iSCAPE

Real time reporting system for monitoring with sensor technologies

D3.5 COMPREHENSIVE

06/2018

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 689954.

Project Acronym and Name	ISCAPE - Improving the Smart Control of Air Pollution in Europe	
Grant Agreement Number	689954	
Document Type	Deliverable	
Document version & WP No.	V. 1.2	WP3
Document Title	Real time reporting system for monitoring with sensor technologies	
Main authors	Guillem Camprodon, Viktor Smari, Victor Barberan, Oscar Gonzalez	
Partner in charge	IAAC	
Contributing partners		
Release date	22/06/18	

Table of Contents

Table of Contents

List of Figures

List of abbreviations

API Application Programming Interface

REST Representational state transfer

CSV Comma Separated Values

JSON Javascript Object Notation

MQTT Message Queue Telemetry Transport

DMP Data Management Plan

1 Executive Summary

The document showcases the current state of the ISCAPE software platform for ingesting, processing, storing, displaying and providing sensor data.

2 Introduction

The document presented here covers the ISCAPE software platform.

The Smart Citizen platform supports the core features of the platform. That means this report documents new components, developed specifically for the project, but also existing components that already existed and made possible the platform.

We believe building modular and reusable software and using existing platforms is critical towards optimizing the research and development effort. By increasing the technology readiness levels of existing technologies, we can drastically improve the project exploitation strategy.

Last but not least a special effort had been carried out to comply with the best practices described in the ISCAPE Data Management Plan (DMP) in special in those topics involving Personal Data and the Fair Data Principles.

3 ISCAPE Software Platform

Sensors and data play a critical role in the ISCAPE project. In one side the project focuses on developing a sensor infrastructure to enable individuals and communities to collect and contribute environmental data at the urban level, in particular, air pollution. On the other, we are also aiming at integrating other existing data sets already available on the existing cities, especially real time information coming from local governments and universities.

That establishes the need for a robust and flexible ecosystem to collect, process, store this data within the project. Furthermore, the project requires for an ecosystem to enable the exploration and exploitation of this data involving different targets from citizens to academia.

The primary requirements for the Software Platform were collected based on the work carried out in WP2 and WP3.

- Provide access to the project environmental sensors in near real time
- Support multiple sensor types and configurations with the associated metadata.
- Facilitate the exploration of data with other contextual data (maps, keywords) for non-expert users.
- Support other application to consume the existing information in to later be used for research.
- Support external data visualizations to be integrated with external websites, like the Virtual Living Lab (Task 8.1)
- Comply with the ISCAPE Data Management Plan (Deliverable 3.2) specification on data storing and archiving.

4 The Smart Citizen Platform

The previous requirements led to the decision of building the core platform on top of the existing Smart Citizen Platform. The platform is a front and backend solution for ingesting, storing and interacting with public data with a particular focus on crowd sensing applications. The platform is the software solution behind the Smart Citizen project as the result of the knowledge acquired after five years of running the Smart Citizen project at IAAC.

That means sensor data on ISCAPE is managed by a robust and mature framework but even more important it stay on the platform designed to remain long after the project is over.

From a user level that are the core features provided by the platform:

- A tested and robust platform towards data reliance and archiving.
- A built-in interface designed for non-experts users to access and explore the data.
- A framework already used by other projects ensuring the platform sustainability while the project is over.
- An open-source code base allowing other parties to run and operate the platform even if this has to close ensuring a long term exploitation strategy for the project.

Despite the potential of the Smart Citizen platform some technical work needed to be carried out. The work primarily involved the following tasks:

- Enhance the application data framework to support multiple sensor types and algorithms as required by the ISCAPE project, some sensor data is processed after it is received,
- Provide support for uploading sensor data collected on an SD card, some sensor data in ISCAPE will gather data offline.
- Improve the Onboarding UI to help new users configuring and managing new devices.
- Improve the overall stability and performance of the platform to guarantee an SLA of >99% by improving the infrastructure management tools.

4.1 Supported datasets

The Smart Citizen platform has been improved in the past 18 months to support a wide variety of sensor data. That means virtually any sensor with a quantifiable output can be stored on the platform before defining the sensor specifications. For the time being this is the primary datasets expected as part of the ISCAPE project:

- **ISCAPE Citizen Kits:** This are low-cost environmental sensors aimed to be deployed by citizens themselves. They transmit the data using citizens home Wi-Fi connection to the platform over the Internet using MQTT, a lightweight protocol for sensor communication. They can also store data offline on an SD card.

- **ISCAPE Living Labs Stations:** This are environmental sensors aimed to be deployed by the Living Lab communities to monitor the effects of the local interventions. They transmit the data using any available Wi-Fi connection to the platform over the Internet using MQTT, a lightweight protocol for sensor communication. In some cases, this are connected to a 3G or LoRa access point to transmit the data over. They can also store data offline on an SD card.

- **Existing cities Air Quality Stations:** It includes data available online or offline in the form of a database, a web page or a file including updated data of a city Air Quality Stations. That is made possible thanks to a collection of software scripts already developed that can be customized to pull data from existing services to the Smart Citizen using the standard REST API. This feature has been tested by connecting the Barcelona Sentilo Sensor platform with the Smart Citizen Platform.

- **On site environmental sensor data collected by local research teams:** It might include other data that might be collected at the different ISCAPE sites or computer data based on existing platform data i.e. aggregated data based on multiple sensors readings. This data can be uploaded manually in a CSV form by a user or programmatically using the approach described for Air Quality Stations above.

