it and enabling adaptive control systems to build systems to better the empowerment of end-users in affecting the quality of their living environment and personal health.