Project title:
Development of sensor-based Citizens' Observatory Community for improving quality of life in cities

Acronym: **CITI-SENSE** Grant Agreement No: **308524**

EU FP7- ENV-2012
Collaborative project

Deliverable D7.6 - Part 3: Citizens' Observatory Toolbox – Developer perspective

Version 4 - Part 3: Citizenz' Observatory Toolbox - Developer perspective

Work Package 7

Date: 07.12.2016

Version: 4.10

Leading Beneficiary:	SINTEF, Snowflake
Editor(s):	Arne J. Berre (SINTEF), Ballal Joglekar (Snowflake)
Author(s) (alphabetically):	Arne J. Berre (SINTEF), Alberto Fernandez (S&C)), Nicolas Ferry (SINTEF), Nicolas Ferry (SINTEF), Zvjezdana Knežević (Dunavnet), Sungchul Hong (KICT), Seonho Kim (Saltlux), Daniele Miorandi (U-Hopper), Richard Rombouts (Snowflake), Mirjam Fredriksen (NILU), Alberto Olivares (Snowflake), Maja Pokric (Dunavnet), Dumitru Roman (SINTEF), Miha Smolnikar (JSI), Andrei Tamlin (U-Hopper), Matevž Vučnik (JSI)
Dissemination level:	Public

Versioning and contribution history

Version	Date issued	Description	Contributors
4.01	01.06.2016	Initial version- part 3– new part extended from D7.6 part 1 Specification and part 2 Operation with focu on the developer perspective on the Citizens' Observatory Toolbox	Arne J. Berre
4.02	15.07.2016	Input on CITI-SENSE Data Web Services and Data Model update with CSV tables	Ballal Joglekar
4.03	15.08.2016	Input on SensApp Server Access	Nicolas Ferry
4.04	24.09.2016	Update on App architecture	Erik Forsen
4.05	29.09.2016	Update on Data Management	Ballal Joglekar
4.06	30.11.2016	Inclusion of new portal description and CSV file management	Arne J. Berre, Zvjezdana Knežević
4.10	07.12.2016	Update of reviewer's comments	Arne J. Berre

Peer review summary

Internal review 1			
Reviewer	Mirjam Fredriksen (NILU)		
Received for review	29.11.2016	**Date of review**	05.12.2016

Internal review 2			
Reviewer	Leonardo Santiago (Ateknea)		
Received for review	29.11.2016	**Date of review**	06.12.2016

Executive Summary

This document D7.6 CITI-SENSE Platform and architecture Version 4 - Part 3: Citizens' Observatory Toolbox - Developer perspective focuses on the project results from a Citizens' Observatory Toolbox (COT) perspective – with a focus on how the software and tools from the CITI-SENSE project might be used in future Citizens' Observatory projects. A user perspective on the Citizens' Observatory Toolbox can be found in deliverable D4.5 Citizens' Observatories Web Portal description.

The related D7.6 part 1 document describes the specification of the overall CITI-SENSE Architecture and platform, while D7.6 part 2 document describes the more operational details of the services, as well as having annexes with practical examples of the usage of the CITI-SENSE Architecture and platform. This D7.6 part 3 document presents the CITI-SENSE Observatory Toolbox results from the seven Toolbox areas of Methods, Data, Web Portals, Widgets, Sensors, Surveys and Mobile Apps.

Methods provides an overview of methodologies and methods developed in the project with references to the relevant deliverables. This document presents in particular a proposed Citizens' Observatory Interoperability method focusing on various interoperability areas that should be considered by any Citizens' Observatory project. Data is a main focus in this part 3 document – with a particular focus on how to set up a WFS server and also on how to access the actual data that has been collected through the CITI-SENSE project. Web Portals represents the most visible interface to the CITI-SENSE result and data. This document contains a further description of a new Map Query portal which provides access to the final collected CITI-SENSE data through a map interface linked to a new relational database containing the data from the CITI-SENSE project based on the CSV data available at the end of the project. Widgets are reusable user interface components that can be reused in future Citizen Observatories. The CITI-SENSE widgets are based on HTML5 to enhance the support for portability across platforms. Sensors and sensor platform development has been a main focus in the CITI-SENSE project. The section in this document describes the two principal ways for sensor platforms to interact with the WFS server – either through a push (API-based) or a pull (file-based) interface.

Surveys provide a good approach for interaction with citizens. CivicFlow is a service that can support surveys both from a smart phone and a web portal. Mobile Apps are important to support citizen's participation. CITI-SENSE has aimed at creating mobile apps that can run on multiple smart phone platforms through the use of frameworks like PhoneGap/Cordova and widgets supporting portability through HTML5. The SENSE-IT-NOW and CityAir App are presented as examples of the approach for the development of Mobile App apps.

In this document the chapters 2 to 6 presents the Citizens' Observatory Toolbox results from the developer's perspective for these areas as described above.

Table of contents

Index of figures

Index of Tables

Abbreviations and Acronyms

Table 0-1 Abbreviations and Acronyms

Abbreviation / Acronym	Description
CAQI	Common Air Quality Index
DAE	Digital Agenda for Europe
EIF	European Interoperability Framework
EIS	European Interoperability Strategy
FP7	Seventh Research Framework Programme of the European Union
GEOSS	Global Earth Observation System of Systems
GFM	General Feature Model
GML	Geography Markup Language
HTML	Hyper Text Markup Language
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
INSPIRE	Infrastructure for Spatial Information in the European Community
IoT	Internet of Things
ISO	International Organization for Standardization
LOD	Linked Open Data
O&M	Observations and Measurements
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
RDF	Resource Description Framework
REST	Representational State Transfer
SaaS	Software as a Service
SDI	Spatial Data Infrastructure
SOA	Service-oriented Architecture
SPARQL	SPARQL Protocol And RDF Query Language
SPARQL	SPARQL Protocol And RDF Query Language
SSNO	Semantic Sensor Networks Ontology
SSNO	Semantic Sensor Networks Ontology
URI	Uniform Resource Identifier
VGI	Volunteered Geographic Information

1 Introduction: Citizens' Observatory Toolbox - Developer

1.1 Introduction

The deliverable, D7.6 "Platform and Architecture – version 4.0" part 1, 2 and 3, provides the final version 4.0 of the specification and description for the CITI-SENSE architecture and platform.

This document D7.6 CITI-SENSE Platform and architecture Version 4 - Part 3: Citizens' Observatory Toolbox - Developer perspective focuses on the project results from a Citizens' Observatory Toolbox (COT) perspective – with a focus on how the software and tools from the CITI-SENSE project might be used in future Citizens' Observatory projects. A user perspective on the Citizens' Observatory Toolbox can be found in deliverable D4.5 Citizens' Observatories Web Portal description and is also reflected to through the CITI-SENSE web portal http://co.citi-sense.eu/

This D7.6 part 3 document presents the CITI-SENSE Observatory Toolbox results from the seven Toolbox areas of Methods, Data, Web Portals, Widgets, Sensors, Surveys and Mobile Apps.

Methods provides an overview of methodologies and methods developed in the project with references to the relevant deliverables. This document presents in particular a proposed Citizens' Observatory Interoperability method focusing on various interoperability areas that should be considered by any Citizens' Observatory project. These areas has also been presented and discussed with the other Citizens' Observatory projects through joint workshops and work in the tool group of ECSA (The European Citizen Science Association). It combines interoperability perspectives from INSPIRE information value chain, the European Interoperability Framework which includes interoperability for the following aspects: Policy, Legal, Organisational, Semantic and Technical. For a further refinement for technical interoperability the ISO 19119 Service areas provides a good foundation with boundary services with sensors and actuators and user interfaces, processing and data management services as well as communication and workflow services. Finally non functional aspects such as interoperability for security and privacy is considered.

Data is a main focus in this part 3 document – with a particular focus on how to set up a WFS server and the actual data that has been collected through the CITI-SENSE project. The CITI-SENSE architecture aims at the support of multiple types of sensors and mobile apps for collecting data, and the support of multiple ways of providing data for further use and processing, with the use of a common data model and WFS storage support for the Citizen Observation data. The approach for operational setup of a WFS storage server is described together with the final version of the CITI-SENSE data model. It is further explained how data storage can be achieved both through a WFS API interface and how data from sensors not using the API directly can be stored through a file ingestion process from CSV or JSON files. A set of relevant queries on observations based on type, time and location is supported through a REST interface. The final data sets collected in the CITI-SENSE project is represented through a collection of CSV files, based on the main classes of the data model. These can be used for further analysis directly or through ingestion into a new storage system, which also has been done for the final CITI-SENSE data into a new relational data base system at the end of the project.

Web Portals represents the most visible interface to the CITI-SENSE result and data. The deliverable D4.5 Citizens' Observatory provides a comprehensive description of the final co.citi-sense.eu web portal. This document contains a further description of a new Map Query portal which provides access to the final collected CITI-SENSE data through a map interface linked to a new relational database

containing the data from the CITI-SENSE project based on the CSV data available at the end of the project.

Widgets are reusable user interface components that can be reused in future Citizen Observatories. The CITI-SENSE widgets are based on HTML5 to enhance the support for portability across platforms. The CITI-SENSE project has provided widgets for a number of areas including maps with sensor locations and index values, real-time and historical sensor values, physical activity level maps and graphs, thermal & acoustic measurement graphs and widgets for questionnaires.

Sensors and sensor platform development has been a main focus in the CITI-SENSE project. The sensor platforms are comprehensively described in the D8.X deliverables from WP8. The data management support for the sensor platforms is described in more detail in various annex chapters in D7.6 part 2 on operational support. The section in this document describes the two principal ways for sensor platforms to interact with the WFS server – either through a push (API-based) or a pull (file-based) interface

Surveys provide a good approach for interaction with citizens. CivicFlow is a service that can support surveys both from a smart phone and a web portal. The widget framework for Questionnaires by U-Hopper is the basis for interaction with the survey services.

Mobile Apps are important for the support citizen's participation. CITI-SENSE has aimed at creating mobile apps that can run on multiple smart phone platforms through the use of frameworks like PhoneGap/Cordova and widgets supporting portability through HTML5. The SENSE-IT-NOW and CityAir App are presented as examples of the approach for the development of Mobile App apps. The CityAir app is provided from a user point of view as apps available through the Google Play store and through the Apple iPhone app store. From a developer's point of view the code is available as open source through a GitHub respository. Further work to abstract and generalise the apps as a basis for a more reusable app framework is in progress.

In this document the chapters 2 to 6 presents the Citizens' Observatory Toolbox results from the develoopers perspective for these areas as described above.

1.2 Citizens' Observatory Toolbox - Developer perspective

The main goal of the Citizens' Observatories Toolbox (COT) developed in the CITI-SENSE project is to provide various tools as guidance for the future usage by citizens, scientists and interest groups, as well as commercial interest.

The CITI-SENSE COT includes resources and guidance, procedures, software, hardware or services developed by CITI-SENSE that can be used to support citizens to participate in environmental monitoring and enable citizens to contribute to community based environmental decision making, and this document presents a developer's perspective into this, and also information about how data from the CITI-SENSE project can be accessed.

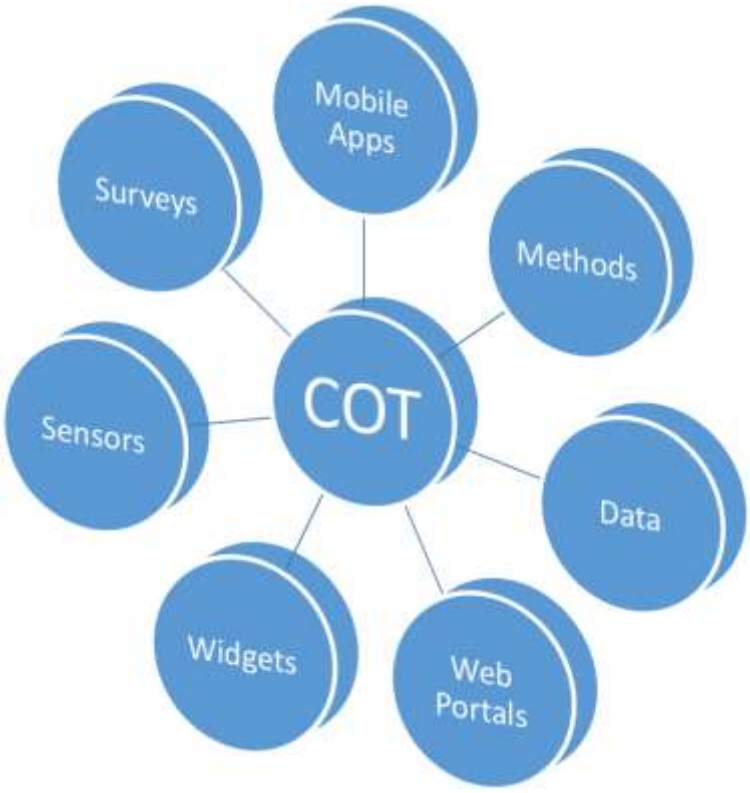

Figure 1-1 CITI-SENSE Citizens' Observatory Toolbox (COT)

The CITI-SENSE Citizens Observatories Toolbox addresses the following components:

- **Methods –** How Citizens' Observatory data is collected, managed and used – in particular focused on by WP2, WP3, WP4, WP5 and WP6
- **Data –** How the collected data is being transfomed, stored and accessed, in particular under the responsibility of WP7 Architecture and Platform.
- **Web Portals –** How data and other relevant information can be made available through web pages and the concept of a Citisen Observatory – focused on by WP4.
- **Widgets –** How data is being visualised – in particular focused on by WP7
- **Sensors –** How data is being collected through sensors – in particular focused on by WP8
- **Surveys –** How a citizen can be asked to provide their subjective views through questionnaires – in particular focused on by WP6 and WP7
- **Mobile Apps –** How a citizen can provide and interact on observation data through the use of smart phones/mobile phones – in particular focused on by WP6 and WP7.

A number of final deliverables in the CITI-SENSE project is also reporting results that are relevant for these areas – as follows – from http://co.citi-sense.eu/TheProject/Deliverables.aspx

The WP2 – Empowerment Initiative for Urban quality provides the final D2.4) Evaluation of the performance of the user cases.

The WP3 - Empowerment Initiative for Schools and Public spaces provides the final D3.4) Evaluation of the performance of the user cases.

The WP4 on Citizens Observatories provides the final D4.4 Methodology assessment – which gives an assesment of the set of methodologies with tools that have been developed in the CITI-SENSE projects.

The WP4 - Citizens' Observatories final deliverable D4.5 The CITI-SENSE Citizens' Observatories Web Portal contains a description of the co.citi-sense.eu web portal, which also presents the Citizens' Observatory Toolbox – with information both from a user's point of view and a developer's point of view.

The WP5 - Methods - Participation and empowerment – provides D5.3 Methodology and protocol for citizen empowerment, D5.4 Methodological study to support the representativeness of citizens' participation, D5.5 Co-ordinated analysis across empowerment initiatives and D5.6 Evaluation of empowerment initiatives

The WP6 - Methods - Information products and services with D6.4 Final report on methodology, D6.5 Report on implementation and demonstration and D6.6 Report on data fusion/data assimilation applications

WP8 with D8.5 Final sensor platform report provides information about the sensor platforms that has been developed and deployed in the CITI-SENSE project.

2 Methods

2.1 Overview of CITI-SENSE Methods

The CITI-SENSE project has produced a number of methodologies and methods.

The WP4 on Citizens Observatories provides the final D4.4 Methodology assessment – which gives an overview of the methodologies shown in Figure 2-1:

Methodology No	Methodology
1	Little Environmental Observatory (LEO)
2	CityAir Mobile App
3	Online Air Quality Perception Questionnaire
4	Environmental Monitoring Toolkit in Public Spaces
5	Data visualization Web Portal for Public Spaces Empowerment Initiatives
6	Data Visualization Web Portal for Outdoor Air Quality
7	Data Download Web Page
8	Citizens' Observatories Web Portal
9	Data Fusion Maps
10	Static Sensor Pack (AQMesh)
11	Spatial and Environmental Data Services (SEDS) Platform
12	Obeo radon sensor
13	Atmosphere sensor package

Figure 2-1 Overview of methodologies from D4.4

WP5 on Methods for Participation and empowerment has provided the D5.3 Methodology and protocol for citizen empowerment, the D5.4 Methodological study to support the representativeness of citizens' participation, the D5.5 Co-ordinated analysis across empowerment initiatives and the D5.6 Evaluation of empowerment initiatives.

WP6 on Methods for Information products and services has provided the D6.4 Final report on methodology the D6.5 Report on implementation and demonstration and the D6.6 Report on data fusion/data assimilation applications.

In this chapter the focus is on a method approach for Citizens' Observatory Interoperability developed in the context of WP7.

2.2 Citizens' Observatory Interoperability Method

In this section the focus is on a method approach on how to handle interoperability related to Citizens Observatories and Citizen Science. This approach has been discussed through a number of joint workshops also with the other Citizens' Observatory projects.

The figure 2-2 shows the various element in a Citizens' Observatory Interoperability reference model.

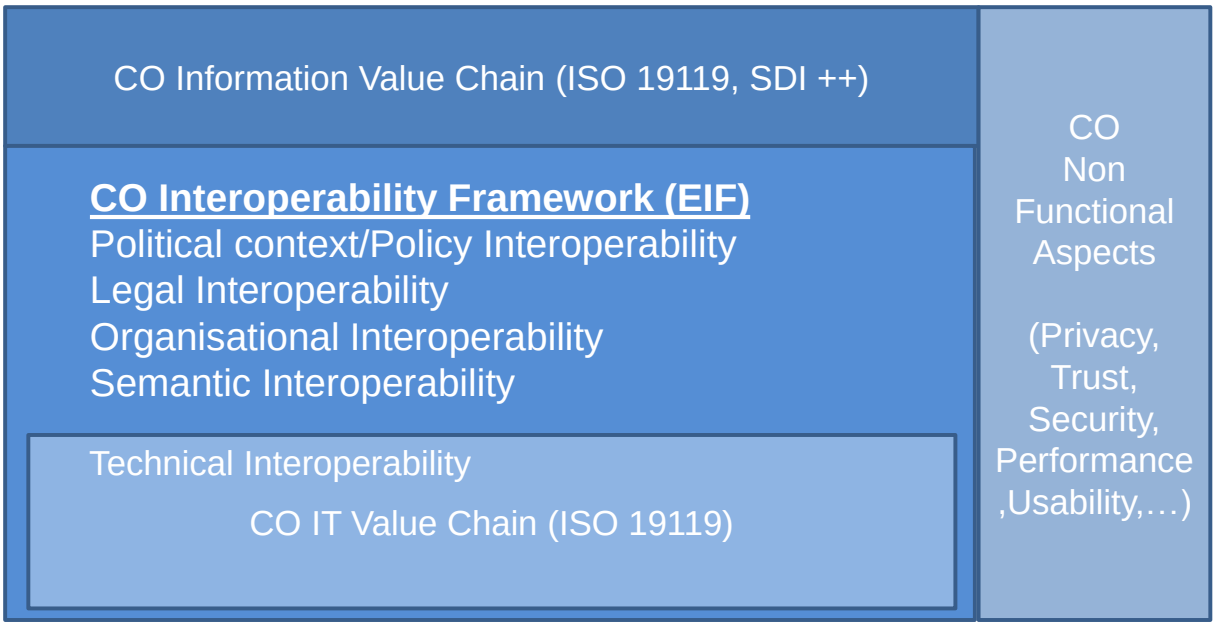

Figure 2-2 Proposed Citizens' Observatory (CO) Interoperability reference model

The figure 2-2 shows the different levels of interoperability that is relevant to consider in the context of a Citizens' Observatory.

CO Information Value Chain Interoperability is adapted from the INSPIRE information life cycle value chain. The CO Interoperability Framework is adapted from the European Interoperability Framework (EIF) with Policy Interoperability, Legal Interoperability, Organisational Interoperability, Semantic Interoperability and Technical Interoperability. In addition there is the orhogonal area of CO Non Functional Interoperability.