4.2 Accessing and archiving datasets

The ISCAPE project aims not just at collecting data but primarily in how this data is used on a multi stakeholder level: from non-experts data visualizations developed at the Living Labs (WP2) to scientific, environmental modeling (WP5). Even more this features are critical towards the future exploitation of the project from an academic but also commercial level (WP7).

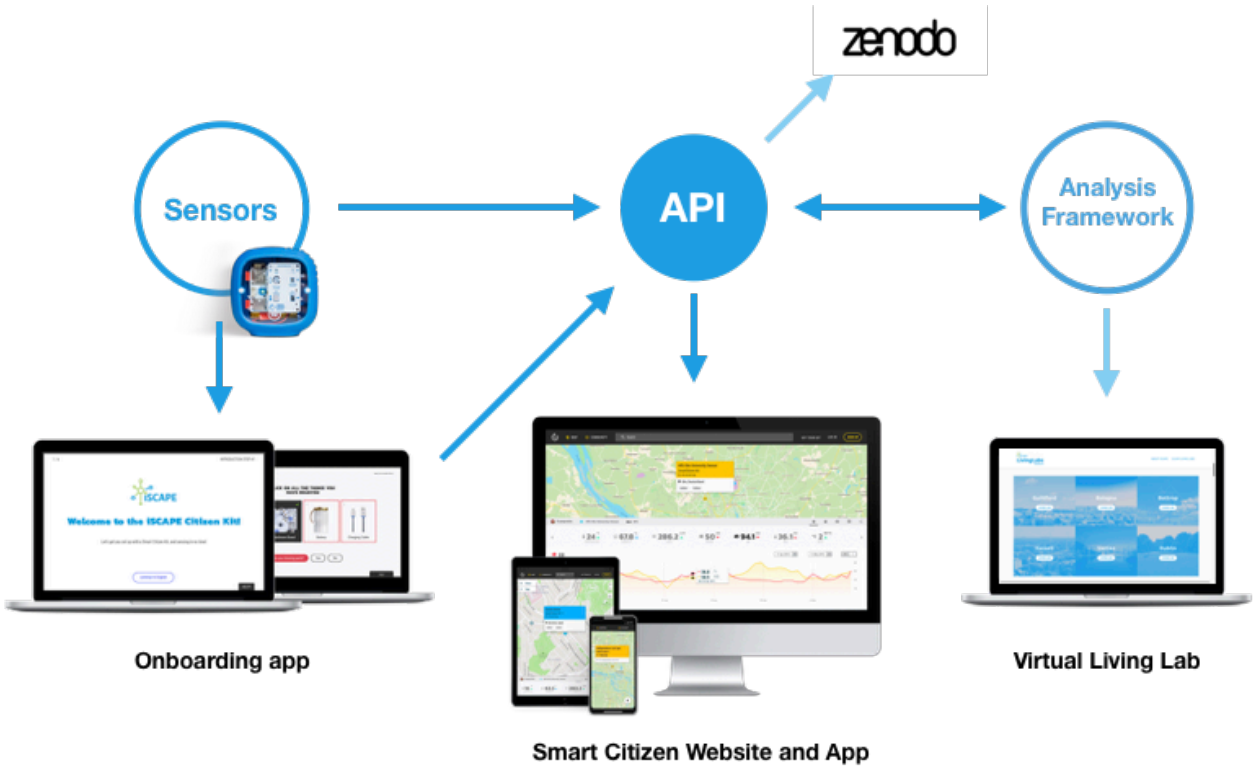

Figure 1 iSCAPE Software Platform

- **Smart Citizen Website:** The platform provides a visual website where the project environmental sensors can be accessed in near real time to facilitate the exploration of data with other contextual data (maps, keywords) and processed reports. This is especially important towards citizens engaging at each local site having a sense of ownership over a technology intervention has been associated with sustained community engagement (Balestrini et al. 2014)

- **Smart Citizen API:** The platform provides a REST interface for all the functionalities available on the Website. That allows applications to be developed on easily on top having access to all the features to create complex and rich tools. Some examples of this tools are: Smart Citizen Android App, ISCAPE Data Analysis Framework and the ISCAPE Virtual Living Lab.

- **Smart Citizen Android App:** The Android application for mobile phones allows users to locate and browse the latest data of a sensor quickly. This especially important for those citizens hosting one of the Citizen Sensor, engaging users by allowing them to look at their sensors data everywhere.

- **ISCAPE Data Analysis Framework**: It is currently being built as part of T3.1 to support the validation and calibration of the data collected by the different sensors. It is built on top of Jupyter Notebooks aimed at the various research departments involved in the project. It can use offline data in CSV format but also retrieve the data from the Smart Citizen API.

- **Virtual Living Lab:** It is currently being built as part of T8.1 and will provide an online web platform where each Living Lab can share their advances and contact with their local communities. The tools will feature different modules allowing the data from the sensors deployed by a Living Lab to be visualized on the site. It retrieves the data using the Smart Citizen API.

- **Onboarding app:** It aim to facilitate the process of sensor setup to ensure that users, irrespective of technical expertise, can install the sensors. It guides the user through the process of the setup using simple language and a friendly graphic language. It is built as a separate tool from the core Smart Citizen Webpage in order it can be customized for each deployment. It exchange data with the core platform using the Smart Citizen API.

- **Archiving for long term preservation:** As described on D3.2 DMP all the sensor data collected during the project will be later on submitted to the Zenodo platform for long term archiving and digital reference. As a European Commission supported initiative and technically supported by CERN, we believe this is the best way to ensure access to the generated data remains long after the project ends.

4.3 Smart Citizen Platform Architecture

The Smart Citizen Platform is a front and backend solution for ingesting, storing and interacting with urban data with a particular focus on crowd sensing applications.