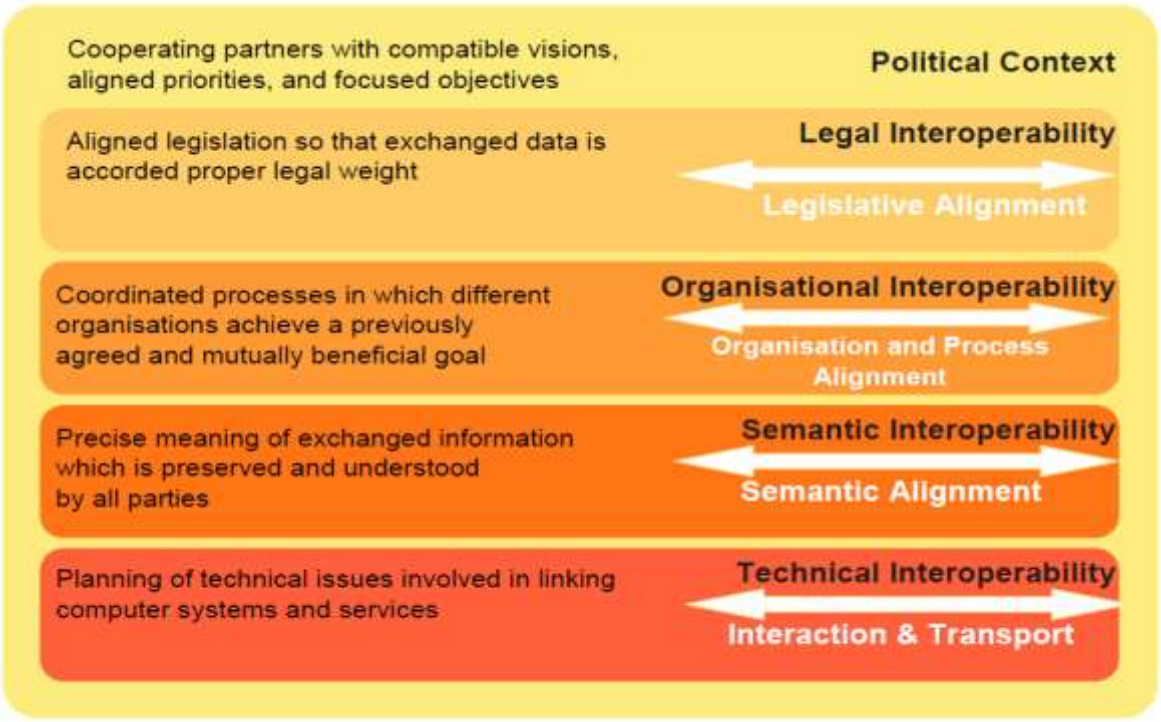

Figure 2-3 European Interoperability Framework

The figure above shows the different levels of interoperability described in the European Interoperability Framework
http://ec.europa.eu/idabc/en/document/2319/5644.html

In the following sections the various interoperability areas will be further described in the context of Citizen Observatories.

2.3 CO Information Value Chain Interoperability

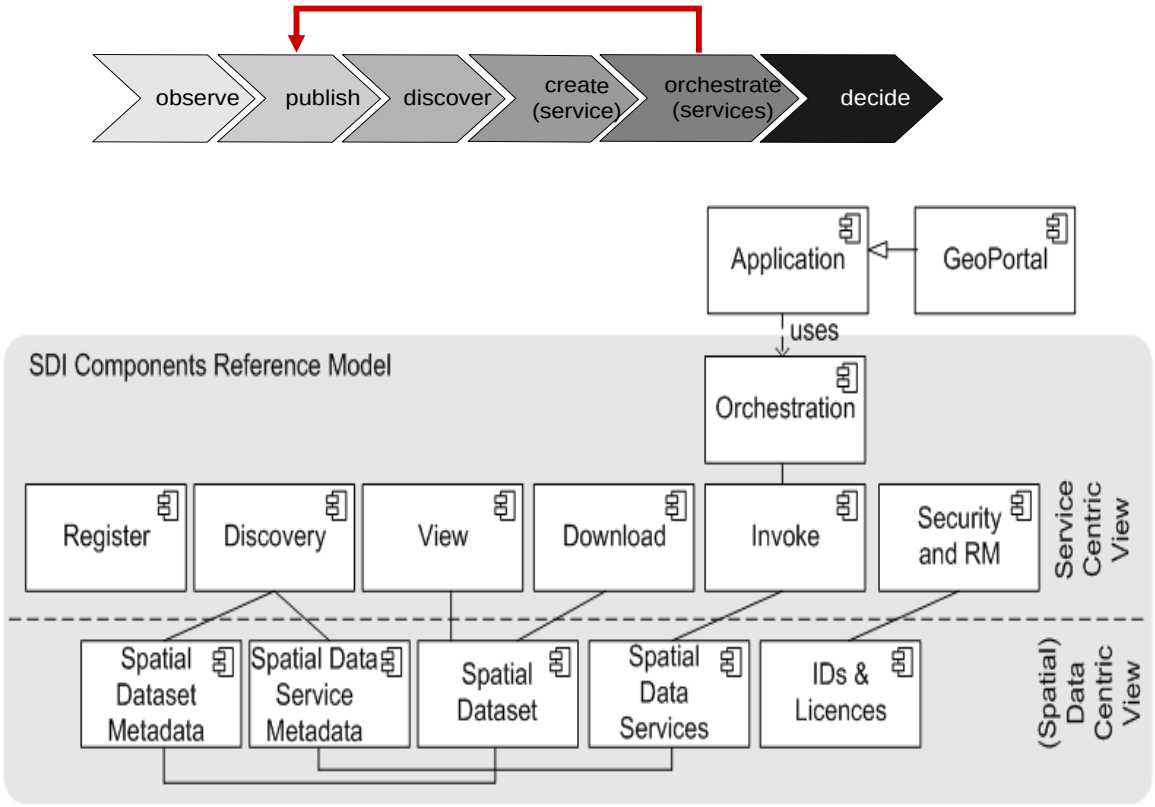

Figure 2-4 Information value chain and related INSPIRE services

The Figure 2-4shows a relevant infomation value chain from initial observations of data and until it is used as a basis for decisions or actions related to use of the data in relevant applications.

The life cycle approach will be able to manage and support a number of different resources, including human (citizen), sensor, data and service resources. Each of these resource types will be handled through the following phases Manage resources, Observe, Publish/Discover, Compose/Visualise/Analyse and Act/Notify. The CITI-SENSE platform and architecture has mainly had a focus on the first parts of the information value chain, in particular as much of the collected data has been of more experimental nature.

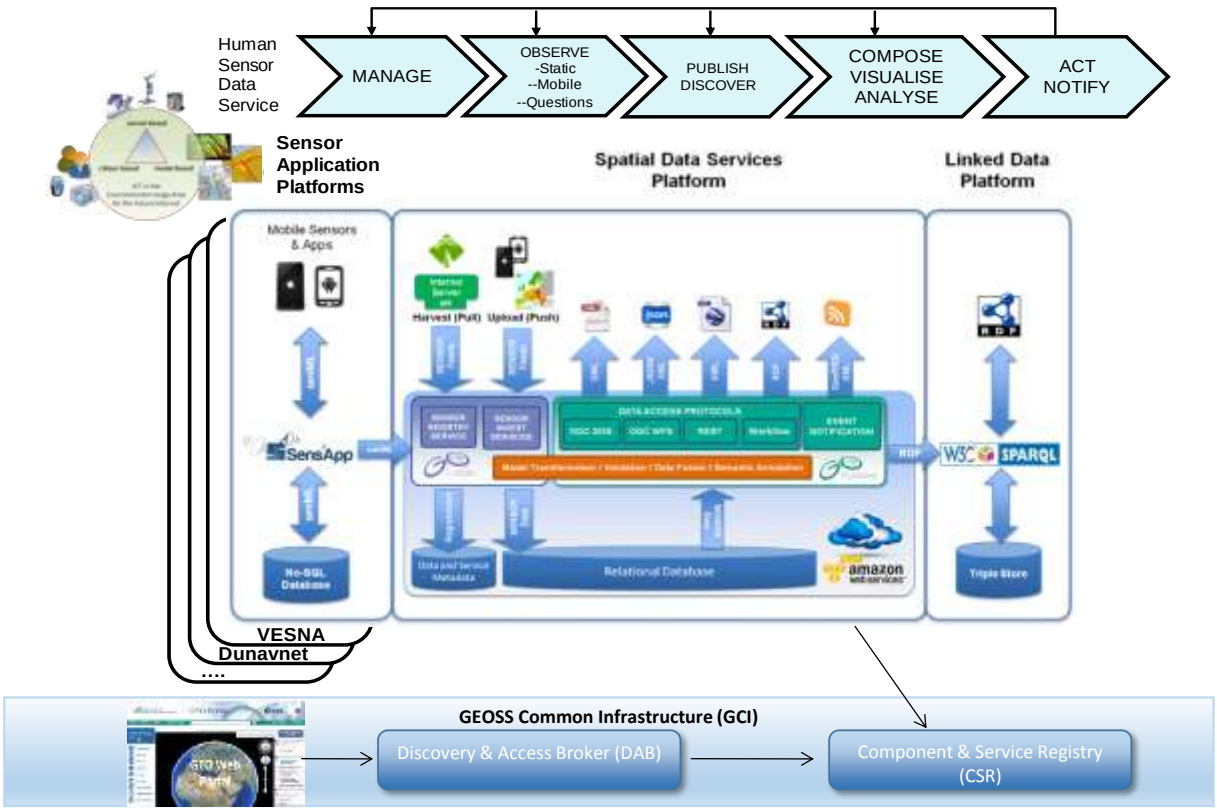

Figure 2-5 CITI-SENSE architecture under the perspective of an Information value chain

The Figure 2-5 shows the information value chain and the underlying architecture of the CITI-SENSE architecture and platform, with multiple input and outputs for the common data model and server.

Life cycle interoperability means that collected observation and measurement data will be followed and supported all theay through this information value chain.

2.4 Policy Interoperability

Policy interoperability means that there is cooperating partners with compatible visions, aligned priorities and shared focused objectives.

In the CO context this means that the observatory approach is aligned with the political objectives of supporting citizen's involvement and citizen science, locally, nationally or internationally.

Ideally, Citizens' Observatory initatives should have both top down support from a political and policy perspective as well as finding bottom up support from relevant citizen groups, both nationally and internationally.

2.5 Legal Interoperability

Legal interoperability means that the activities are related to relevant laws and regulations, and legislation. Citizen Observatories should carefully consider all relevant legislation relating to data exchange, including data protection legislation, when seeking to establish an Observatory service.

In the CO context this means that the observatory approach is aligned with the political objectives of supporting citizens' involvement and citizen science and is meeting related legal laws and regulations. In the context of citizens' observatories it becomes in particular important to comply with the new General Data Protection Regulation (GDPR) coming in Europe as the EU Data protection reform. http://ec.europa.eu/justice/data-protection/reform/index_en.htm

2.6 Organisational Interoperability

Organisational interoperability means that coordinated processes are harmonised among involved organisations in order to achieve previously agreed and mutually beneficial goals.

This aspect of interoperability is concerned with how organisations cooperate to achieve their mutually agreed goals. In practice, organisational interoperability implies integrating business processes and related data exchange. Organisational interoperability also aims to meet the requirements of the user community by making services available, easily identifiable, accessible and user-focused. Organisational interoperability implies alignment of relevant Business Processes or workflow, with clarification of organisational relationships and change management.

In the CO context this means that the observatory approach is executed with the knowedge and active involvement of the relevant organisations.

2.7 Semantic Interoperability

Semantic interoperability means that there is a precise meaning associated with the concepts and information that is being interacted around, which is preserved and understood by all parties.

Semantic interoperability enables organisations to process information from external sources in a meaningful manner. It ensures that the precise meaning of exchanged information is understood and preserved throughout exchanges between parties.

Achieving semantic interoperability for different Citizen Observatories and related data collection systems implies that there is an agreement of the meaning of relevant common terms and concepts.

Within the CO projects it has been an effort to discuss common vocabularies and semantic models. The foundation for relevant semantic models and ontologies has in particular related to the ISOP/TC211 and OGC data models for Observation and Measurement as described in the following.

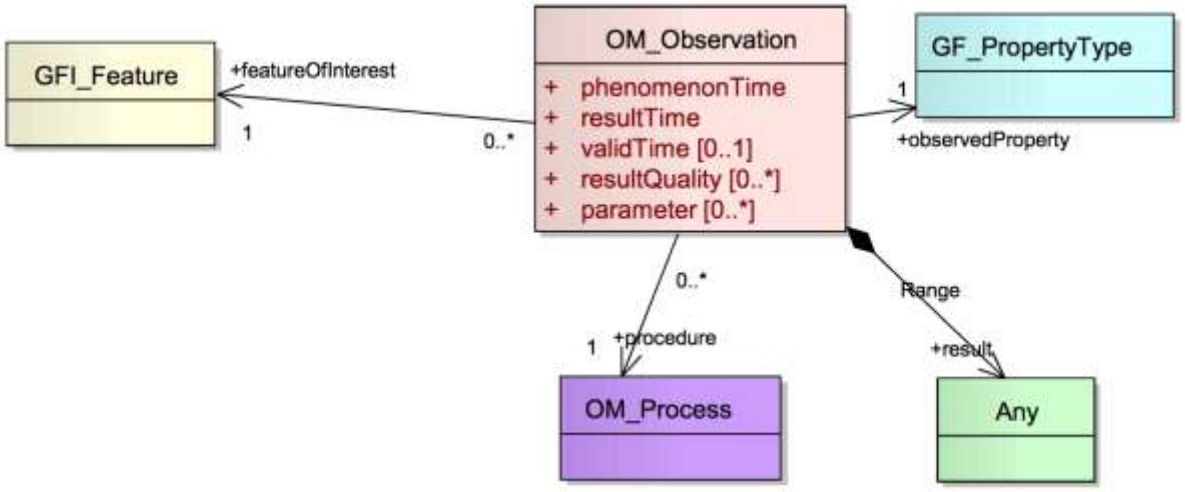

Figure 2-6 Core parts of the ISO/OGC Observation&Measurement base model (ISO 19156)

ISO 19156:2011 defines a conceptual schema for observations, and for features involved in sampling when making observations. These provide models for the exchange of information describing observation acts and their results, both within and between different scientific and technical communities.

Observations commonly involve sampling of an ultimate feature-of-interest. ISO 19156:2011 defines a common set of sampling feature types classified primarily by topological dimension, as well as samples for ex-situ observations. The schema includes relationships between sampling features (sub-sampling, derived samples).

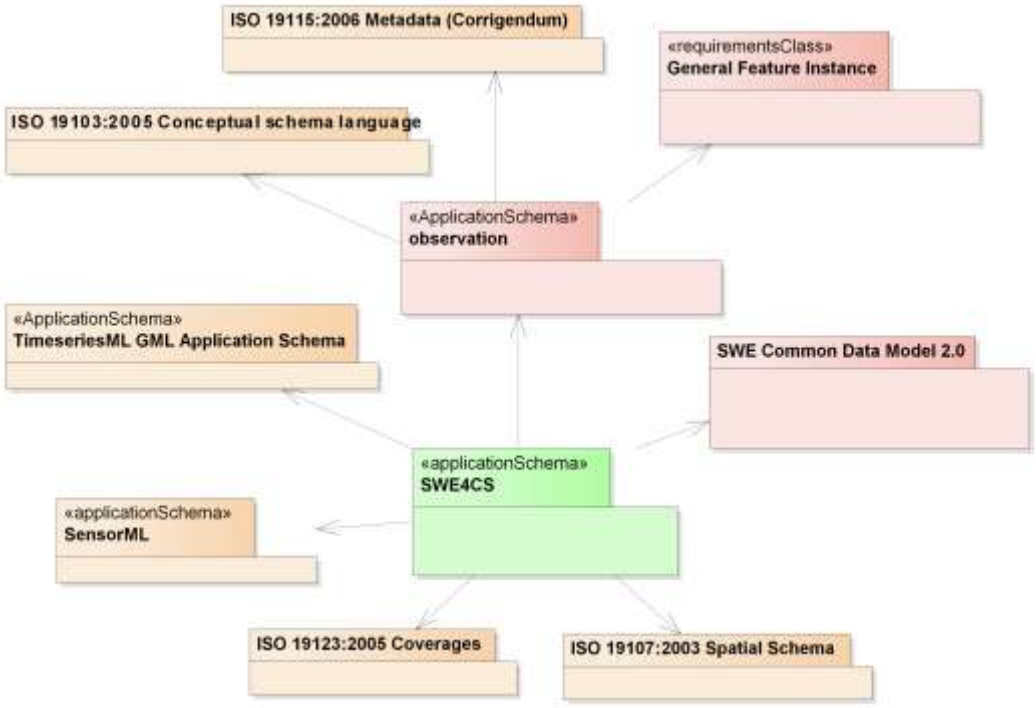

Figure 2-7 Basis for OCG SWE4CS - Sensor Web Enablement profile for Citizen Science

A working group among the Citizens' Observatory projects has arranged a number of workshops during the last years, led by the CobWeb project, which has brought the resulting discussions into a group activitiy in OGC on SWE profile for Citizen Science (SWE4CS).

The CITI-SENSE data model as described in section 3.3 is following the principles of the ISO/OGC Observation and Measurement model in such a way that it is easy to do mappings and transformations to other similar models and representation forms, evidenced by experiments at the end of the CITI-SENSE projects as reported in section 4.3.

2.8 Technical Interoperability

Technical interoperability means that the technical interfaces related to the technology layers in the interacting technologies are well defined in order to support linking of computer systems and services. This is also sometimes also called syntactical interoperability versus semantic interoperability.

This covers the technical aspects of linking information systems. It includes aspects such as interface specifications, interconnection services, data integration services, data presentation and exchange, etc. Citizens Observatories should agree on the formalised specifications to ensure technical interoperability when establishing an Observatory service.

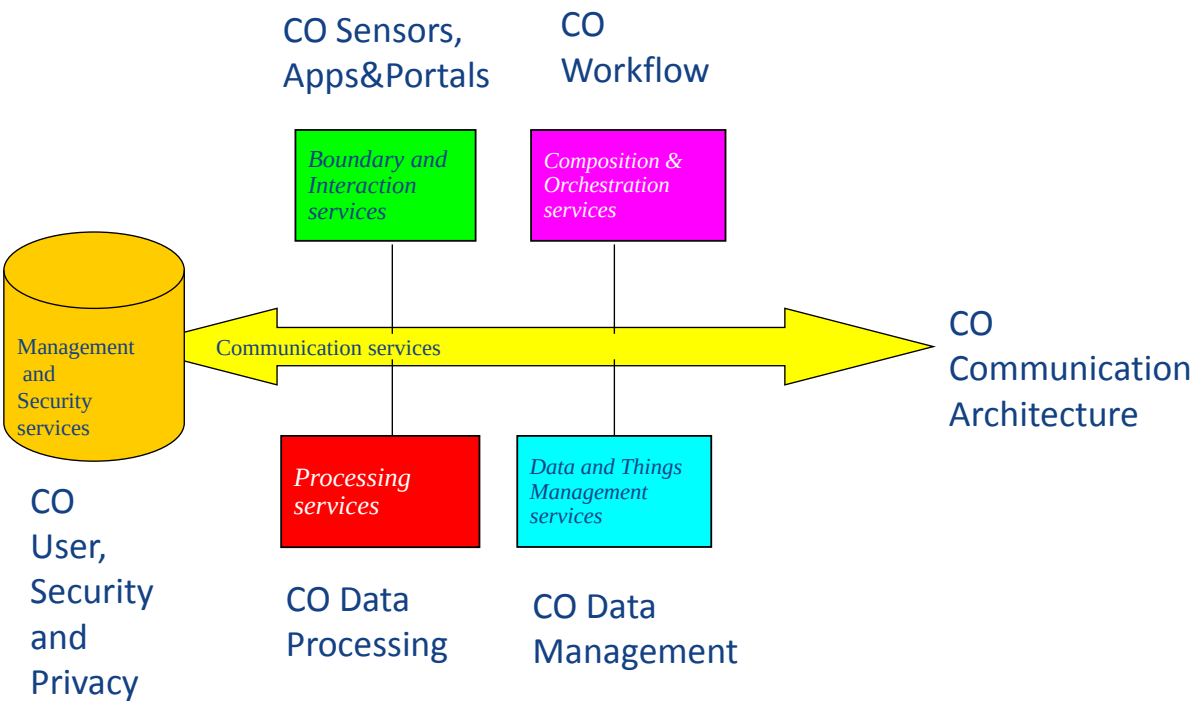

Figure 2-8 Relevant service areas for technical interoperability from ISO 19119

Figure 2-8 shows the areas of various types of technical services in a distributed architecture.
The names/types have been generalised in the new version of the ISO 19119: 2016 standard [5] to be able to support a slightly broader set of services.

Boundary Interaction Service Interoperability are interoperability among services for the management of sensors and user interfaces, graphics, multimedia and for the presentation of compound documents.