The platform follows a modern modular architecture where the core components are independent.

- Robust and fast time series storage and processing
    - Fault tolerant distributed storage
    - On-the-fly data processing and aggregation
- Powerful sensor management
    - Real time sensors data ingestion
    - On-the-fly custom calibrations and validation for sensor data
- Social media capabilities
    - Built-in discussion system on data
- Fast and powerful search
    - Geolocation search engine

- Taxonomies search engine

- Single interface with applications roles management
  - Management back-office
  - Applications authorization management for OAuth 2

- Build on a fast and robust framework
  - Ruby on Rails with continuous integration workflow
  - Detailed public documentation and issue tracker

The platform structures in two main components:

- The Smart Citizen Engine: The cloud based data engine supporting: data ingestion, aggregation and retrieving. It also supports sensors metadata and social features. It is entirely independent of any web front-end exposing all the functionalities over a clear REST API. It is open-source and available under GNU Affero General Public License (AGPL).

- The Smart Citizen Website: The web front-end supporting user on the exploration of the sensors data with other contextual data (maps, keywords). It is open-source and available under GNU Affero General Public License (AGPL)

4.4 Data Ingestion Flow

As already described in the Supported Assets sections above the Platforms supports multiple sensor types and even data coming from other platforms. On the following section, we describe all the features supported when it comes to sending data to the platform.

4.4.1 Ingestion protocols

Two protocols are supported for data to be sent to the platform: MQTT API and HTTP API. MQTT is the one used by constrained devices as the Citizen Sensors and the Living Lab Stations. It allows the devices to post data to the platform after they are registered. It also allows them to receive configuration options (i.e. sensors reading interval) and report errors (i.e. sensors are malfunctioning). HTTP is aimed at applications publishing data to the platform (i.e. an existing sensors platform that also wants to make all the data available to the platform). This API gives access to all the platform functionalities as it is part of the core Smart Citizen API. Over this API we are not just limited to publish data but to register new devices or even users. In one side both ingestion protocols use transport encryption with TLS to ensure secure communication between the client and the server over the Internet.

4.4.2Authorization and authentication

Knowing who posts what is a serious problem when it comes to hundreds of sensor data being published per minute. Constrained hardware devices using the MQTT API use a unique device token given to the device every time is registered on the platform. The token authenticates the devices against the platform, and it can be expired at any time to prevent a device to keep publishing. Instead, the HTTP API supports authentication using an oAuth 2 or a private token. Both mechanisms work at a user level allowing a single process to manage all the devices created by a user.

4.4.3Kits blueprints

Each device sensors configuration needs to be previously registered on the platform to ensure each datapoint published is associated with the required metadata. This information is called a Kit blueprint. The minimal blueprint includes all the necessary data that a user might provide to create a Kit. It is composed of Components, and those can reuse existing Sensors and Measurements. Sensors are the hardware or software components that record the data. Measurements are descriptions of what the sensors are recording.

Kit Blueprints can be shared across many devices or can be unique per device to provide dedicated post processing formulas for Data Postprocessing per individual sensors. This is achieved with the Components binding.

4.4.4Data Postprocessing

Data processing designed to allow data from the sensors to be processed on the platform before being saved. The original, raw data and the processed data are always stored together in order the original data remains available in case it requires to be reprocessed.

This feature is of particular importance when we want a flexible way of processing the data without having to change anything at the device level. Other systems require the software running on the sensors, the firmware, to be updated. Instead, this approach allows to keep both datapoints and change the post processing algorithms at any time. It even allows having post processing algorithms uniquely per device. This use case is particularly important for AQM gas sensors where the calibration values might be defined for each sensor and even updated regularly based on onsite tests using reference equipment.

The post processing algorithms need to be written as a one component function using the Ruby language. Support for common tasks as lookup tables and linear regressions is provided by “Mathematician”, a custom library developed for this purpose. Once defined the function becomes part of the Kit Blueprint and the platform automatically applies it at ingestion time.

4.4.5Data Storage

Once the steps above are completed data is stored in a database cluster performing asynchronous masterless replication to ensure data backup and availability. Each datapoint is stored with the following items:

- Component: A reference to the component type that generated the datapoint.
- Device: A reference to the device that generated the datapoint.
- Raw Data: The datapoint as received to the platform
- Processed Data: The datapoint after applying post processing, when implemented.
- Timestamp: The time the datapoint was generated.

Once stored historical data available via the Smart Citizen API. All the other services, as the Smart Citizen Webpage, access the data from there. The API also exposes a method where data is processed to a CSV file and email to the user. That allows loading the data offline to any software capable of dealing with CSV files (i.e. Microsoft Excel, MATLAB, etc.)

5 Smart Citizen Engine

The Smart Citizen Engine is the cloud based software supporting: data ingestion, aggregation and retrieving. It also supports sensors metadata and user social features. It is entirely independent of any web front-end exposing all the functionalities over a clear REST API. It is open-source and available under GNU Affero General Public License (AGPL).