Workflow/Task Service Interoperability is interoperability among services for support of specific tasks or work-related activities conducted by humans. These services support use of resources and development of products involving a sequence of activities or steps that may be conducted by different persons.

Processing Services Interoperability is interoperability among services with large-scale computations involving substantial amounts of data..

Model/Information Management Service interoperability is interoperability among services for management of the development, manipulation and storage of metadata, conceptual schemas and datasets. The specialization of this class of services focuses on management and administration of geographic information, including conceptual schemas and data. Specific services within this class are identified in EN ISO 19119.

Communication Service interoperability is interoperability among services for encoding and transfer of data across communications networks. The specific services focus on the transfer of geographic information across a computer network.

System Management and Security Service interoperability is interoperability among services for the management of system components, applications and networks. These services also include management of user accounts and user access privileges.

The CITI-SENSE Platform and architecture has experiences various platform services within all of the six categories.

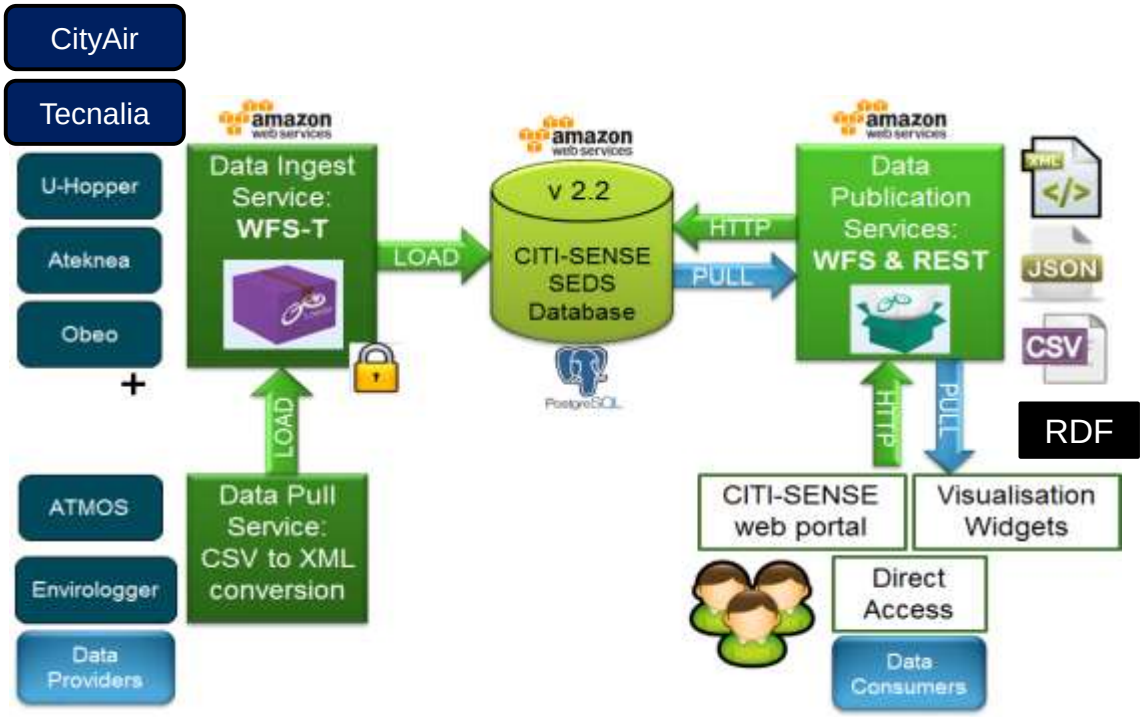

Figure 2-9 CITI-SENSE technical interoperability using WFS-T and REST interfaces

The Figure 2-9shows the technical data flow from the various sensor provider inputs to the CITI-SENSE SEDS Database server with Data ingest services and Data publication services.

Related to the technical interoperability of Citizens Observatories in general and CITI-SENSE in particular, the focus is here on supporting heterogeneity among both sensor platforms and for various user interaction services.

The model management interoperability is supported through the use of a common data model in the CITI-SENSE SEDS server.

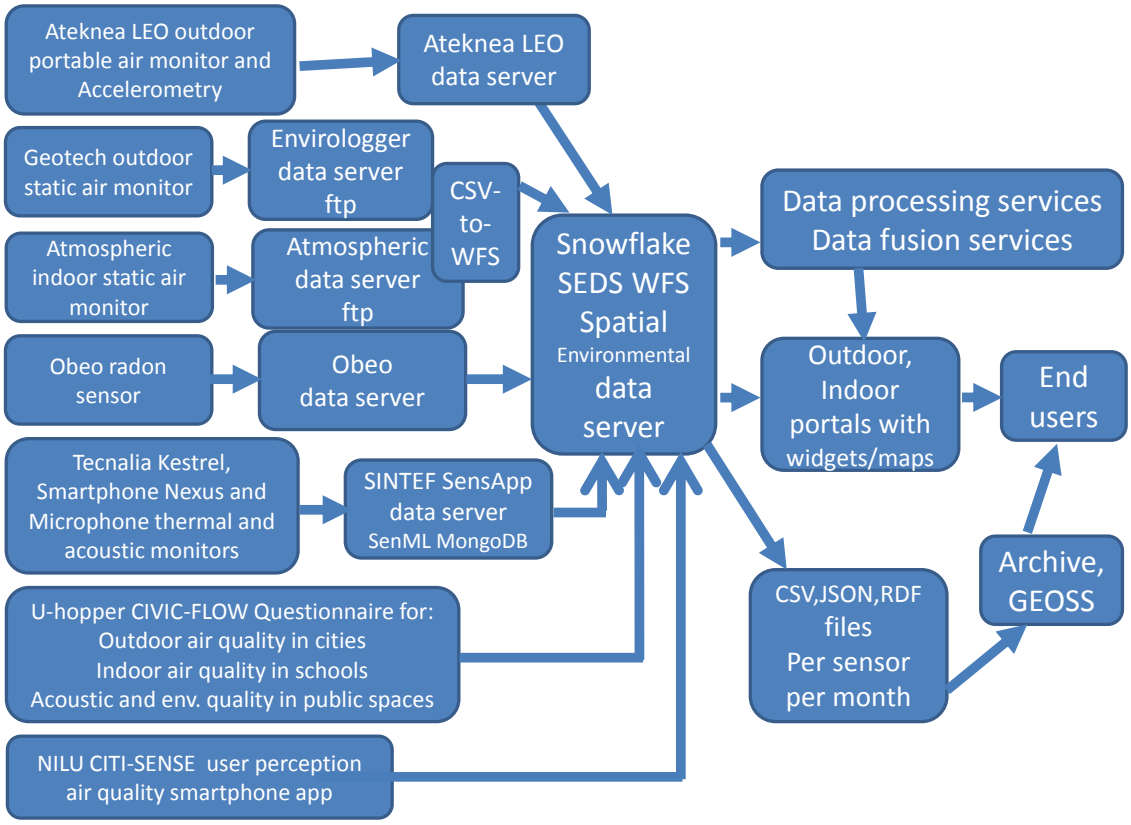

Figure 2-10 CITI-SENSE Data flow to and from the SEDS WFS server

The Figure 2-10shows the variety of different sensor types and apps on the left hand side – and how the data is mapped into a common model in the spatial environmenta data server, before being server out again also in different forms through APIs (WFS) or through various representation forms by CSV, XML, JSON, RDF or others. The data model and the data server will be described in more detail in section 3.3.

2.9 CO Non functional Interoperability

In a Citizens' Observatory it is in particular relevant and important to agree on the principles and solutions for security, access, data protection and the handling of privacy and anonymisation when necessary.

In the CITI-SENSE project data access has been protected by a security layer with user name and password protection. This has been concluded to be sufficient for the support during the life time of the CITI-SENSE project. For future Citizen Observatories it will be important to relate to the new updates of the European Data Protection Directive in 2017 that for instance will require that it shall be possible to delete/remove information that has been provided by an individual person. This functionality has not been supported in the current CITI-SENSE platform.

3 Data

3.1 An overview of Citizens' Observatory Data Management

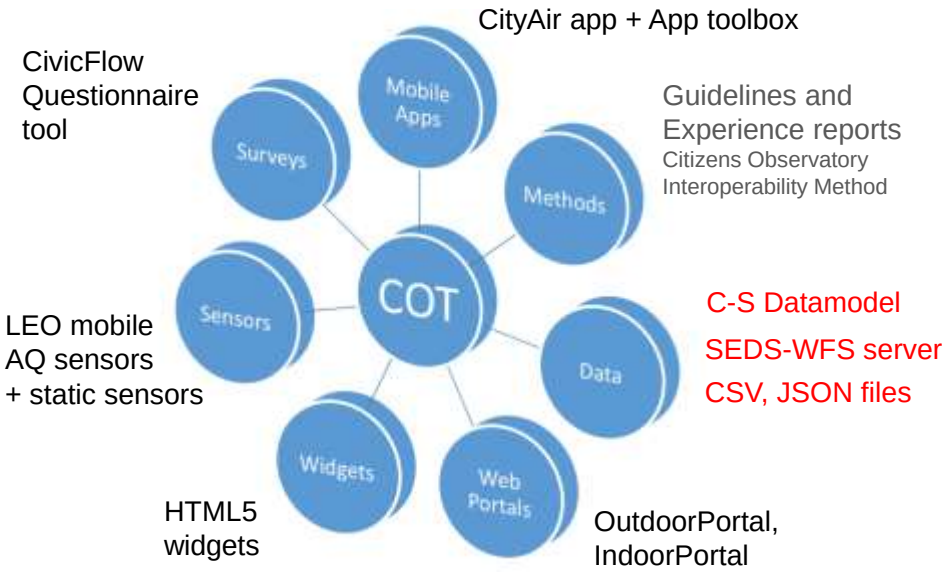

Figure 3-1 CITI-SENSE Citizens' Observatory Toolbox (COT) overview and Data focus

This chapter describes the various aspects of Citizens' Observatory Data Management. The focus here is on the final management of the CITI-SENS WFS server. The potential use of the SensApp server in particular integrated with mobile phone access as a front end before possible storate in a WFS server is described in the SensApp chapters in D7.6 part 1 and 2.

The content of the chapter is presented according to the following structure:

3.1 An overview of Citizens' Observatory Data Management
3.2 Cloud Infrastructure and AWS Setup
3.3 Data – WFS Model and database setup
3.4 Data creation and Access (WFS-T) for new sensors/apps
3.5 Data – CSV file ingestion for third party sensors
3.6 Data Queries and performance
3.7 Data – CSV archive/query structure and GEOSS

The data that has been collected in the CITI-SENSE project is at the end of the project available as a set of CSV files as described in section 3.7. Data for selected time slots and sensors at the various locations is also available through a new Map Query interface described in chapter 4.

3.2 Cloud Infrastructure and AWS Setup

Amazon Web Services

Amazon Web Services (AWS) is a collection of remote computing services that together make up a cloud computing platform, offered over the Internet by Amazon. The most central and well-known of these services are Amazon EC2, Amazon RDS and Amazon S3.

In the CITI-SENSE architecture, the Spatial and Environmental Data Service Platform is deployed on AWS.

Amazon EC2 Instances
Amazon EC2 or Amazon Elastic Compute Cloud provides scalable computing capacity in the Amazon Web Services (AWS) cloud. With Amazon EC2 there is no need to invest in hardware and developing and deploying applications can be quick and easy. Amazon EC2 provides virtual computing environments, known as instances.

Amazon Machine Images or AMIs are preconfigured templates for instances. These have certain things like the operating system and additional software pre-installed. There are various configurations of CPU, memory, storage and networking capacity (known as instance types) available for your instances. Other features for Amazon EC2 instances include security features that include login information that uses key pairs, persistent storage volumes for data using Amazon Elastic Block Store (Amazon EBS), multiple physical locations for resources known as Availability Zones, firewall to manage certain protocols, ports and IP ranges using security groups and virtual networks that can logically isolate your instances from the rest of the AWS Cloud, called Virtual Private Cloud (VPC).
(Source: Amazon AWS Documentation:
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html)

Setting up Amazon EC2 Instances
Once signed up for Amazon Web Services setting up an Amazon EC2 instance is fairly straight forward and something that is well documented. Amazon AWS have very good documentation to walk you through the step required to setup an Amazon EC2 Instance.
(Source: Amazon AWS Documentation – Setting up with Amazon EC2
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html)
Below is a summary of the key steps that are required to create an EC2 instance:

1) Sign up for Amazon AWS

A user is automatically signed up for all services in the Amazon AWS once they sign up for an Amazon AWS account. Amazon only charges the user for the services that they use.
Visit http://aws.amazon.com and then click "Create an AWS Account". Follow the instructions to complete registration.

2) Create an IAM User

Amazon don't recommend using the credentials for your AWS account when accessing AWS. Instead they recommend using the AWS Identity and Access Management (IAM). Create an IAM User and then add the user to an IAM group with administrative privileges or grant the IAM user administrative privileges. Once the IAM user is created the AWS can be accessed using a special URL and the credentials for the IAM user.

3) Create a Key Pair

AWS uses public-key cryptography to secure the login information for the EC2 instance. A Linux instance has no password, the user uses a key pair to log into the instance securely. The name of the key pair is specified when launching the instance and then the private key is provided when the users logs in using SSH.

4) Create a Virtual Private Cloud (VPC)

Amazon VPC enables the user to launch AWS resources into a virtual network in order to isolate the resources from the rest of the AWS Cloud. The default VPC was chosen for the CITI-SENSE instances.

5) Create a Security Group

Security groups act as a firewall for associated instances, controlling both inbound and outbound traffic at the instance level. Users add rules to the security group that enable the user to connect to the instance from the remote IP address using SSH. Users can add rules that allow inbound and outbound HTTP and HTTPS access from anywhere.
For the CITI-SENSE project there were 7 Amazon EC2 instances that were created. These are as shown in the image below.

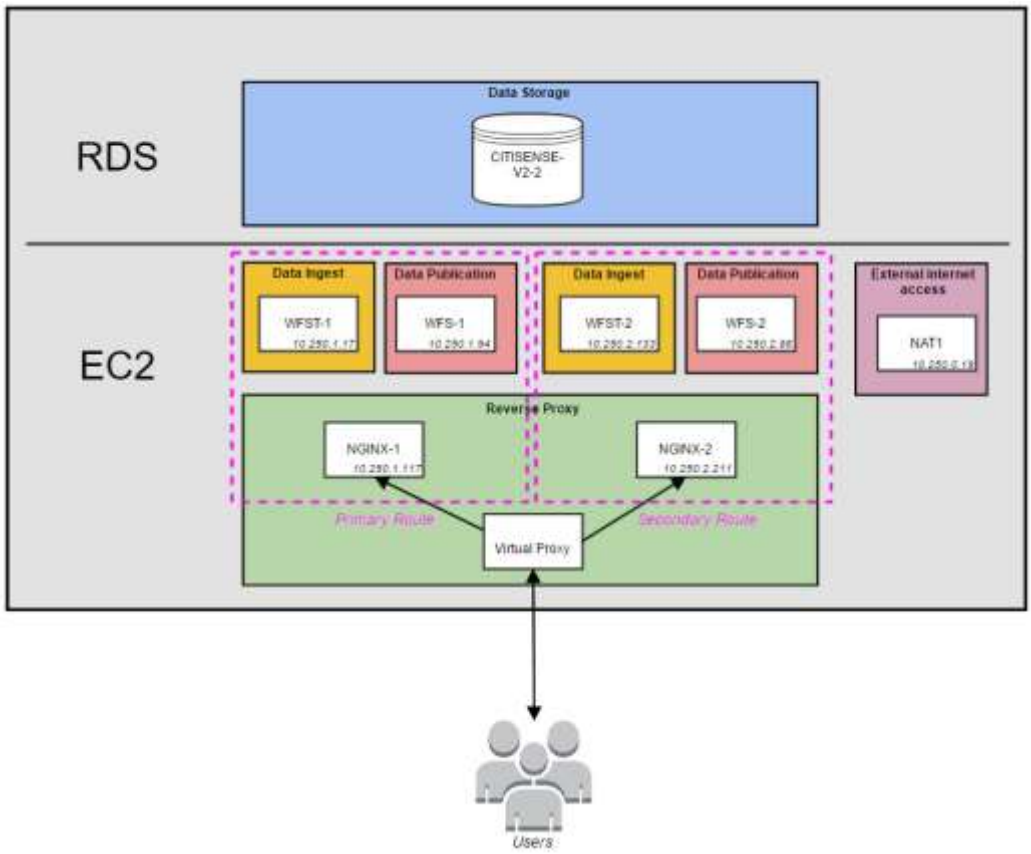

Figure 3-2 Overview of SEDS server EC2 instances

The following table describes the 7 EC2 instances:

Table 3-1 Description of EC instances

Instance Name	Description
WFS-1	Primary WFS service which provides OGC WFS and REST service endpoint to publish data in the XML, CSV and JSON encodings.

WFS-2	Secondary WFS service which provides OGC WFS and REST service endpoint to publish data in the XML, CSV and JSON encodings.
WFST-1	Primary WFS-T service which provides data ingestion services for data providers.
WFST-2	Secondary WFS-T service which provides data ingestion services for data providers.
NGINX-1	Primary NGINX service that acts as a reverse proxy server and provides load balancing functionality.
NGINX-2	Secondary NGINX service that acts as a reverse proxy server and provides load balancing functionality.
NAT1	Network Address Translation instance that allows internet access to the other EC2 instances (for added security reasons).

NAT instance or Network Address Translation instance is generally in a public subnet in the VPC to enable other instances in the private subnet to initiate outbound traffic to the internet or other AWS services, but prevents the instances from receiving inbound traffic initiated by someone on the internet.
(Source: Amazon AWS Documentation – Setting up the NAT Instance http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_NAT_Instance.html#NATInstance)
The EC2 instances were placed in two different subnets. The primary and secondary components each reside in their own subnets.

Elastic Load Balancing
AWS provides Elastic Load Balancing to automatically distribute incoming web traffic across multiple EC2 instances. It simply means that if one instance is getting too much traffic, some of it can be routed to another instance for processing. For CITI-SENSE the load balancer balances the load between the two NGINX EC2 instances. Each NGINX instance spreads the load over the data ingestion (WFST-1 and WFST-2) and data publication services (WFS-1 and WFS-2).

Setting up load balancers improves scalability. The number of EC2 instances in a load balancer can be increased or decreased as per requirement without disrupting the overall flow of information.
More information on setting up an elastic load balancer can be found on the AWS documentation here: http://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/load-balancer-getting-started.html

Availability Zones
Amazon EC2 instances are hosted in multiple locations across the world. These locations are comprised of regions and availability zones. A region is geographic area. Each region is a separate geographic area. There are multiple, isolated locations within each region known as availability zones. AWS gives you the ability to places resources, such as EC2 instances in multiple locations. Distributing EC2 instances across multiple availability zones reduces the risk of losing all the data and applications should that availability zone have failures or have downtime.

Amazon RDS
Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate and scale a relational database in the cloud. Amazon RDS allows users to create a database instance that uses MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Amazon Aurora or MariaDB as the database engine.

Database (DB) Instance:
A database (DB) instance is an isolated database environment in the cloud. A DB instance can contain multiple user-created databases and these databases can be accessed using the same client applications that are used to access stand-alone database instances. Amazon RDS offers pre-determined configurations (i.e. memory and computation capacity) for DB Instances and users can select the DB instance that meets their needs.
Amazon RDS manages backups, software patching, automatic failure detection and recovery.
Similar to EC2 instances, DB instances can be run in several Availability Zones, which is called Multi-AZ deployment. When this option is selected, Amazon automatically provisions and maintains a synchronous standby replica of your DB instance in a different Availability Zone. The primary DB instance is synchronously replicated across Availability zones to the standby replica to provide data redundancy, failover support and eliminate I/O freezes and minimise latency spikes during system backups.

For the CITI-SENSE project, the DB instance on Amazon RDS used the PostgreSQL database engine. It also had the PostGIS extension.
More information on Amazon RDS can be found on the Amazon RDS Documentation here: http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html#Welcome.Concepts

NGINX

NGINX, pronounced as "engine-x", is a free, open-source, high-performance HTTP server and reverse proxy. It has a global user base and is being implemented in numerous applications.

In the SEDS Platform NGINX is used to fulfil the role as reverse proxy server. In computer networks a reverse proxy retrieves resources on behalf of a client from one or more servers. The resources are then returned to the client as though they originated from the proxy server itself.

The main benefit of implementing a reverse proxy is that it hides the existence and characteristics of the back-end servers, thus improving ease-of-access for client applications and adds additional security to the SEDS Platform.

The NGINX product comes with additional functionality for load balancing and data caching, both of which have been used in the implementation of the SEDS Platform. NGINX is also capable of performing URL redirects which can be benefitial in creating RESTful endpoints.

3.3 Data – WFS Model and database setup

Data – WFS Model and database setup (for a new Citizens' Observatory)
On a high level, the CITI-SENSE SEDS Platform provides two main interfaces with external actors: one for Data Ingestion and one for Data Publication/Data Access. Both interfaces have been realised as a range of web services that can be accessed via standard HTTP protocols.

OGC Web Services
OGC web services will provide access to CITI-SENSE data for clients. This suite of standards is also used by GEOSS and INSPIRE and so by providing access via OGC services. CITI-SENSE data will be easy to integrate into national, European and international data sharing frameworks.

The Web Feature Service (WFS) standard web interface (ISO-19142) is implemented. This interface allows clients to query and request data. It also provides so-called "capabilities documents", which describe the service. The capabilities documents can be registered with discovery services such as the INSPIRE and GEOSS registries to make these services discoverable to the communities using those registries.

Content returned through the WFS service is in GML. GML is a standard data encoding for exchanging geographical data. Existing schemas will be used where possible and new schemas will be defined where necessary. The schema translation capabilities of the CITI-SENSE hub will be used to provide the same data through multiple service endpoints with different application schemas. This will allow harmonisation with other frameworks. For example, CITI-SENSE data can be translated into INSPIRE schemas to provide data which is harmonised with other INSPIRE datasets. It is likely that the CITI-SENSE data will be richer than the INSPIRE schema requires in some cases. In these cases an alternative services with the full attribution of the data can also be provided using an alternative schema.

Lightweight Web Services
The CITI-SENSE Platform provides a number of lightweight web services suitable for SMEs or citizen users. These are based on RESTful principles and will provide content in well supported formats including CSV and JSON.

REST stands for Representational State Transfer and is a style of software architecture. REST has the concept of linking to resources or data via a simple HTTP URL. Services that conform to the REST constraints are referred to as being "RESTful". The simplest example of how REST works can be shown by links on a webpage. If you click on a hyperlink on a webpage it can take you to another webpage where it is essentially returning the HTML code (which can be thought of as data) of the page to be displayed. These links don't have to return just a new web page however but can also take the user to other types of data such as images, XML files or ZIP files for example.
REST can be used to provide end users with a simplified and pre-packaged way of accessing data or resources. REST is a different approach to WFS for example, as where WFS is open ended because of its query language, REST allows users to be sent down a pre-defined pattern of access.

CITI-SENSE UML Model
In the CITI-SENSE UML model the following classes have been defined –with the red rectangle highlighting the new geometry property added to the Measurement class being new addition to this version 2.2.1 of the model:

Table 3-2 Classes in the CITI-SENSE UML model

Class name	Description
Cities	Pilot cities participating in CITI-SENSE (e.g., Barcelona)
SensorProvider	Details about the sensor data providers (e.g., AirBase)
Sensor	Information about the sensor that is used to make the observation (e.g., Nexus4).

Observation	Information about the observation, such as start and finish time, location and type.
Measurement	The result(s) of an observation in a specific time. E.g., temperature, pollution level and location.
Questionnaire	A specific type of observation used by the CivicFlow and CityAir applications which capture subjective data.
Question	Each Questionnaire contains one or many questions.
Response	The Response to a Question
Answer	The Answer to a Question
CAQI Value	Common Air Quality Index; provides a general overview of the air quality
Global CAQI	Global Common Air Quality Index

The full UML class diagram is pictured below:

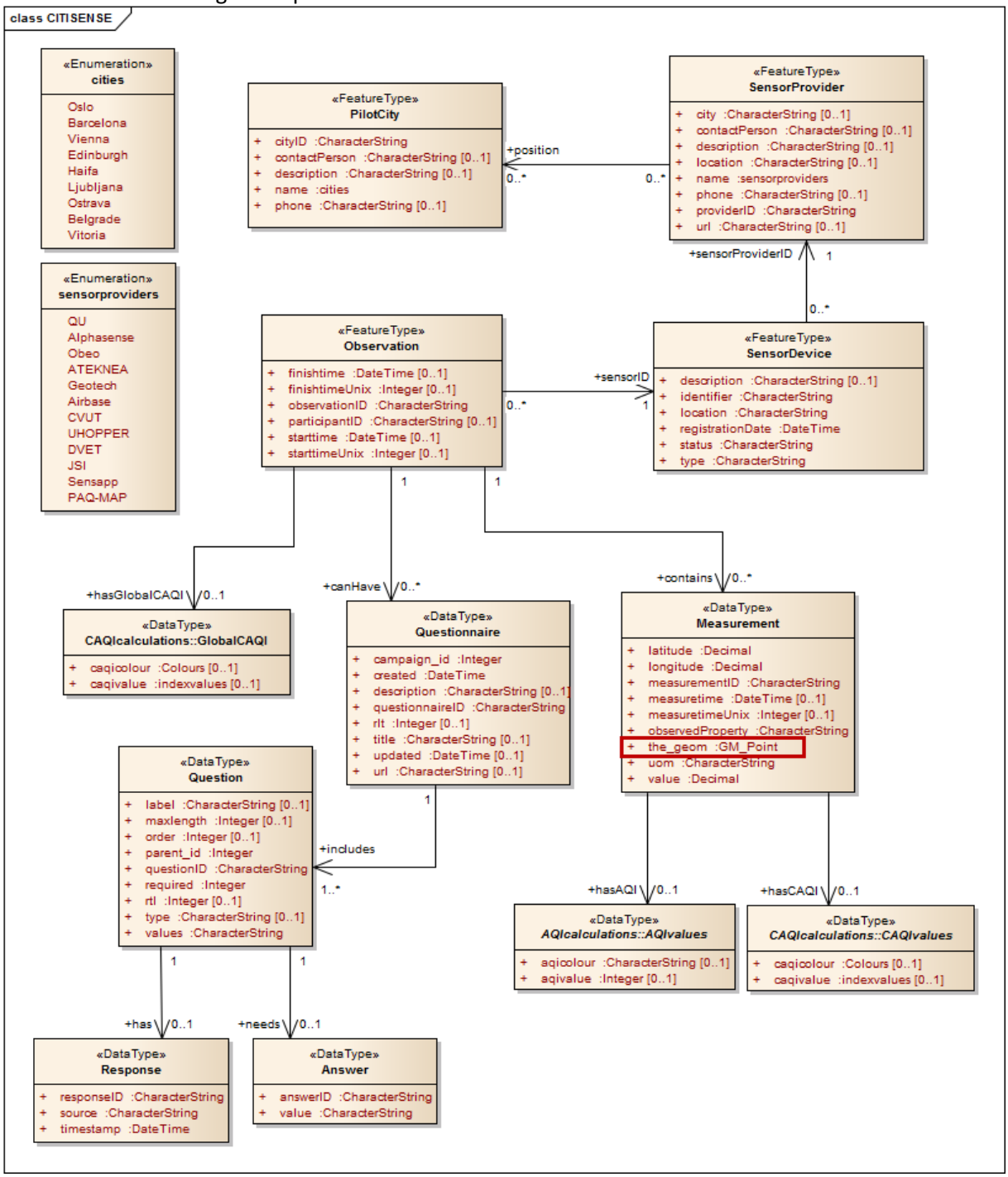

Figure 3-3 Data model UML class diagram

Data Storage

The Data Storage component in the SEDS Platform is implemented as a relational SQL database. The SQL query interface will be a common interface at the back end of the majority of the CITI-SENSE services. The SQL interface will not be available to external applications and external applications will communicate with the CITI-SENSE platform via open standard web services.

Database technology

The free and open source database technology PostgreSQL is implemented to realise the SEDS Data Repository. PostgreSQL is cross-platform and supports many different operating systems. Currently PostgreSQL is widely used across the world and underpins some large IT infrastructures.

For storing geographical data, the PostgreSQL database is extended with the PostGIS add-on. PostGIS is a free and open source software application which adds support for geographic objects to the PostgreSQL database. PostGIS follows the Simple Features for SQL specification from the Open Geospatial Consortium[1]. PostgreSQL version 9.4.5 along with PostGIS extension verison 2.1.5 was used to store and manage the SEDS backend database.

The solution adopted to create the PostgreSQL/PostGIS Relational Database has been the one provided by the Amazon Web Service. The following image shows how is connected to the different WFS services created for the communication with the actors involved in the CITISENSE project:

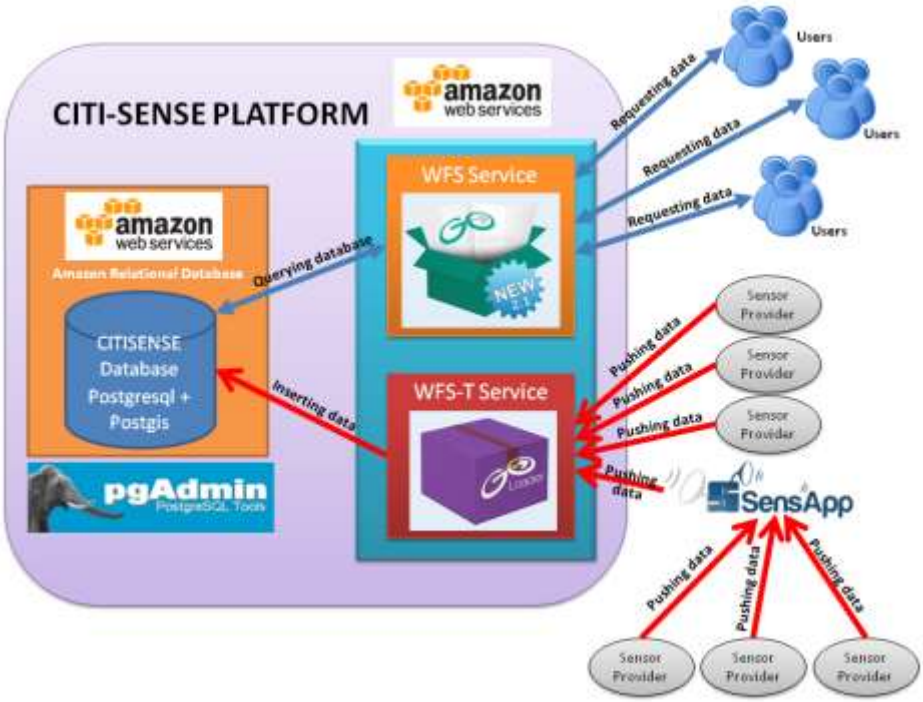

Figure 3-4 PostgreSQL/Postgis RDS connected to the WFS and WFS-T

Database creation scripts

The basic building block of the SEDS platform is a relational database. In order to create a new SEDS platform, we need a relational database (with native spatial data type). The relational database used in the CITI-SENSE project was PostgreSQL with a PostGIS extension as mentioned earlier. In theory any relational database, such as Oracle or SQL Server, can be used but the current toolkit resources will work with a PostgreSQL database.

During the CITI-SENSE project, the database was created using Snowflake Software's GO Loader product. The use of Snowflake Software's GO Loader product helped in rapid deployment of the database when any changes to the UML model were made. As GO Loader is not part of the CITI-SENSE

[1] http://www.opengeospatial.org/standards/sfs

Toolkit, database scripts have been included to allow users create the database structure needed for the SEDS platform.

The database creation scripts will create the tables needed to store the CITI-SENSE data. The database structure has been based on the CITI-SENSE application schema. The database creation scripts also create certain database triggers and functions.
There are triggers and functions which convert observed property values to micrograms per cubic metre, to calculate CAQI value and CAQI colour, convert latitude and longitude (where present) into a PostGIS point geometry and also to convert Unix time values to ISO time values. These functions and triggers are fired whenever new data gets inserted.

3.4 Data creation and Access (WFS-T) for new sensors/apps

3.4.1 Sensor Registration

A key requirement for sensor data networks to start adding their sensor data into the CITI-SENSE SEDS Platform is that they register themselves first with the SEDS Platform. This allows users of the CITI-SENSE SEDS Platform to find (discover) which sensors and sensor networks are available in the CITI-SENSE SEDS Platform and which capabilities they offer.
Sensor and sensor network providers must provide all the necessary information required for discovery.

To register a sensor, a "Sensor Registration Service" is provided by the CITI-SENSE SEDS Platform. This service provides a single entry point to formally register individual sensors, sensor networks or services. The data that is provided will form the Metadata for that sensor, sensor network or service and will be stored in the CITI-SENSE Platform centralised data repository.

The OGC SensorML standard has been identified for describing the sensors themselves. SensorML provides standard models and an XML encoding for describing sensors and measurement processes. SensorML can be used to describe a wide range of sensors, including both dynamic and stationary platforms and both in-situ and remote sensors.

3.4.2 Sensor Data Ingest

Once a sensor has been successfully registered in the CITI-SENSE SEDS Platform, it can start feeding sensor data into the SEDS Platform. The SEDS Platform stores and manages the ingested sensor data in a centralised and structured way.

Within the CITI-SENSE project various sensor data providers are active. Many of them already have processes and technology in place that exchange data from their sensor network, however quite often these processes and technology are optimised to work only within this bounded ecosystem.
The challenge is to connect these disparate federated sensor network ecosystems to the single, centralised Data Repository.

There are two main mechanisms to ingest data from the sensor data provider networks into the SEDS Data Repository:

1. **Push** - the sensor data provider network actively pushes data to the SEDS Data Repository. This can be done periodically or in (near) real-time.

2. **Pull** - the SEDS Data Repository actively pulls data from the sensor data provider network. This can be done periodically.

Which mechanism, push or pull, to use depends very much on the capabilities that the sensor data provider network can provide. In the CITI-SENSE project both mechanisms proved to be required to deploy.

Table 3-3 Use of push and pull interfaces for observation data providers

Data Provider	Push or Pull	Description
Geotech/AQMesh	Pull	Data provider provides an external FTP server on which datasets are stored. The Sensor Data Ingest service periodically harvests the data from this FTP server.
AirMonitors	Pull	Data Provider provides an external FTP server on which datasets are stored. The Sensor Data Ingest service periodically harvests the data from this FTP server.
SensApp	Push	Data Provider provides a web service that pushes data as SenML in a HTTP POST operation to the SEDS Platform.
City Air	Push	Data Provider provides a web service that pushes data as XML in a HTTP POST operation to the SEDS Platform

DNET	Push	Provides a web service (API) that publishes data as CSV or JSON.
JSI	Push	Data Provider provides a web service that pushes data as XML in a HTTP POST operation to the SEDS Platform.
CVUT	Push	Data Provider provides a web service that pushes data as JSON in a HTTP POST operation to the SEDS Platform.
U-Hopper	Push	Data Provider provides a web service that pushes data as XML in a HTTP POST operation to the SEDS Platform.
LEO/Ateknea	Push	Data Provider provides a web service that pushes data as XML in a HTTP POST operation to the SEDS Platform.

One of the key assumptions in the CITI-SENSE SEDS Platform is that it should adopt Open Standards by default. This allows the CITI-SENSE SEDS Platform to be interoperable, flexible and future-proof. For ingesting sensor data into the SEDS Platform the ISO/OGC Web Feature Standard (WFS) is used. A WFS web service endpoint has been made available that allow sensor data providers to post their datasets against.

Each sensor provider was provided with login credentials. Access to the Data Ingestion service requires a secure HTTP (HTTPS) connection. Clients can only access this service if they provide the correct credentials in their requests.

3.5 Data – CSV file ingestion for third party sensors

Geotech with AQMesh
Every hour a data ingestion web service on the SEDS Platform connects to the AQ Mesh FTP server and copies all existing CSV files to the SEDS Platform for ingestion into the database, which stores the sensor data.

The Data Ingest process consists of two components: 1) CSV Data Pull Service and 2) Data Ingest Service.
The 'CSV Data Pull Service' establishes a secure HTTPS connection to the remote AQMesh FTP server and copies all existing CSV datasets to the SEDS Platform. Here each dataset is converted to a standard HTTP POST Request that includes the sensor data as its payload. The post request is an XML file that contains the observations and measurements from the original CSV file.

This request is posted to the 'Data Ingest Service', which validates the request before ingesting the dataset into the relational SQL database, which forms the persistent data storage component.
The AQMesh Data Ingest Service also uses a Python script to update the locations of particular sensors. For certain AQMesh sensors the location contained in the incoming data is the old/wrong location for the sensor. A simple CSV file contains the updated location values for the sensors. The Python script uses this CSV file as a lookup and updates the location in the payload that is included in the HTTP Post request. The updated locations for the sensors listed in the CSV file are not applied retrospectively on historic data. These are only applied to new data.

ATMOS
Every hour a data ingestion web service on the SEDS Platform connects to the ATMOS FTP server and copies all existing CSV files to the SEDS Platform for conversion and ingestion into the database.

The Data Ingest process consists of two components: 1) CSV Data Conversion Service and 2) Data Ingest Service.

The CSV Data Conversion Service establishes a secure HTTPS connection to the remote ATMOS FTP server and copies all existing CSV files to the SEDS Platform. After the files have been successfully copied, all CSV files on the FTP server are deleted.

Once the files have been copied to the SEDS Platform, the CSV Data Conversion Service converts each file to a HTTP Post request and posts it to the Data Ingest Service. The post request is an XML file that contains the observations and measurements from the original CSV file.

The Data Ingest Service is implemented as an open standard Web Feature Service (WFS) web interface. WFS is a standard web interface specification developed and managed by the Open Geospatial Consortium (OGC) and is also an ISO standard (ISO/DIS-19142).

The HTTP Post request is posted to the endpoint of the Data Ingest Service web service:

https://prod.citisense.snowflakesoftware.com/wfst

Apply corrections to CO2 values for ATMOS sensors (Section 6 Part 2 ATMOS)
Following analysis by WP3b, it was concluded that the ATMOS sensor data required a correction for CO2 values.
A correction formula was supplied. The correction formula was:
NewCO2Value = (MeasuredCO2Value + OFFSET)/SLOPE

Each city provided the OFFSET and SLOPE values for CO2 for each node. The default value for OFFSET is set to 0 and the SLOPE to 1.

The following table lists all the sensor ids by city that have the CO2 correction factors applied. The table shows the OFFSET and SLOPE values for each sensor within each city. For all other ATMOS sensors the default values of OFFSET and SLOPE, mentioned above, are used.

Table 3-4 Sensor ids by city with CO2 correction factors

Sensor Id	Offset	Slope	City
AT_3	-29.9084	1.0293	Oslo
AT_5	-55.5349	0.82432	Oslo
AT_6	-77.1947	0.83074	Oslo
AT_7	-47.6365	0.79157	Oslo
AT_8	-12.435	0.93143	Oslo
AT_9	-39.2376	0.93565	Oslo
AT_10	-22.4759	1.1084	Oslo
AT_11	-74.3624	0.95993	Oslo
AT_13	-66.7177	0.96443	Oslo
AT_14	-78.2772	0.87574	Oslo
AT_71	-3.1727	1.0676	Ljubljana
AT_72	-19.554	1.0571	Ljubljana
AT_73	-16.666	1.0798	Ljubljana
AT_75	-27.279	1.0977	Ljubljana
AT_77	-74.216	1.1126	Ljubljana
AT_78	-41.125	1.0596	Ljubljana
AT_79	-35.177	1.0785	Ljubljana
AT_80	-10.413	1.0859	Ljubljana
AT_81	-5.8694	1.0436	Ljubljana
AT_82	-33.165	1.1321	Ljubljana
AT_83	-12.447	1.0552	Ljubljana
AT_84	92.54723	1.405964	Belgrade
AT_86	-129.359	0.944334	Belgrade
AT_92	-83.8587	0.971562	Belgrade
AT_93	50.30511	1.176108	Belgrade
AT_94	25.26433	1.102721	Belgrade
AT_96	144.8644	1.598459	Belgrade
AT_97	-95.2308	0.965916	Belgrade
AT_98	-55.2177	0.945097	Belgrade
AT_99	340.2093	1.858132	Belgrade

A simple database lookup table was created with the above OFFSET and SLOPE values along with the sensor ids. The CSV Data Conversion Service reads the lookup table and applies the CO2 correction using the formula mentioned above and then converts the incoming data into XML files that get posted to the Data Ingest Service.

The CO2 corrections were being applied to new data captured by the ATMOS sensors and were not retrospectively applied to historic data.

3.5.1 Data Ingestion Services

The data ingestion services are web services that have been deployed on Amazon AWS. Currently it contains a single WFS endpoint which data providers can use to a) register their sensors, and b) upload their sensor observations. This interface is realised by implementing an ISO 19142/OGC Web Feature Service (WFS).