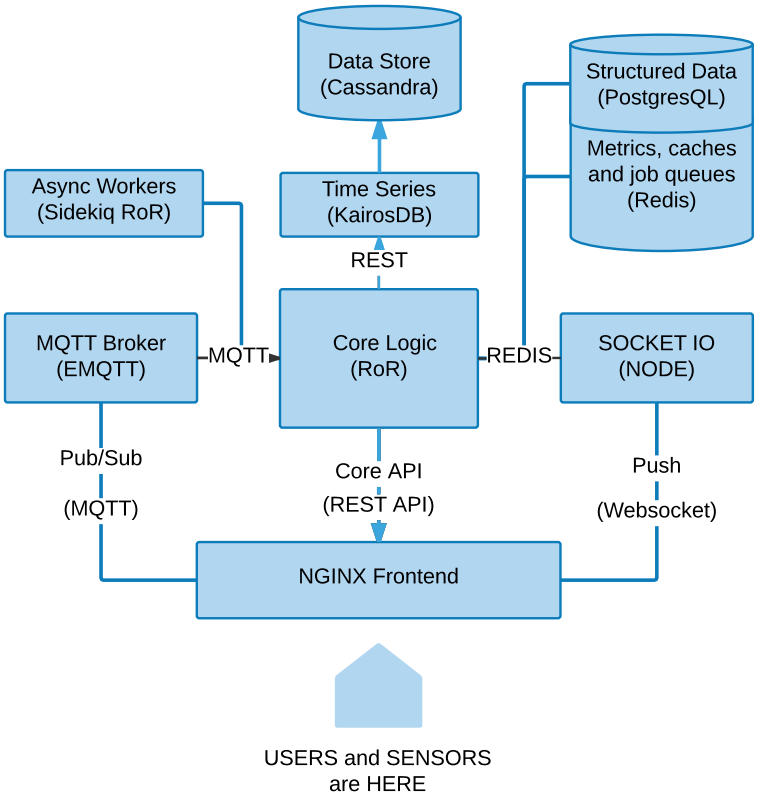

Figure 22 Smart Citizen Engine Architecture

5.1 Smart Citizen API

The following section describes how to use the API to retrieve and post sensors data to the Smart Citizen Platform. The API can also be used to manage devices and users automatically.

- Readings: Allows you to create and read datapoints for device in different forms
    - Get Latest Readings
    - Get Historical Readings
    - CSV Archive of readings

  - Post Readings
- Kits: Allows you to read all the kit blueprints available
  - Get All Kits
  - Get a single Kit
- Measurements: Allows you to read all the measurement types available
  - Get All Measurements
  - Get a single Measurement
- Sensors: Allows you to read all the sensor types available
  - Get all Sensors
  - Get a single Sensor
- Components: Allows you to read all the components types available
  - Get All Components
- Tags: Allows you to create, read, update and delete tags and associate them to devices
  - Get all Tags
  - Adding/editing a device's tags
  - Creating a Tag
  - Updating a Tag
  - Deleting a Tag
- Users: Allows you to create, read, update and delete users
  - Add a User
  - Get Current User (me)
  - Update Current User
  - Change an Avatar

- Reset a Password
- Get All Users
- Get a User

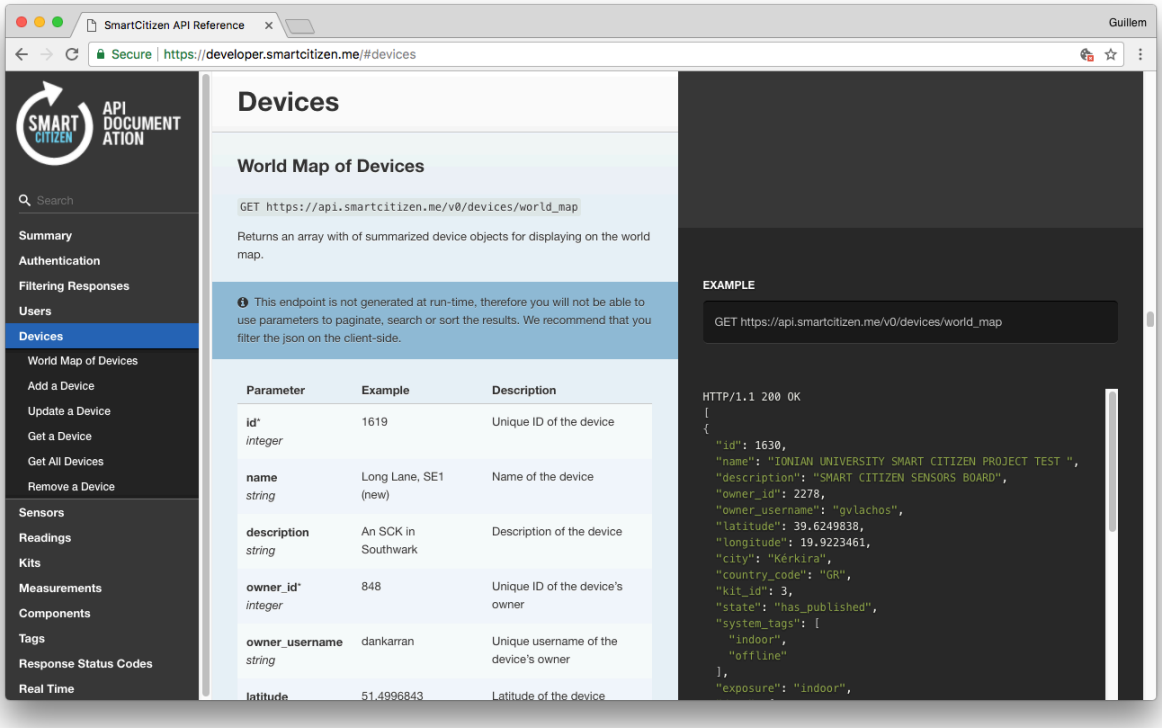

Figure 3 Smart Citizen API Documentation

https://developer.smartcitizen.me/

5.2 Implementation

From an internal architecture point of view, the Smart Citizen follows a service-oriented architecture build on flexible, independently deployable software systems. The benefit of distributing different responsibilities of the system into different smaller services is that it enhances the cohesion and decreases the coupling. That makes it easier to change and add functions and qualities to the system at any time. The following is a list of the most critical services inside the platform:

- Front Man Service: Acts a s load balancer, static file server and reverse proxy receiving all the connections from sensors and users and directing them to the right services. It is built around NGINX.