The WFS endpoint is located at https://prod.citisense.snowflakesoftware.com/wfst and supports the following WFS operations:

- GetCapabilities: provides information about the functionality that the WFS supports;
- DescribeFeatureType: provides a list of feature types that the WFS offers;
- GetFeature: allows the request and response of data via WFS request.

For security reasons, access to the Data Ingestions services is prohibited for clients without the correct credentials. This protects the SEDS Platform from any unauthorised ingestion of data.

The Data Ingestion services primarily consist of a WFS web service that allows client to post data ingest requests against. However, a small number of data providers in CITI-SENSE cannot provide this functionality. Therefore, an additional component has been added to the Data Ingestion services. This component connects periodically to a secure FTP server at the data provider and copies sensor data to the SEDS Platform. Here the datasets are converted into HTTP POST requests which are made against the main WFS Data Ingestion web service.

3.6 Data Queries and performance

Two types of web interfaces have been realised on the Amazon Cloud AWS which allow users to get access to the sensor and sensor observation data stored in the SEDS Platform:

1. WFS
2. REST

Web Feature Service

A number of WFS service endpoints are available to end-users. After successfully connecting a WFS web service, an end-user can use standard WFS queries encoded using the OGC Filter Encoding Specification (FES)[2].

REST

Whilst the WFS web service provides a rich and comprehensive specification to interact with a web service, some end-users require a simpler, more lightweight specification for example to build mobile applications.

To satisfy the requirements of this user group in CITI-SENSE a number of REST web services have been deployed alongside the WFS web services. The RESTful webservices are capable of serving data formatted in either in XML or JSON.

Following the requirements of Work Packages 2 and 3, specific REST services have been implemented. Each of these web services fulfils a particular query pattern, such as the all observations for a particular sensor between two dates.

[2] OGC 09-026r1 - OpenGIS Filter Encoding 2.0 Encoding Standard, 2010, Open Geo Spatial Consortium

3.6.1 Publication Scenarios

In close cooperation with Work Package 5, a number of common query patterns (i.e. publication scenarios) have been identified and implemented. These scenarios form a logical consequence from the user specifications detailed in D7.5-Part 3.

These scenarios helped in focussing the development of a specific set of publication services that were required by data consumers.

The following scenarios have been agreed:

- **Scenario 1**: Publish all the observations for a specific sensor device.
- **Scenario 2**: Publish all observations for a specific sensor device at a specific point in time ("*snapshot*").
- **Scenario 3**: Publish all observations for a specific sensor device in a given time period.
- **Scenario 4**: Publish the latest observation for a specific sensor device.
- **Scenario 5**: Publish the latest observation for a specific sensor device for a specific pollutant.

All these scenarios have been implemented and tested for the WFS and the REST web services.

A sixth publication scenario was later agreed for the WFS service. This was achieved by creating a new WFS endpoint. This scenario was particularly useful for the widgets in visualising, on a map, the latest observation for all the sensors within a particular city. The endpoint can be queried spatially or using other filters such as by the city name. The gml:name element was used to publish the location names and could be queried using a simple PropertyIsLike filter (see section 2.2.2 below). The bounding box spatial filter could query the observations based on the spatial location. The spatial filter is limited to the observations that contain a geometry in the database.

- **Scenario 6:** Publish the latest observation for all the sensors devices for a particular location.

3.6.2 Overview Data Publication services

The tables below provide samples of the REST and WFS request patterns for the 5 scenarios defined in the previous paragraph.

Note: Replace {sensor-id} with an actual sensor id.
REST services

Scenario 1: Give me all observations for a specific sensor-id
CSV:
https://prod.citisense.snowflakesoftware.com/csv/sensor/allobservations?sensorid={sensor-id}
JSON:
https://prod.citisense.snowflakesoftware.com/json/sensor/allobservations?sensorid={sensor-id}

Scenario 2: Give me all observations for a specific sensor-id at a specific point in time (time instant)
CSV:
https://prod.citisense.snowflakesoftware.com/csv/sensor/observationfinishtime?sensorid={sensor-id}&finishtime=2001-12-18T10:00:00.000
JSON:
https://prod.citisense.snowflakesoftware.com/json/sensor/observationfinishtime?sensorid={sensor-id}&finishtime=2001-12-18T10:00:00.000

Scenario 3: Give me all observations for a specific sensor-id in a specific time period
CSV:
https://prod.citisense.snowflakesoftware.com/csv/sensor/observationfinishtime/between?sensorid={sensor-id}&from=2001-12-17T09:00.000&to=2001-12-18T09:00:00.000
JSON:
https://tprod.citisense.snowflakesoftware.com/json/sensor/observationfinishtime/between?sensorid={sensor-id}&from=2001-12-17T09:00.000&to=2001-12-18T09:00:00.000

Scenario 4: Give me the last observation for a specific sensor-id
CSV:
https://prod.citisense.snowflakesoftware.com/csv/sensor/lastobservation?sensorid={sensor-id}
JSON:
https://prod.citisense.snowflakesoftware.com/json/sensor/lastobservation?sensorid={sensor-id}

Scenario 5: Give me the last observation for a specific sensor-id for a specific observed property (eg. pollutant, temperature, etc.)
CSV:
https://prod.citisense.snowflakesoftware.com/csv/sensor/lastobservation/observedproperty?sensorid={sensor-id}&observedproperty=NO2
JSON:
https://prod.citisense.snowflakesoftware.com/json/sensor/lastobservation/observedproperty?sensorid={sensor-id}&observedproperty=NO2

WFS service

There are two distinct end points for the WFS:

WFS End Points	Purpose
https://prod.citisense.snowflakesoftware.com/wfs	General, all purpose, WFS endpoint
https://prod.citisense.snowflakesoftware.com/wfslr	Endpoint specifically for accessing the latest observation.

Scenario 1: Give me all observations for a specific sensor-id
XML
https://prod.citisense.snowflakesoftware.com/wfs?service=wfs& version=2.0.0& request=GetFeature& typename=cts:Observation& filter= <Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http:www.citi-sense.eu/citisense"> <PropertyIsEqualTo> <ValueReference>cts:sensorID/@xlink:href</ValueReference> <Literal>{sensor-id}</Literal> </PropertyIsEqualTo> </Filter>

Scenario 2: Give me all observations for a specific sensor-id at a specific point in time (time instant)
XML
https://prod.citisense.snowflakesoftware.com/wfs?service=wfs &version=2.0.0& request=GetFeature& typename=cts:Observation& filter= <Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http:www.citi-sense.eu/citisense"> <And> <PropertyIsEqualTo> <ValueReference>cts:sensorID/@xlink:href</ValueReference> <Literal>{sensor-id}</Literal> </PropertyIsEqualTo> <PropertyIsEqualTo> <ValueReference>//cts:finishtime</ValueReference> <Literal>2001-12-18T10:30:47.000</Literal> </PropertyIsEqualTo> </And> </Filter>

Scenario 3: Give me all observations for a specific sensor-id in a specific time period
XML

Scenario 3: Give me all observations for a specific sensor-id in a specific time period

```
https://prod.citisense.snowflakesoftware.com/wfs?service=wfs&
version=2.0.0&request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http:www.citi-sense.eu/citisense">
<And>
<PropertyIsEqualTo>
<ValueReference>cts:sensorID/@xlink:href</ValueReference>
<Literal>{sensor-id}</Literal>
</PropertyIsEqualTo>
<PropertyIsBetween>
<ValueReference>//cts:finishtime</ValueReference>
<LowerBoundary>
<Literal>2001-12-17T09:00:00.000</Literal>
</LowerBoundary>
<UpperBoundary>
<Literal>2001-12-18T09:00:00.000</Literal>
</UpperBoundary>
</PropertyIsBetween>
</And>
</Filter>
```

Scenario 4: Give me the last observation for a specific sensor-id

XML

```
https://prod.citisense.snowflakesoftware.com/wfslr?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http:www.citi-sense.eu/citisense">
<PropertyIsEqualTo>
<ValueReference>cts:sensorID/@xlink:href</ValueReference>
<Literal>{sensor-id}</Literal>
</PropertyIsEqualTo>
</Filter>
```

Scenario 5: Give me the last observation for a specific sensor-id for a specific observed property (eg. pollutant, temperature, etc.)

XML

```
https://prod.citisense.snowflakesoftware.com/wfslr?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http:www.citi-sense.eu/citisense">
<And>
<PropertyIsEqualTo>
```

Scenario 5: Give me the last observation for a specific sensor-id for a specific observed property (eg. pollutant, temperature, etc.)

<ValueReference>cts:sensorID/@xlink:href</ValueReference>
<Literal>{sensor-id}</Literal>
</PropertyIsEqualTo>
<PropertyIsEqualTo>
<ValueReference>//cts:observedProperty</ValueReference>
<Literal>NO2</Literal>
</PropertyIsEqualTo>
</And>
</Filter>

Scenario 6: Give me the last observation for all sensors for a particular location by name

XML

https://prod.citisense.snowflakesoftware.com/wfslr_location?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:cts="http:www.citi-sense.eu/citisense">
<Or>
<PropertyIsLike escape="\" singleChar="_" wildCard="*">
<ValueReference>gml:name</ValueReference>
<Literal>*oslo*</Literal>
</PropertyIsLike>
<PropertyIsLike escape="\" singleChar="_" wildCard="*">
<ValueReference>gml:name</ValueReference>
<Literal>*Oslo*</Literal>
</PropertyIsLike>
<PropertyIsLike escape="\" singleChar="_" wildCard="*">
<ValueReference>gml:name</ValueReference>
<Literal>*OSLO*</Literal>
</PropertyIsLike>
</Or>
</Filter>

Scenario 6: Give me the last observation for all sensors for a particular location using a spatial bounding box filter

XML

https://prod.citisense.snowflakesoftware.com/wfslr_location?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http:www.citi-sense.eu/citisense"
xmlns:gml="http://www.opengis.net/gml">
<And>

Scenario 6: Give me the last observation for all sensors for a particular location using a spatial bounding box filter

```
<BBOX>
<ValueReference>cts:the_geom</ValueReference>
<gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
<gml:lowerCorner>32.72 34.92</gml:lowerCorner>
<gml:upperCorner>32.85 35.06</gml:upperCorner>
</gml:Envelope>
</BBOX>
</And>
</Filter>
```

Scenario 6: Give me the last observation for all sensors for a particular location using a logical bounding box filter

XML

```
https://prod.citisense.snowflakesoftware.com/wfslr_location?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http:www.citi-sense.eu/citisense"
xmlns:gml="http://www.opengis.net/gml">
<And>
<And>
<PropertyIsGreaterThanOrEqualTo>
<ValueReference>//cts:latitude</ValueReference>
<Literal>32.72</Literal>
</PropertyIsGreaterThanOrEqualTo>
<PropertyIsLessThanOrEqualTo>
<ValueReference>//cts:latitude</ValueReference>
<Literal>32.85</Literal>
</PropertyIsLessThanOrEqualTo>
</And>
<And>
<PropertyIsGreaterThanOrEqualTo>
<ValueReference>//cts:longitude</ValueReference>
<Literal>34.92</Literal>
</PropertyIsGreaterThanOrEqualTo>
<PropertyIsLessThanOrEqualTo>
<ValueReference>//cts:longitude</ValueReference>
<Literal>35.06</Literal>
</PropertyIsLessThanOrEqualTo>
</And>
</And>
</Filter>
```

3.7 Data – CSV archive/query structure and GEOSS

The following describes the CSV file structure that has been generated as part of making the CITI-SENSE Observations available for analysis by external tools – as well as for ingestion into other database systems. Monthly database dumps were exported as CSV files and made available via an FTP server

hosted by NILU. The final data files are retained in this FTP server archive, available at the ftp address: ftp://zardoz.nilu.no

The CSV files can be used in a number of ways that could include analysing the data or importing the data into another database. The format of the CSV files are described in the following.

3.7.1 Measurement CSV

Table 3-5 Measurement CSV

Variable	Meanings
measurementid	unique measurement identification
fkey_observation	Foreign key observation (please explain more)
value	Value of the measurement
Uom	Unit of measurement e.g ug/m3 , % , C
observedproperty	It explains the type of measurement if it is measuring temperature or humidity or a type of gas
latitude	Latitude is used together with longitude to specify the precise location of features on the surface of the Earth.
longitude	Longitutde is used together with latitude to specify the precise location of features on the surface of the Earth.
measuretime	Time the measurement is recorded (UTC format)
measuretime_unix	unix measurement time in Epoch. Unix time (also known as POSIX time or Epoch time) is a system for describing instants in time, defined as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970
ug_m3	Micrograms per Cubic Meter of Air
caqi_value	Common air quality index calculation value
caqi_colour	Common air quality index calculation colour
the_geom	Geometry for relative position

Table 3-6 Measurement CSV Example values

measurementid	fkey_observation	value	uom	observedproperty	latitude	longitude	measuretime
2297430075	2288380008	-22.371	ppb	NO	32.7931	35.00967	3/24/2016 7:08
2297430360	2288380039	-37.851	ppb	NO	32.7931	35.00967	3/24/2016 7:39
2297430532	2288380057	640	C	Noise Level	32.7931	35.00967	3/24/2016 7:57
2297430533	2288380057	698	%	Noise Peak	32.7931	35.00967	3/24/2016 7:57
2297430534	2288380058	-42.483	ppb	NO	32.7931	35.00967	3/24/2016 7:58
2297433154	2288383016	-1	C	Noise Level	32.7942	35.01246	3/24/2016 7:16
2297433155	2288383016	0	%	Noise Peak	32.7942	35.01246	3/24/2016 7:16
2297433156	2288383017	-58.175	ppb	NO	32.7942	35.01246	3/24/2016 7:17
2297433271	2288383029	686	C	Noise Level	32.7942	35.01246	3/24/2016 7:29
2296656414	2287608045	-22.718	ppb	NO	32.7897	35.01862	3/24/2016 6:45
2296656471	2287608051	-17.42	ppb	NO	32.7897	35.01862	3/24/2016 6:51
2296661046	2287613004	697	C	Noise Level	32.7874	35.02139	3/24/2016 6:04
2296661047	2287613004	763	%	Noise Peak	32.7874	35.02139	3/24/2016 6:04
2296661048	2287613005	-15.019	ppb	NO	32.7874	35.02139	3/24/2016 6:05
2296664009	2287616000	0.826	C	Total Count	55.9447	-3.20391	3/24/2016 6:00
2296706008	2287658000	23	%	Humidity	44.758	20.60201	3/24/2016 9:07
2296718002	2287670000	802	ppm	NO	44.7326	20.57328	3/24/2016 9:09
2296737002	2287689000	834	ppm	NO	44.758	20.60184	3/24/2016 9:11
2296742002	2287694000	814	ppm	NO	44.7816	20.47513	3/24/2016 9:11
2296753002	2287705000	817	ppm	NO	44.8	20.39553	3/24/2016 9:12
2296769002	2287721000	828	ppm	NO	44.8631	20.47064	3/24/2016 9:14
2296847001	2287799000	745	ppm	CO2	44.8	20.39549	3/24/2016 9:21
2296847002	2287799000	805	ppm	NO	44.8	20.39549	3/24/2016 9:21
2296873008	2287825000	43	%	Humidity	44.7865	20.52176	3/24/2016 9:24

3.7.2 Observation CSV

Table 3-7 Observation CSV

Variable	Meanings
observationid	Unique observation identification
sensorid_feature_id	Tells us about vendor and id of sensor
participantid	Id of individual participant
starttime	Start time when observation started
finishtime	Time when observation was finished
starttime_unix	Start time of observation in unix
finishtime_unix	End time for observation in unix

Table 3-8 Observation CSV Example values

observationid	sensorid_feature_id	participantid	starttime	finishtime	starttime_unix	finishtime_unix
1842002006	AT_73		3/1/2016 0:02	3/1/2016 0:02		
1843645004	AT_8		3/1/2016 0:02	3/1/2016 0:02		
1839918000	3.55255E+14	test_testID1	3/1/2016 0:02	3/1/2016 0:02		
1839930000	CITISENSE-OBEO-NORBGO28339		2/29/2016 23:03	3/1/2016 0:03		
1840604000	LEO-666B9D4F	LEOTEST255	3/1/2016 0:03	3/1/2016 0:03	1456790580	1456790580
1845047000	CITISENSE-OBEO-NOROSL28545		2/29/2016 23:03	3/1/2016 0:03		
1841844004	AT_33		3/1/2016 0:03	3/1/2016 0:03		
1843890005	AT_99		3/1/2016 0:03	3/1/2016 0:03		
1851208008	AT_2		3/1/2016 0:03	3/1/2016 0:03		
1841904004	AT_5		3/1/2016 0:03	3/1/2016 0:03		
1839919000	3.55255E+14	test_testID1	3/1/2016 0:03	3/1/2016 0:03		
1843607004	AT_89		3/1/2016 0:03	3/1/2016 0:03		
1842108010	AT_81		3/1/2016 0:03	3/1/2016 0:03		
1843785007	AT_96		3/1/2016 0:03	3/1/2016 0:03		
1839920000	3.55255E+14	test_testID1	3/1/2016 0:03	3/1/2016 0:03		
1839922000	3.55255E+14	test_testID1	3/1/2016 0:03	3/1/2016 0:03		
1839924000	3.55255E+14	test_testID1	3/1/2016 0:03	3/1/2016 0:03		
1839921000	3.55255E+14	test_testID1	3/1/2016 0:03	3/1/2016 0:03		
1839925000	3.55255E+14	test_testID1	3/1/2016 0:03	3/1/2016 0:03		
1839923000	3.55255E+14	test_testID1	3/1/2016 0:03	3/1/2016 0:03		
1839926000	3.55255E+14	test_testID1	3/1/2016 0:03	3/1/2016 0:03		
1842240003	AT_85		3/1/2016 0:03	3/1/2016 0:03		
1840613000	LEO-666B9D4F	LEOTEST255	3/1/2016 0:04	3/1/2016 0:04	1456790640	1456790640
1845055000	CITISENSE-OBEO-NOROSL56039		2/29/2016 23:04	3/1/2016 0:04		

3.7.3 Pilotcity

Table 3-9 Pilotcity CSV

Variable	Meanings
cityid	Identification no. for all cities
fkey_sensorprovider	blank
pilotcity_feature_id	Blank
name	Name of the city corresponding to city id
contactperson	Blank
Phone	Blank
Description	Blank

Table 3-10 Pilotcity CSV Example values

cityid	fkey_sensorprovider	pilotcity_feature_id	name	contact	phone	description
1			Oslo			
2			Barcelona			
3			Vienna			
4			Edinburgh			
5			Haifa			
6			Ljubiana			
7			Ostrava			
8			Belgrade			
9			Vitoria			

3.7.4 Question CSV

Table 3-11 Question CSV

Variable	Meanings
questionid	Unique identification for each question
fkey_questionnaire	Foreign key value for each questionnaire
parent_id	Reference to parent question
question_order	Blank
Type	Blank
Label	Question label
maxlength	Blank
question_values	Blank
Required	null
Rtl	blank

3.7.5 Questionnaire CSV

Table 3-12 Questionnaire CSV

Variable	Meanings
questionnaireid	Unique identification for each questionnaire
fkey_observation	Foreign key observation
campaign_id	null
Created	Date and time of creation
Description	Blank
Updated	Blank
Rlt	Blank
Title	Blank
url	Blank

Table 3-13 Questionnaire CSV example values

questionnaireid	fkey_observation	campaign_id	created	description	updated	rlt	title	url
1	57104001	0	9/15/2015 10:57					
1001	57256001	0	9/15/2015 12:19					
2001	59075001	0	9/16/2015 8:19					
3001	60177001	0	9/17/2015 8:25					
4001	60180001	0	9/17/2015 8:29					
5001	60186001	0	9/17/2015 8:33					
6001	60542001	0	9/17/2015 12:22					
7001	60543001	0	9/17/2015 12:22					
8001	61575001	0	9/18/2015 6:52					
9001	61576001	0	9/18/2015 6:54					
10001	61614001	0	9/18/2015 7:11					
11001	61649001	0	9/18/2015 7:46					
12001	61832001	0	9/18/2015 8:57					
13001	61928001	0	9/18/2015 10:50					
14001	62167001	0	9/18/2015 13:09					
15001	62357001	0	9/18/2015 15:17					
16001	62358001	0	9/18/2015 15:17					
17001	62631001	0	9/18/2015 13:30					
18001	66602001	0	9/20/2015 16:24					

3.7.6 Response CSV

Table 3-14 Response CSV

Variable	Meanings
Responseid	Unique identification for each response
fkey_question	Foreign key of question
Source	Null
Timestamp	Date and time of response