- Core Logic: Manages the core application logic of the platform and is built using the Ruby on Rails framework.

- Time Series Service: Stores and aggregates all the time series sensor data. It is based around Apache Cassandra.

- Database Service: Stores all the structured data on the platform including users and sensors metadata. It is built around Postgres SQL.

- MQTT Service: A pub/sub MQTT broker where the sensor devices connect to post and receive messages.

- oAuth Service: Manages access by third party applications to the platform

- Push Service: Exposes a WebSockets Push API for browsers notifications.

All the service are open-source. The core logic is available under GNU Affero General Public License (AGPL) requiring copies or adaptations of the codebase to be released under the same license. The other components are mostly released under BSD or Apache License.

6 Smart Citizen Website

The Smart Citizen Website supports the users to find relevant data. It does so by allowing for easy discovery and filtering of assets, based on geolocation, tags, groups, name and owners. Furthermore, the website also supports the explorative discovery of relevant assets through various data visualizations of the data from the different sensors – this allows the users to quickly and easily obtain relevant information and knowledge about the sensors.

For those users owning one of the sensors on the platform i.e. an ISCAPE Citizen Kit the website also supports them in managing the sensors. That involves deleting a sensor or updating its name or location.

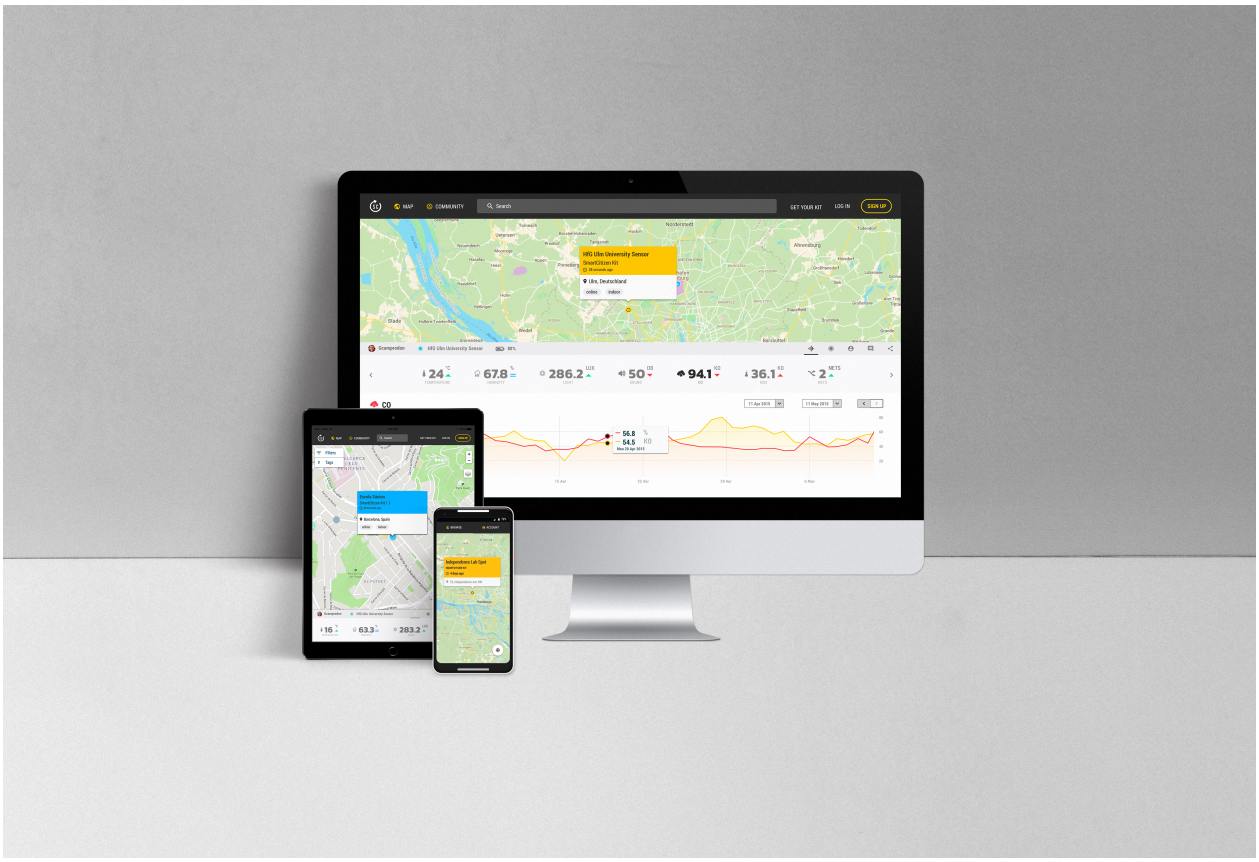

Figure 4 Smart Citizen Website

https://smartcitizen.me/kits/

6.1 User Interface

The Smart Citizen Website user interface is a set of visual tools to support users to discover data and manage their sensors on a single unified front-end. From the user side, the website together with the Virtual Living Lab platform will be their entry point to the data collected on the project. The key functionalities available on the platform are:

- Geo-centric browsable resources catalog
- Displays state and metadata from sources
- Incremental search for sensors and locations
- Supports time-series visualization
- Built-in commenting system
- Sensor management interface

It is important to notice the website is built on top of the Smart Citizen public API. This means that any operation performed by the current website is available over the API, allowing anyone new user interfaces to be developed or tools to perform some of the operations programmatically.