3.7.7 SensorDevice CSV

Table 3-15 SensorDevice CSV

Variable	Meanings
identifier	Tells us about vendor and id of sensor
sensorprovider_feature_id	Unique identification for sensor provider
Description	Which type of properties it measure e.g which gases , is it perception
Location	In which city is it located
Registrationdate	The date sensor was registered
Status	Is it active or inactive
Type	Is the sensor static or mobile

Table 3-16 SensorDevice CSV Example values

identifier	sensorprovider_feature_id	description	location	registrationdate	status	type
LEO-666BA0BF	4	NO2,O3,NO,TEMP,RH,ACTIVITY_INDEX	BARCELONA_TEST	10/16/2015 8:55	active	mobile
LEO-666BA0BE	4	NO2,O3,NO,TEMP,RH,ACTIVITY_INDEX	NONE	10/16/2015 8:55	active	mobile
AQAP_6c620f63-7888-48cf-d0d2-a175b0dcc3c2	1001	CityAir Perception sensor	Skopje, Macedonia (FYR	11/10/2015 21:17	active	mobile
CITISENSE-CIVICFLOW-TEST	8	Perception		6/14/2015 14:53	active	
CITISENSE-test44	8	Perception	CivicFlow	6/14/2015 14:53	active	static
LEO-666B9049	4	NO2,O3,NO,TEMP,RH,ACTIVITY_INDEX	BELGRADE	10/27/2015 10:14	active	mobile
AQAP_a5c8313f-f4ea-4202-e3de-a7ec1c2d93bd	1001	CityAir Perception sensor	Å kofja Loka, Slovenia	10/27/2015 20:49	active	mobile
CITISENSE-NiluTest-849150		NO,NO2,CO,O3	Oslo	24:10.5	inactive	static
AQAP_75699501-e1a6-4343-9763-cf917557c692	1001	CityAir Perception sensor	Serbia	11/11/2015 17:42	active	mobile
CITISENSE-NiluTest-855150		NO,NO2,CO,O3	Oslo	28:47.6	inactive	static
AQAP_8602b325-1d6a-4a13-a2b3-72119870096d	1001	CityAir Perception sensor	City of Belgrade, Serbia	12/3/2015 10:18	active	mobile
LEO-666B9035	4	NO2,O3,NO,TEMP,RH,ACTIVITY_INDEX	VIENNA	10/19/2015 9:16	active	mobile
LEO-666B902E	4	NO2,O3,NO,TEMP,RH,ACTIVITY_INDEX	OSLO	10/16/2015 10:27	active	mobile
AQAP_98cf5bdb-582e-4c6b-a079-43374013f9f0	1001	CityAir Perception sensor	Norway	11/19/2015 12:37	active	mobile
AQAP_fd7ab794-1e26-4309-ef8f-98ebcbeb797a	1001	CityAir Perception sensor	28054 Madrid, Madrid, S	12/4/2015 21:43	active	mobile
AQAP_da85f8c8-ec7f-48bc-d374-10a869bd9ee9	QU	Questionnaire 001	unknown	1/22/2016 3:01	active	static
CITISENSE-NiluTest-861150		NO,NO2,CO,O3	Oslo	33:15.8	inactive	static
CITISENSE-NiluTest-785150		NO,NO2,CO,O3	Oslo	09:31.2	inactive	static
AQAP_d3b979d1-342c-4bfa-c380-0ec16832b111	1001	CityAir Perception sensor	Serbia	11/4/2015 14:55	active	mobile
CITISENSE-NiluTest-Statens_vegvesen_665		Hjortnes	Oslo	29:53.3	active	static
LEO-666B9878	4	NO2,O3,NO,TEMP,RH,ACTIVITY_INDEX	BARCELONA	11/9/2015 11:10	active	mobile
CITISENSE-NiluTest-Statens_vegvesen_11		Manglerud	Oslo	31:45.1	active	static
AQAP_3c210af1-e4f7-4a52-f466-3c7e74bdee74	QU	Questionnaire 001	unknown	11/9/2015 9:53	active	static
CITISENSE-NiluTest-Oslo_kommune_12		Sk?yen	Oslo	31:52.2	active	static
CITISENSE-NiluTest-Oslo_kommune_303		Sofienbergparken	Oslo	31:54.1	active	static
CITISENSE-NiluTest-Oslo_kommune_809		?kebergveien	Oslo	31:57.2	active	static
AQAP_9db5dc96-dacc-43f0-9fb3-effcb741d091	1001	CityAir Perception sensor	Kjeller	2/23/2016 9:28	active	mobile
CITISENSE-NiluTest-712150		NO,NO2,CO,O3	Oslo	42:26.2	inactive	static
AQAP_ff88ada0-de9a-4ca7-95b4-7ce86f8e71dd	1001	CityAir Perception sensor	unknown	11/9/2015 18:46	active	mobile
CITISENSE-NiluTest-863150		NO,NO2,CO,O3	Oslo	37:06.3	inactive	static
AQAP_5e10d581-ef52-43b0-ce92-ad597318bd82	QU	Questionnaire 001	unknown	11/16/2015 13:36	active	static
LEO-666937AC	4	NO2,O3,NO,TEMP,RH,ACTIVITY_INDEX	VIENNA	10/21/2015 14:50	active	mobile
LEO-666BA0B7	4	NO2,O3,NO,TEMP,RH,ACTIVITY_INDEX	OSLO	10/16/2015 9:48	active	mobile
CITISENSE-OBEO-NOROSL56039	3	BQ, C	Oslo	2/23/2016 11:42	active	static

3.7.8 SensorProvider CSV

Table 3-17 SensorProvider CSV

Variable	Meanings
providerid	Unique identification of sensor provider
Name	Name of the company providing sensor corresponding to identification no.
url	Blank
City	Blank
Location	Blank
Contactperson	Blank
Description	Detail about the provider
Phone	Blank
providerid_character	Unique identification of sensor provider
Character	blank

3.8 Availability of CITI-SENSE Data

The following describes the CSV file structure that has been generated as part of making the CITI-SENSE Observations available for further use.

ADR: ftp://zardoz.nilu.no/
USER: citi-sense
PWD: <provided on request>

FTP root at zardoz.nilu.no

To view this FTP site in File Explorer: press Alt, click **View**, and then click **Open FTP Site in File Explorer**.

```
07/26/2016  03:56PM       Directory  April2016Export
09/02/2016  09:27AM       Directory  August2016Export
07/26/2016  03:23PM       Directory  December2015Export
07/26/2016  03:25PM       Directory  February2016Export
11/14/2016  09:21AM       Directory  FinalCITISENSEDatabaseDump
07/26/2016  03:24PM       Directory  January2016Export
08/02/2016  03:55PM       Directory  July2016Export
07/11/2016  04:32PM       Directory  June2016Export
07/26/2016  03:54PM       Directory  March2016Export
07/26/2016  03:57PM       Directory  May2016Export
07/26/2016  11:08AM       Directory  November2015Export
07/26/2016  10:45AM       Directory  October2015Export
11/01/2016  12:25PM       Directory  PerceptionsCityAirsep15nov16
10/07/2016  10:04AM       Directory  September2016Export
```

Figure 3-5 Overview of available FTP file directories per month

Table 3-18 Statistics of registered data

Sensors registered in July 2016 per provider

Sensor Provider	Number of Sensors
QU	4
PAQ-MAP	64

Sensors registered in July 2016 per city

City	Number of Sensors
Oslo	59
Others	9

Sensors registered since October 2015 per provider

Sensor Provider	Number of Sensors
QU	5
DVET	9
Obeo	13
Geotech	26
JSI	38
Ateknea	86
QU	105
PAQ-MAP	702

City	Number of Sensors
Haifa	17
Ostrava	18
Edinburgh	20
Belgrade	45
Vienna	47
Barcelona	62
Ljubljana	71
Oslo	195
Others	222
Unknown	287

Observations registered in July 2016 per provider

Sensor Provider	Number of Sensors
PAQ-MAP	38
Obeo	1412
QU	21130
Ateknea	63064
DVET	125624
Unknown	153869
Alphasense	198107
Geotech	348539

Observations registered since October 2015 per provider

Sensor Provider	Number of Sensors
JSI	311
PAQ-MAP	2895
Obeo	10012
QU	78725
DVET	589199
Unknown	688694
Ateknea	1357187
Alphasense	3712540
Geotech	4930418

Observations registered in July 2016 per city

City	Number of Observations
Ostrava	1
Vienna	1615
Barcelona	15917
Edinburgh	34169
Haifa	80334
Ljubljana	88173
Belgrade	97509
Unknown	153869
Others	182341
Oslo	207826

Observations registered since October 2015 per city

City	Number of Observations
Vienna	21038
Ostrava	434990
Unknown	688944
Barcelona	770437
Others	997274
Edinburgh	1078226
Ljubljana	1180329
Belgrade	1259890
Oslo	2303988
Haifa	2638422

Measurements registered in July 2016 per provider

Sensor Provider	Number of Sensors
PAQ-MAP	38
Obeo	2824
QU	190170
Ateknea	378384
DVET	1130616
Unknown	1384821
Geotech	2254975
Alphasense	2377284

Measurements registered since October 2015 per provider

Sensor Provider	Number of Sensors
JSI	311
PAQ-MAP	2895
Obeo	20024
QU	708584
DVET	5303167
Unknown	6198858
Ateknea	8097787
Geotech	39090763
Alphasense	44550480

3.9 CITI-SENSE Data available after the end of the project

Since the CITI-SENSE SEDS server has been running on an optimised AWS configuration supporting realtime access and queries with a relatively high monthly cost, there has been a change in the setup after the end of the CITI-SENSE project where none of the sensor platforms will continue to be in operation and continuoulsy provide data in the same way as during the project lifetime.

All of the collected data has now been preserved through monthly files according to the descriptions in the previous sections, and also a full dump of the database has been made.

These data can be accessed through a provided FTP site user name and password, and further manipulated in different ways.

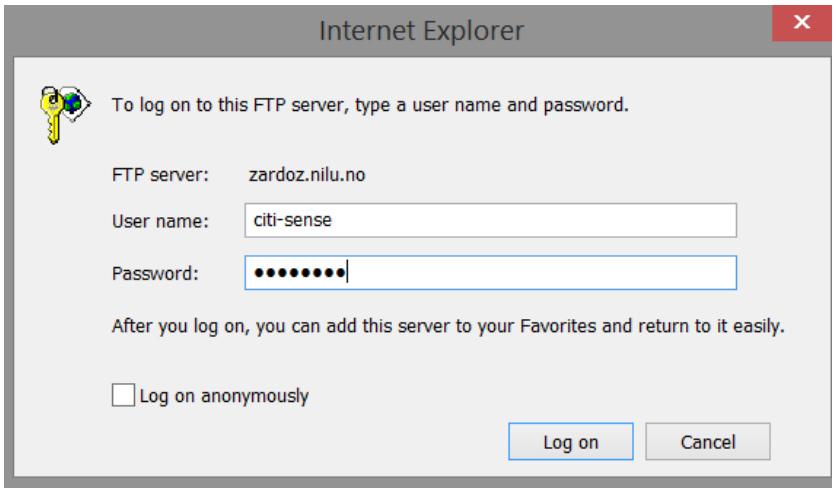

Figure 3-6 Existing CITI-SENSE data in CSV files can be found at this FTP site

The CSV file structure that has been generated as part of making the CITI-SENSE Observations available for analysis by external tools – as well as for ingestion into other database systems has been used as a basis for different data usage situations.

One transfer has been done within the CITI-SENSE project to a relational database system hosting the data for the Map portal data access described in the next section.

As another experiment in data access, a data transfer from the FTP representation files to a new third party database structure, was experimented with during the INSPIRE Hackathon on Citizens Observatories in Barcelona in September 2016. This database structure was also related to the concepts of the ISO 19154 Observation&Measurement model and this facilitated an easy mapping which allowed for the data conversion to be done within a few hours during the hackathon event. The results are examplified through a web portal demonstration described in the next section.

An experiment was also done in mapping the data into a linked data/RDF representation form, as shown in the figure below.

Figure 3-7 Mapping CITI-SENSE data into RDF/Linked Data

The conclusion from the RDF/Linked Data mapping exercise is that the RDF triple data representation became about 10 times larger in space requirement, and that this was not optimal to do on the lowest level of granularity at the sensor measurement level. It seems more feasible to store the time series data in a more optimised structure and instead relate the RDF/Linked data structure to information on the meta data/schema level.

4 Web Portals

4.1 Overview of CITI-SENSE web portals

A number of web portals was created in the CITI-SENSE project – using various web portal technologies, as further described in deliverable D4.5 on Citizens Observatories.

CITI-SENSE - http://www.citi-sense.eu/ - a web portal providing general information about the CITI-SENSE project.

CO.CITI-SENSE - http://co.citi-sense.eu/ - a web portal providing access to the results from the CITI-SENSE project, including project deliverables.

The next section will describe a new Map Query portal developed at the end of the CITI-SENSE project and now also embedded into the CO.CITI-SENSE top level portal.

4.2 New CITI-SENSE Map Query Portal

At the end of the CITI-SENSE project a new Map Query portal was created to access the historical CITI-SENSE data through a map interface. A new relational database was created to host the data available as CSV files from the CITI-SENSE SEDS server.

The Map Query Visualization web portal provides information about collected sensor data on the CITI-SENSE platform. Results are displayed on the map with various options for enabling and showing different type of data, statistical analysis and data filtering.

This portal was designed in a form of responsive, single page application with a side menu, search bar and a main map, developed with Front-End technologies.

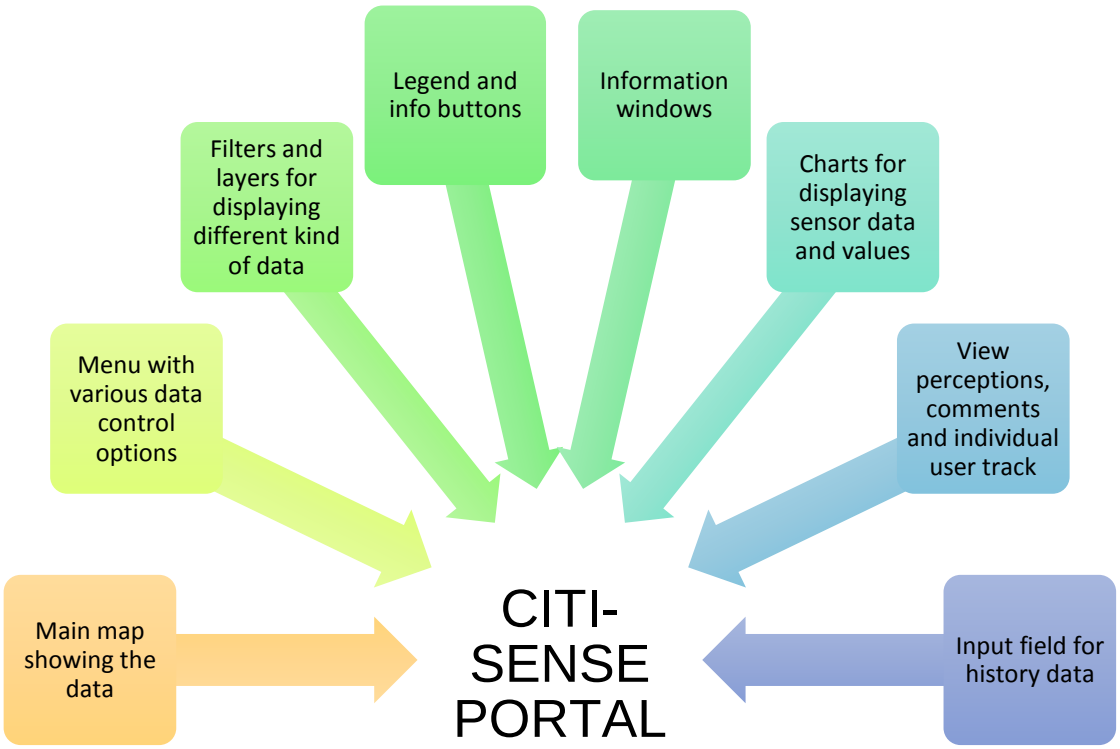

Figure 4-1 MapQuery CITI-SENSE Portal

Front-End technologies used:

1. HTML5
2. CSS3
3. Bootstrap
4. JQuery, JQuery-UI
5. Highcharts
6. AJAX
7. JSON

Main functionalities for displaying different kind of data have used modified U-hopper widgets, adjusted for portal needs.

There are a few widget structures used for gathering data on a portal:

1. **Latest data widget**, used for displaying newest data, in this case from August/September 2016

```
$("#MapId").sensorsbylocation({
        title: "Caqi in Oslo for providers 4 or 5"
        ,subtitle: "O3"
        , type: 'map'
        ,from: "2016-08-19T11:19:21"
        ,to: "2016-08-29T11:19:21"
        ,callbacks: {
                dataLoadCompleted: dataLoadedCallback,
                dataLoadFailed: function() {console.log('data load failed');},
                visualizationReady: visualizationIsReady
        }
        ,layers: [
        {show: showMap, url: urlForInterpolatedM}
        ]
        ,filters: {
        bbox: {
                lowerCorner: {
                        lat: bbxLcorLat,
                        lon: bbxLcorLon
                },
                upperCorner: {
                        lat: bbxUcorLat,
                        lon: bbxUcorLon
                }
        },
        provider: '4,5,9'
        }
        ,measurement: {
                type: 'globalcaqi'
        }
});
```

2. **History data widget**, used for displaying data collected in some period in the past.

```
$("#MapId").sensorslookup({
        type: 'map',
        from: sendFromDateTime,
        to: sendToDateTime,
        callbacks: {
                dataLoadCompleted: dataLoadedCallback,
                dataLoadFailed: function() {console.log('data load failed');},
                visualizationReady: visualizationIsReady
        },
        filters: {
                provider: "4,5,9",
                bbox: {
                        upperCorner: {lat: bbxUcorLat, lon: bbxUcorLon},
                        lowerCorner: {lat: bbxLcorLat, lon: bbxLcorLon}
                }
        },
        measurement: {
                average: true,
                latest: false,
                type: 'globalcaqi'
        }
});
```

3. **Individual User track widget**, where user can follow his state and motion by entering user ID

```
$("#MapId").aqi({
        type: 'map'
        ,env: 'prod'
        , participant_id: userId
```

```
        , measurement: measurementValue
        , observedProperty: propertySelected
        , aqis: [
                {
                        from: 1,
                        to: 2,
                        color: 'rgb(0,255,0)'
                }, {
                        from: 3,
                        to: 4,
                        color: 'rgb(255,255,0)'
                }, {
                        from: 5,
                        to: 6,
                        color: 'rgb(255,0,0)'
                }
                ]
});
```

4. **History graphicon widget**, used for displaying chart data from some period in the past

```
$("#OneDaydataHistory").sensorstatistics({
                type: 'historic'
                ,filters: {
                        sensorId: sensorId
                }
                ,measurement: {
                        type: "caqi",
                        observedProperty: property
                }
                ,title: property
                , subtitle: "Measurements from " + displayDateFrom[0] + " to " + displayDateTo[0]
                ,from: dateFrom
                ,to: dateTo
        });
```

For displaying rest of the data, such are User Perception, Comments, Latest data charts, CITY APIN...etc, custom made code is used with AJAX.

SOME OF THE MAIN FUNCTIONALITIES

1. **Selecting a location** from a predefined set of locations gathered from server, the web page will upload on default the last measured values on the following sensors:

- Static sensors - **Latest data widget**
- Mobile sensors - **Latest data widget**
- User Perceptions - custom made code
- User Comments - custom made code

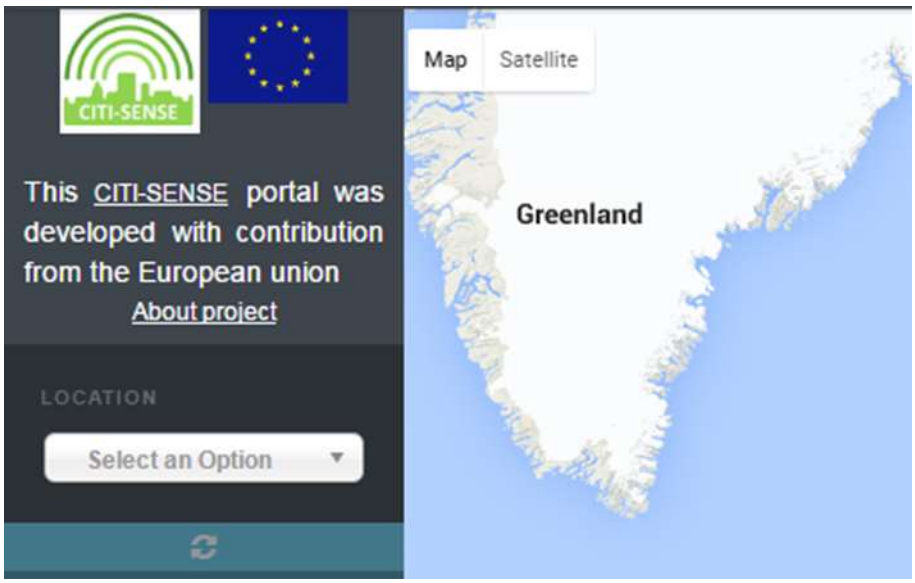

Figure 4-2 Selecting a location

2. **Selecting a time period for specific location** with following options:

- Latest observation - **Latest data widget**
- Specific time period - **History data widget**

Specific time period is used to see data from some period in the past.