6.1.1Sensors discovery

The map interface provides a geographical exploration of assets at multiple scales. To maintain a low entry barrier, the interface is modeled on other existing map services the users might be familiar as Google Maps. Markers are aggregated by groups in order to improve the browsing experience.

This includes the following key features:

- Manually navigating the map by standard zoom and pan actions.
- Browsing the map by clicking at specific assets.
- Incremental search featuring simple places search.
- Client geolocation to center the map on to the user location.

Search is also considered a top level exploration tool by providing incremental search for searching across the different sensors and places progressively. Incremental search allows searching for assets name and metadata with auto-complete features. The search offers a simplified list of the available resources that users can quickly access without any intermediate process.

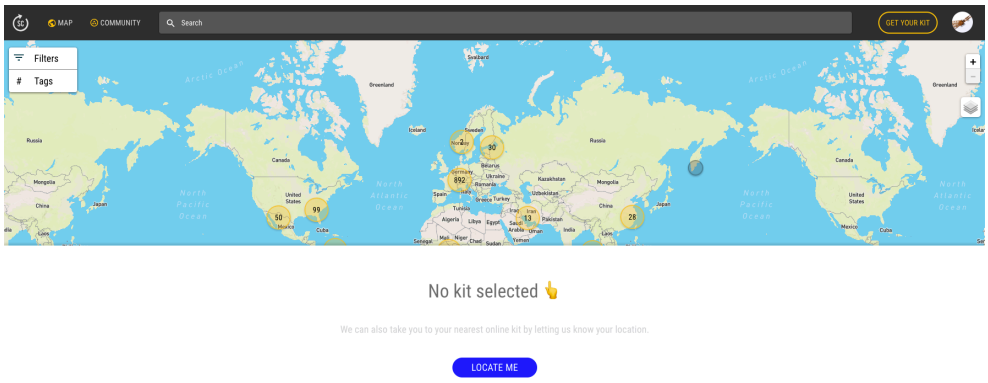

Figure 5 Smart Citizen Map Browser

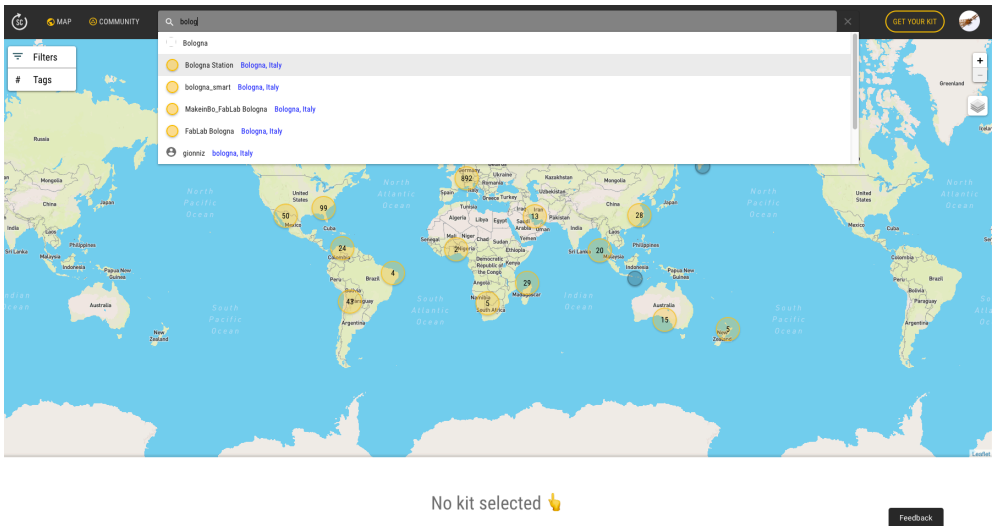

Figure 6 Smart Citizen Incremental Search

6.1.2Sensors view

The assets view is the core of the data exploration allowing the user to view all details of a sensor when selected during the exploration process. The main UI guidelines followed are focused on modularity. The system is designed following a system of horizontal blocks. The anatomy of the assets view contains the following sections:

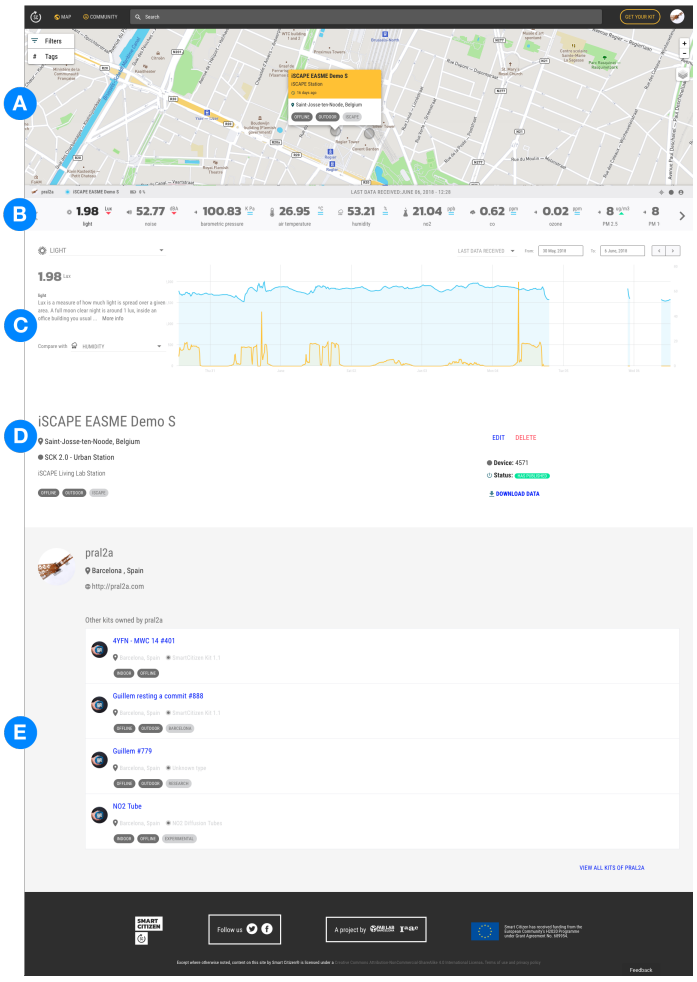

Figure 7 Smart Citizen Sensor View

- A Data Location: This module shows the geographical location of a resource on the map as described on the previous section. OC Assets always include location information as a common metadata

- B and C Data Visualization: This module is designed to support users on sensors data exploration.

    - Latest data module: This is designed as a carousel view displaying the latest data values of multiple attributes. It can work in conjunction with other data views or on its own if just latest data is available for an asset.

    - Series chart module: This module provides historical visualization of the data. It is a chart view capable of dealing with numeric data representation over time as the one coming from the sensors resources. The module includes date navigation

options as date pickers in order to access the historical data based on its time frame. It also features the possibility of comparing two metrics simultaneously.

- D Sensors details and metadata: This modules supplies a detailed insight on the sensors metadata. It provides information about the resource such as the owner or the sensor type, and thus supports users in getting a clear understanding of the data they are seeing. Data included in the view are:
    - The Asset Name
    - The latest time the Asset was updated
    - The Asset Unique Resource Identifier
    - The Asset Location
    - The Asset Description
- E User details: That modules provides a list with all the sensors owned by a user. It is designed to give users a sensor of ownership on their sensors and help them locate new sources.

6.1.3 Sensor and users management

The profile section is accessible by registered users and includes two main sections: user profile, where users can change their user information, and kits, where users can edit and delete the settings for the devices they own.

6.2 Implementation

The Smart Citizen Website user interface is a set of visual tools to support users to discover data and manage their sensors on a single unified front-end.

The Urban Data Observatory website is implemented as MVC browser application using Angular JS. It means the website behaves as a standalone app living in the user's browser and performing the required requests to the different services asynchronously.

From a development perspective, this has the advantage of simplifying the workflow of creating new features by providing a modular architecture of components structured as views, controllers and models or services. That allows different teams to work on the integration of new services independently while sharing reusing the interface components. By sharing a common front-end framework, we ensure user experience consistency and reduce development time.

An interface framework was created to provide a consistent visual interface across the entire application, built with Angular Material. That uses the Google Material design guidelines utilized by the company across their web and mobile applications.

To support the Geographic Information System allowing to display assets on a map a library has been developed based on Leaflet. The data visualization framework for time series sensor data and also for other datatypes that might come in the future has been built on top of D3 a library for producing dynamic, interactive data visualizations in web browsers. Disqus provides the comments and discussion interface.

7 ISCAPE Data Analysis Framework

The Data Analysis Framework is built with the purpose to help the project research community to process and analyze the data obtained from the sensors. The framework is based on Jupyter Notebooks, and the data analysis tools are based on Pandas and are ready to support Scikit later.

The current version supports data from the ISCAPE Citizen Sensors currently under tests, but support to integrate the data from existing equipment is also on its way. Currently, all the data is loaded as CSV files, but it is also ready to get live data directly from the Smart Citizen API.

The primary goal of the tools is to help us validate the different ISCAPE sensors and calculate their calibration values that later might automatically apply to the data the sensors push online using the Kit Blueprint feature described in previous sections.

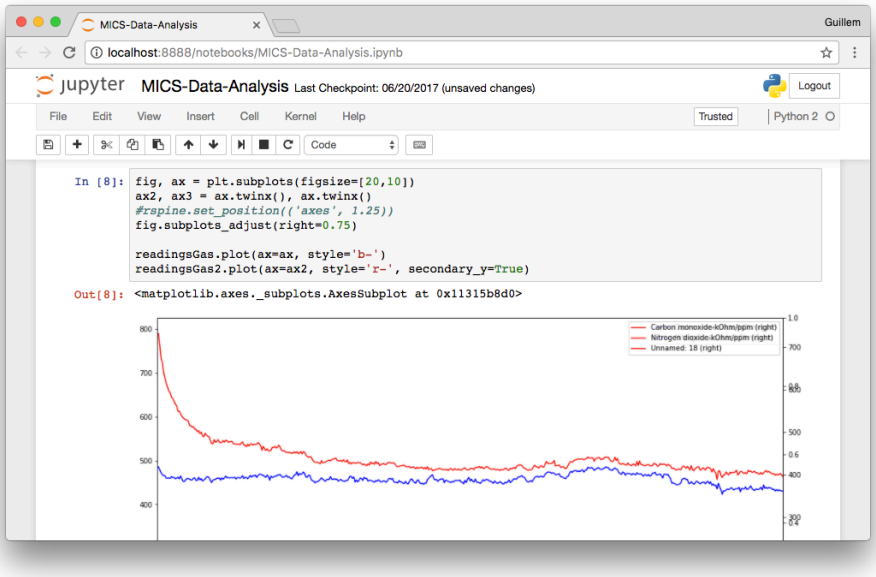

Figure 8 One of the ISCAPE Data Analysis Temlates for Jupyter

https://github.com/fablabbcn/smartcitizen-iscape-data/tree/internal_dev

8 Virtual Living Lab

The Virtual Living Lab has been already released online as in documented on ISCAPE D8.3 in Section 7