3.**Filtering and showing/hiding different kind of data** with following options:

- Layers - Hiding and showing data for Static, Mobile sensors and User Perception and Comments

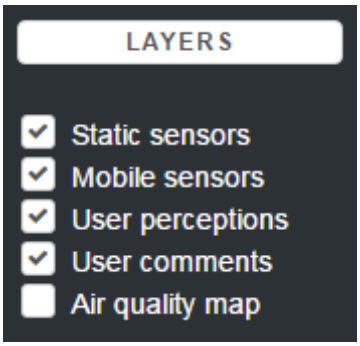

Figure 4-3 Filtering options

- Filters - **Latest data widget for some specific caqi,** showing and hiding User Perception for different time

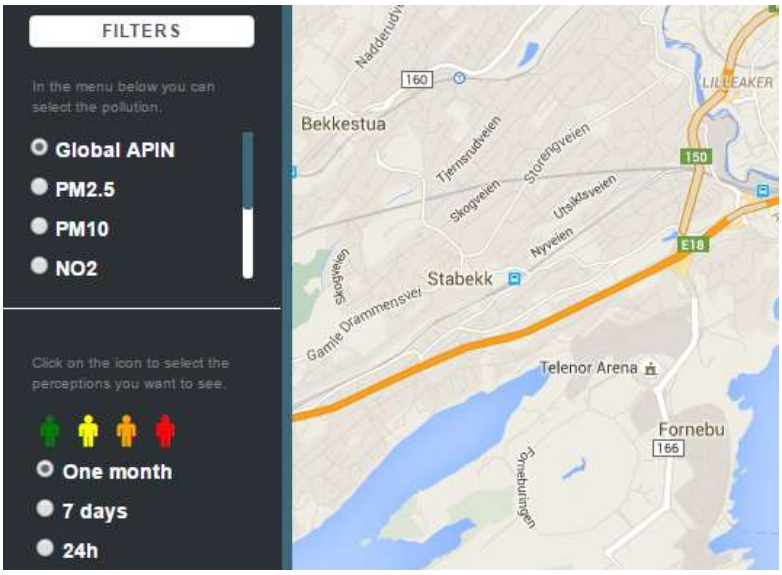

Figure 4-4 Filter for various air quality factors

4. **Individual user track** where user can see his personal data, by entering his id. **Individual User track widget** is used.

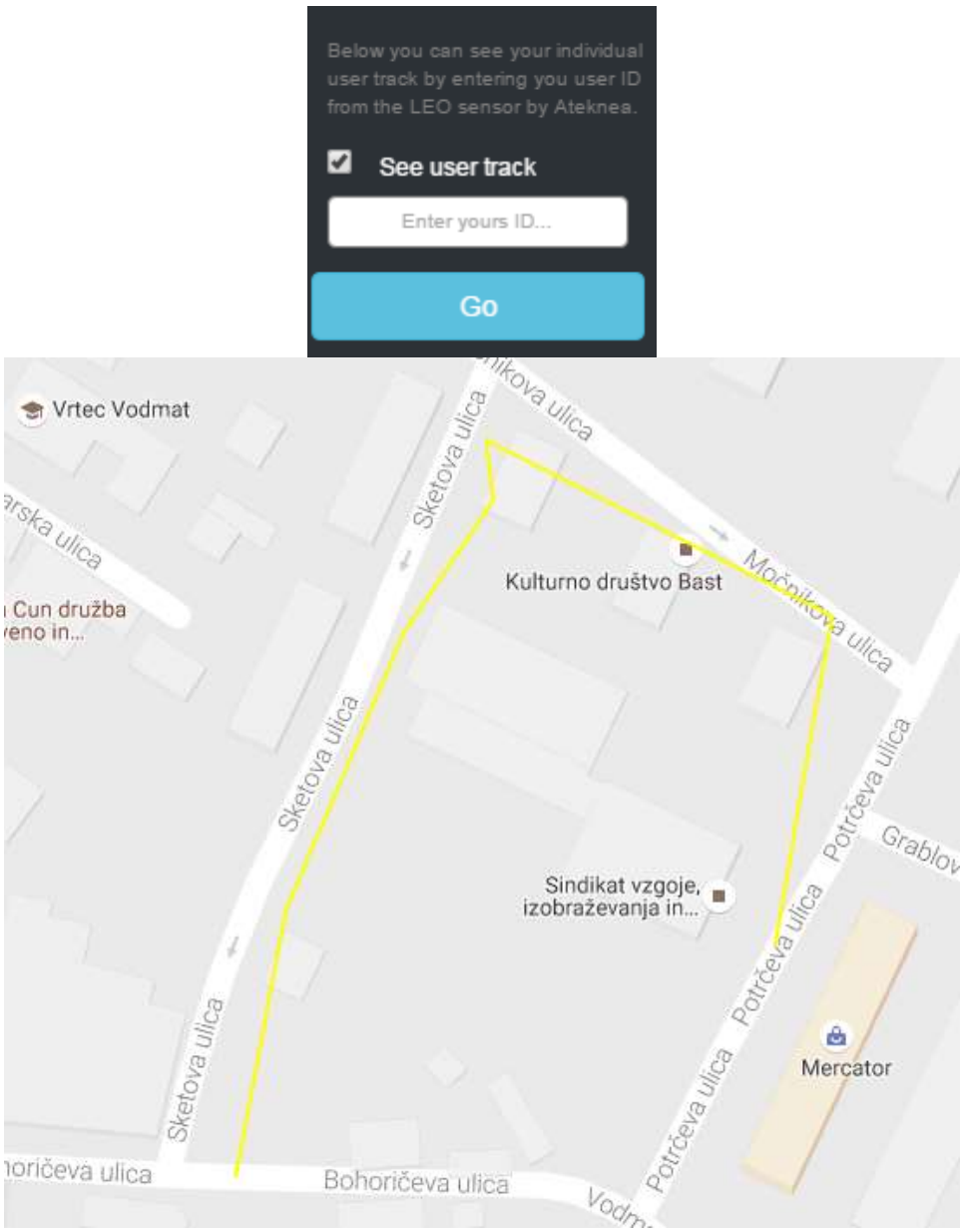

Figure 4-5 See user track

5. **See individual sensor informations** - by clicking on sensor on a map, information and charts about sensor data are displayed in a window.

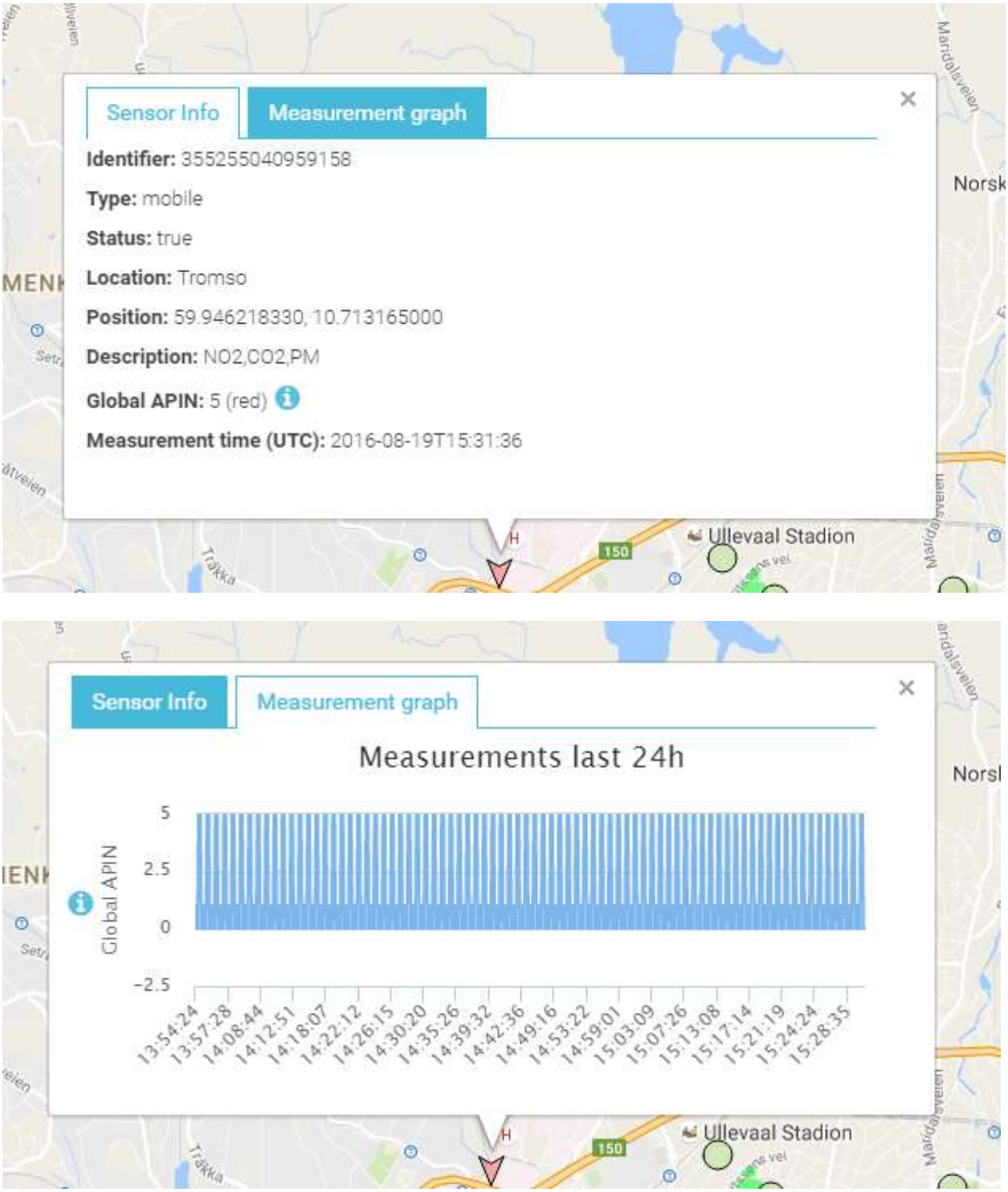

Figure 4-6 Observation & Measurement data for a specific time period

For the sensor data displayed from some period in the past **History graphicon widget** is used.

4.3 Experiment with 3rd party Portal and Data access

The CITI-SENSE data made available through the CSV files was used as basis in a hackathon event on Citizens Observatories and VGI that we arranged during the INSPIRE'2016 conference in Barcelona in end of September 2016.
http://inspire.ec.europa.eu/events/conferences/inspire_2016/schedule/submissions/365.html

The discussion was also related to the use cases from the 4 new Citizens' Observatories on Land Coverage and Land Use that is starting in the fall of 2016. How can data from a variety of data sources be made interoperable and integrated. Can we have a common data model or ontology across different areas? How to represent information for answering concrete environmental issues, such as maintaining biodiversity, health, improve waste re-use, prevent flooding, reduce noise, ensure dark nights, etc ? Can the sensor/observation data be stored in an interoperable way through the data models and observations of the various project approaches ?

The experiment successfully demonstrated that data from the CITI-SENSE project based on the CITI-SENSE data model easily (within a couple of hours) could be mapped into the data model of SensLog fromthe SDI4Apps project, http://sdi4apps.eu/2015/12/senslog-solution-for-sensor-networks/ - and CITI-SENSE data could be displayed through a new web portal interacting with a relational database storing the corresponding representation of CITI-SENSE data.

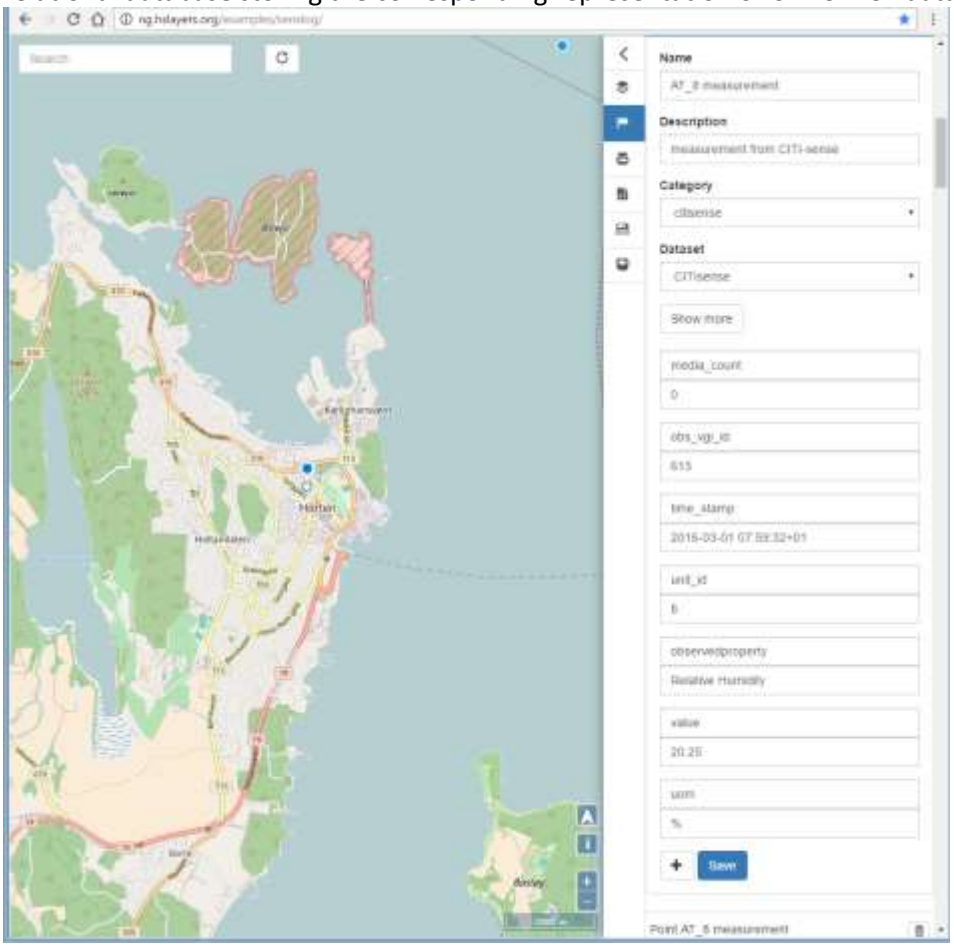

Figure 4-7 Showing CITI-SENSE observation data through a third party system

The Figure 4-7 shows access to CITI-SENSE data that was instantiated into a relational database storage from the historical CITI-SENSE CSV files.

5 Widgets

5.1 Overview of CITI-SENSE widgets

CITI-SENSE has used and developed a number of Widget frameworks. An overview is provided below.

More detailed information about the CITI-SENSE widgets can be found in the part 2 document of this D7.6 deliverable.

5.2 Widget framework for Measurements

The following widget framework for measurements has been designed and provided by the CITI-SENSE partner Dunavnet.

The documentation and demo source code has been provided so that mobile and web applications can easily be developed.

The framework has the following features:

- Based on Highcharts widgets (http://www.highcharts.com/demo/)
- Integrated with PHP in order to allow simple customization
- Demos with source code provided for iOS, Android and web platforms
- Supports mysql, mssql, postgresql and mongoDB
- Currently hosted on DNET server, but can also run other places.
- Documentation is here: http://srv.dunavnet.eu/citisense/
- Demo page is here: http://srv.dunavnet.eu/citisense/demo/

The installation and usage of this is further described in D7.6 part 2.

5.3 Widget framework for Questionaires

A widget framework for multi-channel questionnaires has been developed by U-Hopper.

Go to www.civicflow.com and choose "CivicFlow or Social Login" or choose "Register" if you want or need to create a new account.

Registering as a user is uncomplicated and takes a minute or two.

The further description of this can be found in D7.6 part 2.

5.4 CITI-SENSE Sensors Visualization Widgets

The CITI-SENSE visualization widgets that has been presented in D7.6 part 1 and part 2, has extensively covered various possibilities for visualizing the data acquired by a single sensor. Given a sensor, a set of presented widgets allow to visually depict a real-time sensor measurement value (in a form of configurable Gauge chart), draw historical sensor measurements on a timeline (in a form of Line chart), and display historical sensor measurements on a map (for sensors publishing geographical coordinates of a measurement point).

The widgets have been designed with the following set of requirements in mind: (1) configurability, allowing developers to easily configure the widgets for visualizing various aspects of measured data (e.g., range of measured values, coloring schemes for visual interpretation of data, etc.), (2) reusability, allowing developers to easily reuse the same widget for multiple measurements, and finally (3) embedding, allowing to easily plug widgets into 3rd party desktop and mobile applications and as such promote the CITI-SENSE data for the community. To reach that goal, the CITI-SENSE widgets has been developed using modern HTML5 techniques.

The following shows a reference to code and further descriptions for the CITI-SENSE widgets.

http://co.citi-sense.eu/CitizensObservatoriesToolbox/Widgets.aspx

Widget name	Storage platform	Code link	Usage description and link
Real-time sensor value	WFS	http://citisense.u-hopper.com/widgets/sensors/jquery.sensor.statistics.js	http://citisense.u-hopper.com/pilots/common/#tabs-realtime
Historical sensor values	WFS	http://citisense.u-hopper.com/widgets/sensors/jquery.sensor.statistics.js	http://citisense.u-hopper.com/pilots/common/#tabs-historic
Physical activity levels map	WFS	http://citisense.u-hopper.com/widgets/activities/jquery.activities.js	http://citisense.u-hopper.com/pilots/barcelona/#tabs-activity-map
Physical activity levels graphs	WFS	http://citisense.u-hopper.com/widgets/activities/jquery.activities.js	http://citisense.u-hopper.com/pilots/barcelona/#tabs-activity-graph
Real time graphs	SensApp	http://toolbox.citi-sense.eu/citisense.graph2.js	Widget Real Time Thermal Widget Real Time Acoustic
Results Acoustic & Thermal	SensApp	http://toolbox.citi-sense.eu/citisense.results.js	Widget Result Thermal & Acoustic
CivicFlow Web questionnaire	WFS	http://citisense.u-hopper.com/widgets/perceptions/jquery.questionnaire.js	http://citisense.u-hopper.com/pilots/schools/#tabs-questionnaire
CivicFlow Web questionnaire results	WFS	http://citisense.u-hopper.com/widgets/perceptions/jquery.questionnaire.statistics.js	http://citisense.u-hopper.com/pilots/schools/#tabs-questionnaire-results

6 Sensors, Surveys and Mobile Apps

6.1 Sensors

A number of sensor packages and sensor platforms has been developed and deployed during the CITI-SENSE project.

They are document in more detail in the public WP8 deliverables – in particular in:

- D8.4 Sensor platforms enhancement
- D8.5 Final sensor platform report

In this section the two principal architectures of connecting sensors to the CITI-SENSE data server (SEDS) will be described.

Figure 6-1 Sensor platforms from the CITI-SENSE project

6.1.1 Data Integration through use of WFS API

LEO stands for Little Environmental Observatory. LEO device has been developed by ATEKNEA partner in WP8. LEO targets WP2 – Urban Quality, and provides sensors to capture NO (nitrate oxide), NO2 (nitrate dioxide) and O3 (ozone). As the device is target to be carried by an end user, it is used in combination with an Android application called ExpoApp2, a merge between former Sensing&Control's CitisensePSP and Ateknea's ExpoSomics android app. ExpoApp2 connects via Bluetooth channel with LEO, and combines smartphone's position and physical activity (calculated using accelerometer sensors) information with NO, NO2 and O3 from LEO device.

Figure 6-2: LEO Device

More information about the LEO sensor can be found at http://citisense.ateknea.com/

The data is being uploaded continuously to the CITISENSE SEDS server.

The following shows an architectural picture of the data flow related to this sensor platform.