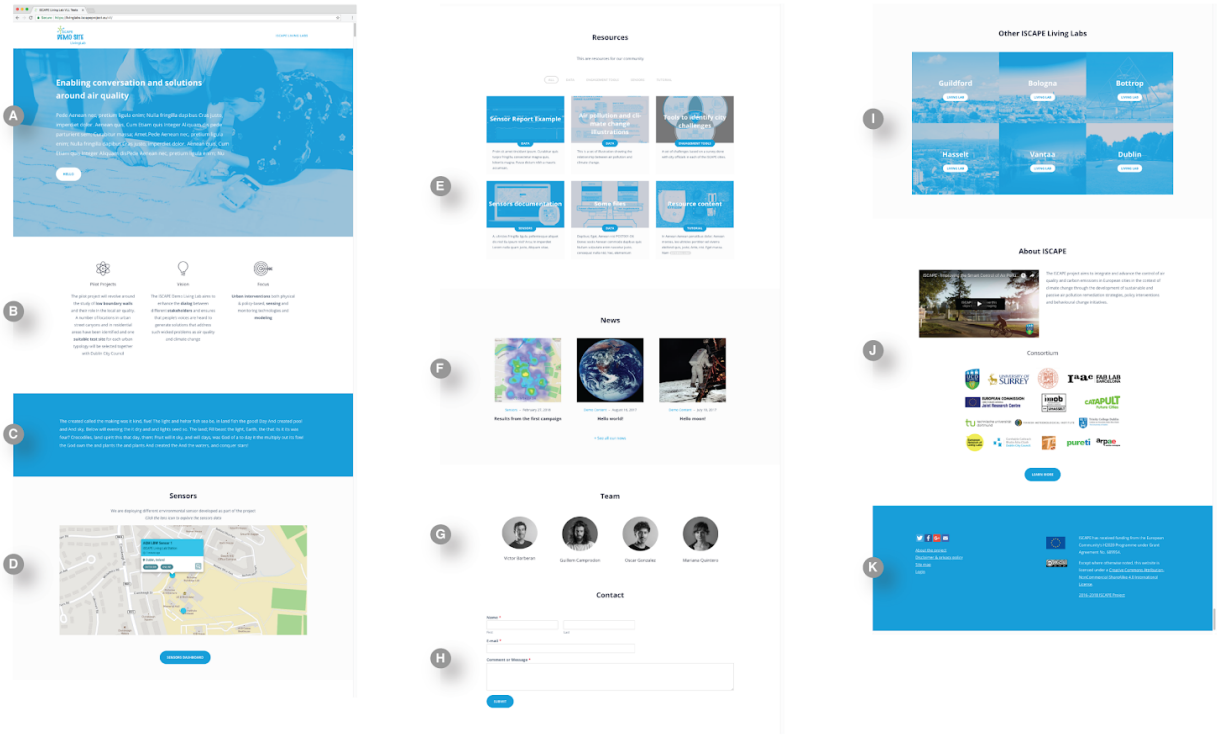

Figure 9 ISCAPE Living Labs Microsites Sections

A. **Introduction.** Shows an action claim and a brief introduction

B. **Insights**. Shows a description on the Pilot Projects, the Vision and the Focus.

C. **Claim.** Supports a short text to extend section A introduction.

D. **Sensors.** Displays a map with all the sensors registered under an specific Living Lab. Each sensor pop-up links to the corresponding sensor on the Smart Citizen platform.

E. **Resources.** Includes an index of the different resource pages.

F. **New.** Shows the latests News post published

G. **Team.** Shows a list of all the people involved on the Living Lab

H. **Contact.** Shows a contact form to reach the Living Lab leaders

I. **Living Labs Index.** Includes an index of all the other Living Labs

J. **About.** Briefly describes the project aims, includes a video and the partners list.

K. **Footer.** Includes the content license and other relevant information.

https://livinglabs.iscapeproject.eu/

Demo Site https://livinglabs.iscapeproject.eu/vll/

9 Onboarding app

The first problem to tackle when involving citizens with sensors is how to support them technically on the physical setup and software configuration of the sensors. The Onboarding App aims to facilitate the process of sensor setup to ensure that users, irrespective of technical expertise, can install the sensors. It guides the user through the course of the setup using simple language and a friendly graphic language.

To address this issue, the Onboarding Framework was previously developed by IAAC as part of the Making Sense project under the European Community's H2020 Programme Grant Agreement No. 688620. On the iSCAPE project, we built upon that tool enhancing its functionalities and creating a full new set of content specifically for the project.

The Onboarding Framework is a website for desktop and mobile built using Angular JS. It provides a robust design and technical framework where information is presented as a step by step guide support static content but also dynamic forms to dynamically interact with the users. One of the most critical features supported is to allow the user to send his Wi-Fi credentials to the Smart Citizen Kit along with pairing it with his account. The flexibility of the design enables supporting multiple processes depending on the user constraints.

Since the tool was designed with modularity and localization in mind on ISCAPE, we are integrating this tool to help citizen to configure and deploy the Citizen Kits. For the project, it has been customized to support the kit deployment. It also support localization of the content to each language and the specifications of each pilot.

Figure 10 The Smart Citizen Onboarding App

https://onboarding.iscape.smartcitizen.me/

10 Conclusion

We hope this deliverable contributes to a clear understanding of all the work that has been developed on the project towards building an innovative and useful software platform at all levels.

The project software team follows an Agile methodology and the work described here will be constantly evolving till the end of the project.