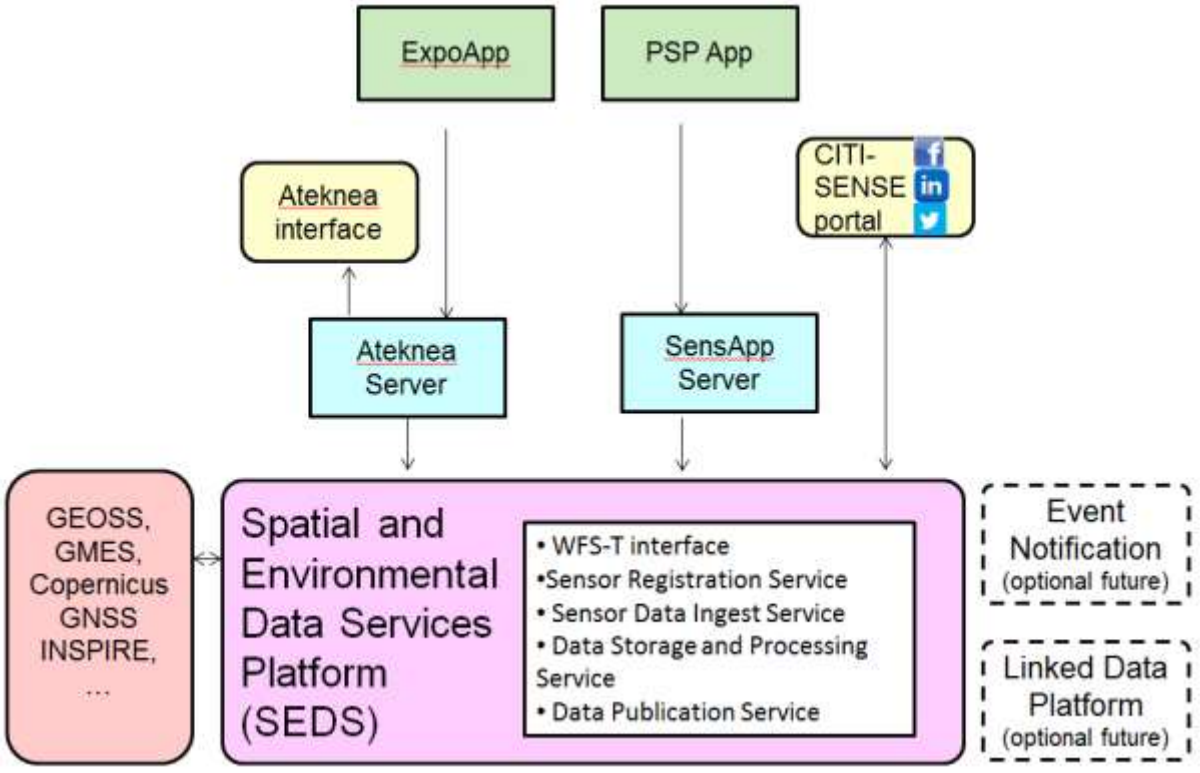

Figure 6-3 Data Flows for the CITI-SENSE PSP App and ExpoApp

In general terms, CitiSensePSP establishes a Bluetooth communication with the Ateknea Personal Sensor Pack (PSP) and reads all the data gathered from the gas and ambient sensors. This data is then processed by a specific Ateknea module and then saved in the smartphone's SD Card. Finally, the processed data is sent to the SensApp server where it can be visualized.

CitiSensePSP is a mobile application that runs on Android devices. This application communicates with the Ateknea's Personal Sensor Pack (PSP) and sends the data to the SensApp Server. The PSP is composed by 5 different sensors: temperature, relative humidity, O_3, NO_2 and CO. The data obtained from these sensors is processed and then sent to the server, where it can be consulted with only some minutes of delay (depending on the configured upload frequency within the app).

Further information about CitiSensePSP can be found in D8.4 and D8.5 and further information about the details of the architecture described here can be found in D7.6 part 2 of this deliverable.

The code for CityAir is open source and available upon request.

6.1.2 Data integration through file ingestion

Geotech, in partnership with Envirologger, provides the commercial sensor platform AQMesh. AQMesh is a static, wireless air quality monitor system to measure the main air pollution gases. It works through a network of arrayed monitors to measure a number of gases.

Sensor data is periodically transferred to a secure, centralized server within the AQMesh network, where it is stored and is used for further dissemination.

The AQMesh sensor network is mainly used as a static sensor observation platform for measuring outdoor air quality in Work Package 2.

The following shows an architectural picture of the data flow related to this sensor platform.

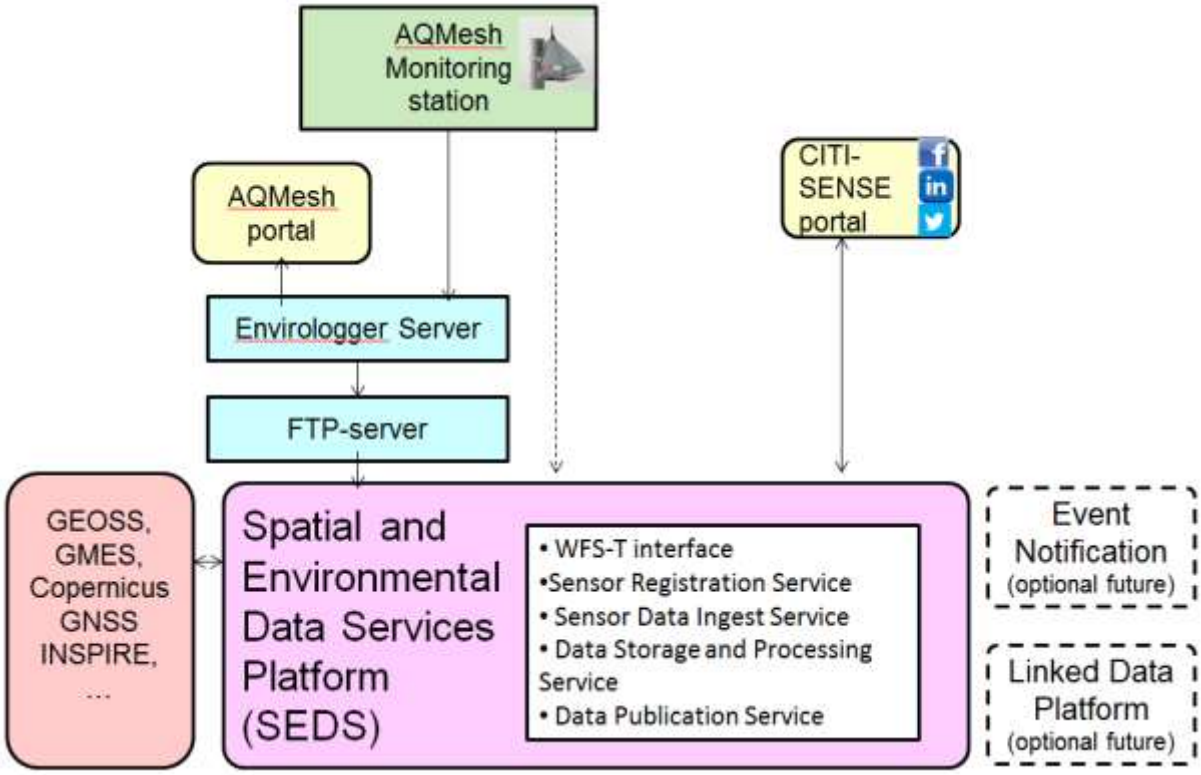

Figure 6-4 Data Flows for the CITI-SENSE Geotech AQMesh Sensor architecture

AQMesh monitoring stations make regular observations and send the resulting measurements to the AQMesh Server, using GPRS communication, where the data is stored on a secure FTP server for further dissemination.

It has been identified that the Data Ingestion component of the SEDS Platform requires a web service that connects to the AQMesh FTP service and pulls data into the SEDS Platform. This requirement and development has been designed using principle of adopting open standards for the data encoding and web service interfaces.

Further information about GeoTech AQMesh can be found in D8.4 and D8.5 and further information about the details of the architecture described here can be found in D7.6 part 2 of this deliverable.

6.2 Surveys

6.2.1 An Introduction to CivicFLow

A widget framework for multi-channel questionnaires has been developed by U-Hopper.

6.2.2 Widget framework for Questionaires by U-Hopper

A widget framework for multi-channel questionnaires has been developed by U-Hopper.

Go to www.civicflow.com and choose "CivicFlow or Social Login" or choose "Register" if you want or need to create a new account.

Registering as a user is uncomplicated and takes a minute or two.

The further description of this can be found in D7.6 part 2.

6.3 Mobile Apps

6.3.1 Introduction to Mobile Apps

The Mobile Apps are an important part of the interaction technologies with citizens.

In the following the architecture and data server connection of the SENSE-IT-NOW App and the CityAir App will be described. These Mobile Apps are further described in the D6.4 Final report on methodology and the D6.5 Report on implementation and demonstration and also in

More detailed information about the CITI-SENSE Mobile Apps architecture can also be found in the part 2 document of this D7.6 deliverable.

6.3.2 SENSE-IT-NOW App

The SENSE-IT-NOW App is a mobile application developed specifically as a toolkit for environmental monitoring. It was developed using Cordova (HTML5, javascript) for cross platform support.

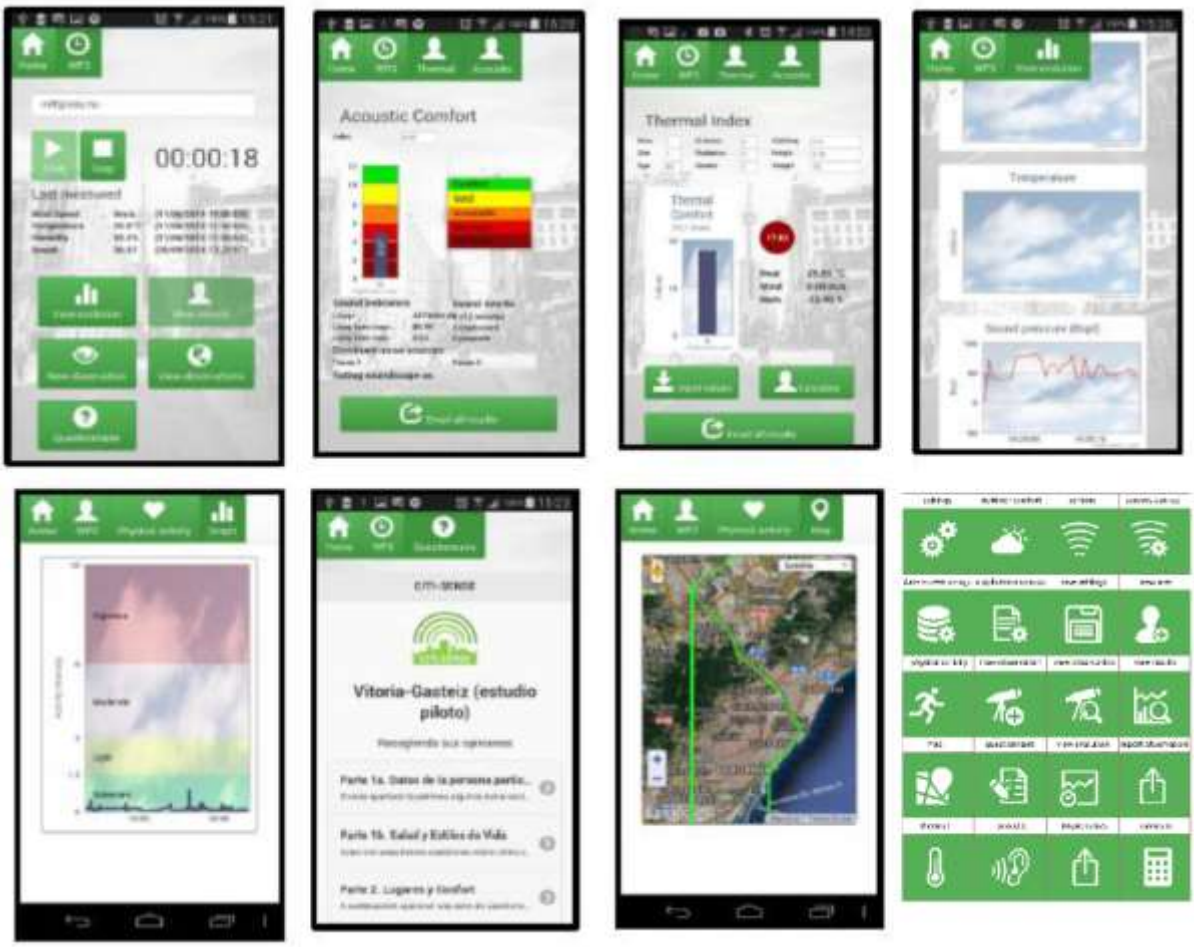

Figure 6-5 Screenshot from SENSE-IT-NOW smartphone application

The following shows the current architectural picture of the data flow related to the SENSE-IT-NOW application.

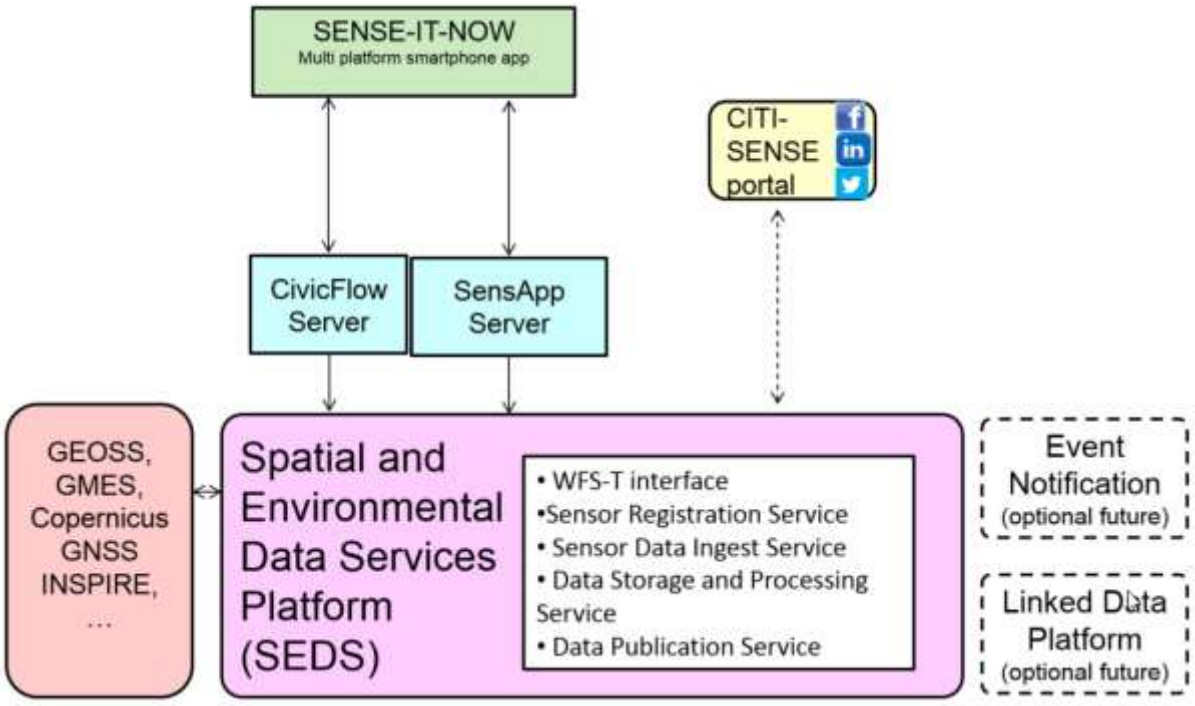

Figure 6-6 Data Flows for the SENSE-IT-NOW architecture

In general, the application can show measurements of any sensor data stored in the CITI-SENSE platform. It also uses the platform for storage of observations of the surroundings (photos taken by the smartphone with a perception tag of pleasant or unpleasant). The CivicFlow application is an integrated part of the SENSE-IT-NOW application and provides an interface for answering questionnaires. Some of these answers are required for the application to calculate comfort indexes and present that to the user.

Further information about the SENSE-IT-NOW app can be found in D6.4 and D6.5 and further information about the details of the architecture described here can be found in D7.6 part 2 of this deliverable.

The code for SENSE-IT-NOW is open source and available upon request.

6.3.3 CityAir App

The main purpose of the CityAir smartphone app is to collect people's perception on air quality and allow the user to register how he/she perceive the air quality where they are at any time. It also gives the people a possibility to view perceptions and comments made by others.

The app's intention is to be easy and simple to use. The app is not location or country specific, and can therefore be used at a global scale.

The users can give their perception about the air quality where they currently are located, by adding an icon of a man. The colour of the icon decides the perception where any other colour apart from green indicates that the current air quality is not good. The users can then choose to suggest the source of the pollution.

The CityAir app is an app that has been developed based on part of the work done in the Citi-Sense-MOB project. CityAir is developed using the same code base and are simplified and modified to be used in a more generic way to support all EIs in the CITI-SENSE project.

In deliverable D6.4, the CityAir has been extensively documented and the deliverable D6.5 has described the test phases involved in the process.

CityAir on Google Play Store
https://play.google.com/store/apps/details?id=io.cordova.CityAir

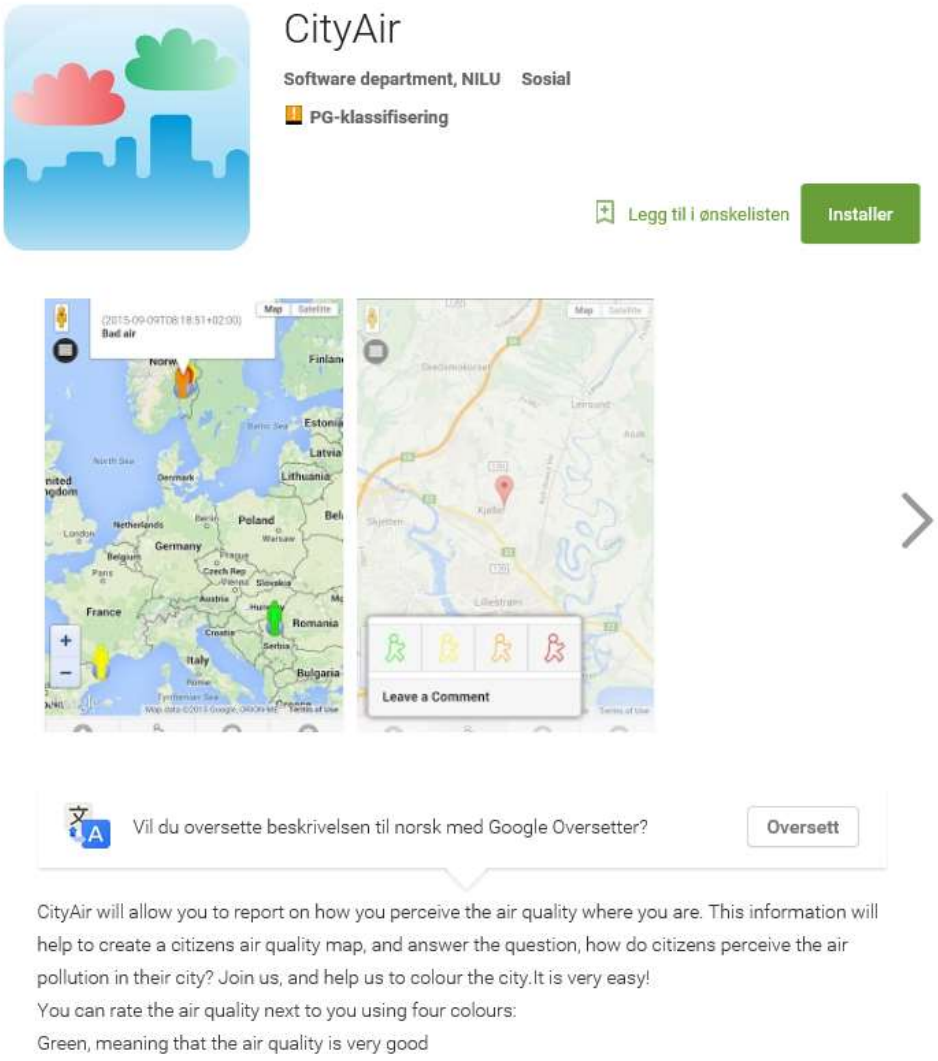

Figure 6-7 CityAir description from Google Play store

GET IT ON Google play

CityAir on iTunes
https://itunes.apple.com/us/app/cityair-perception/id1045646666?l=nb&ls=1&mt=8

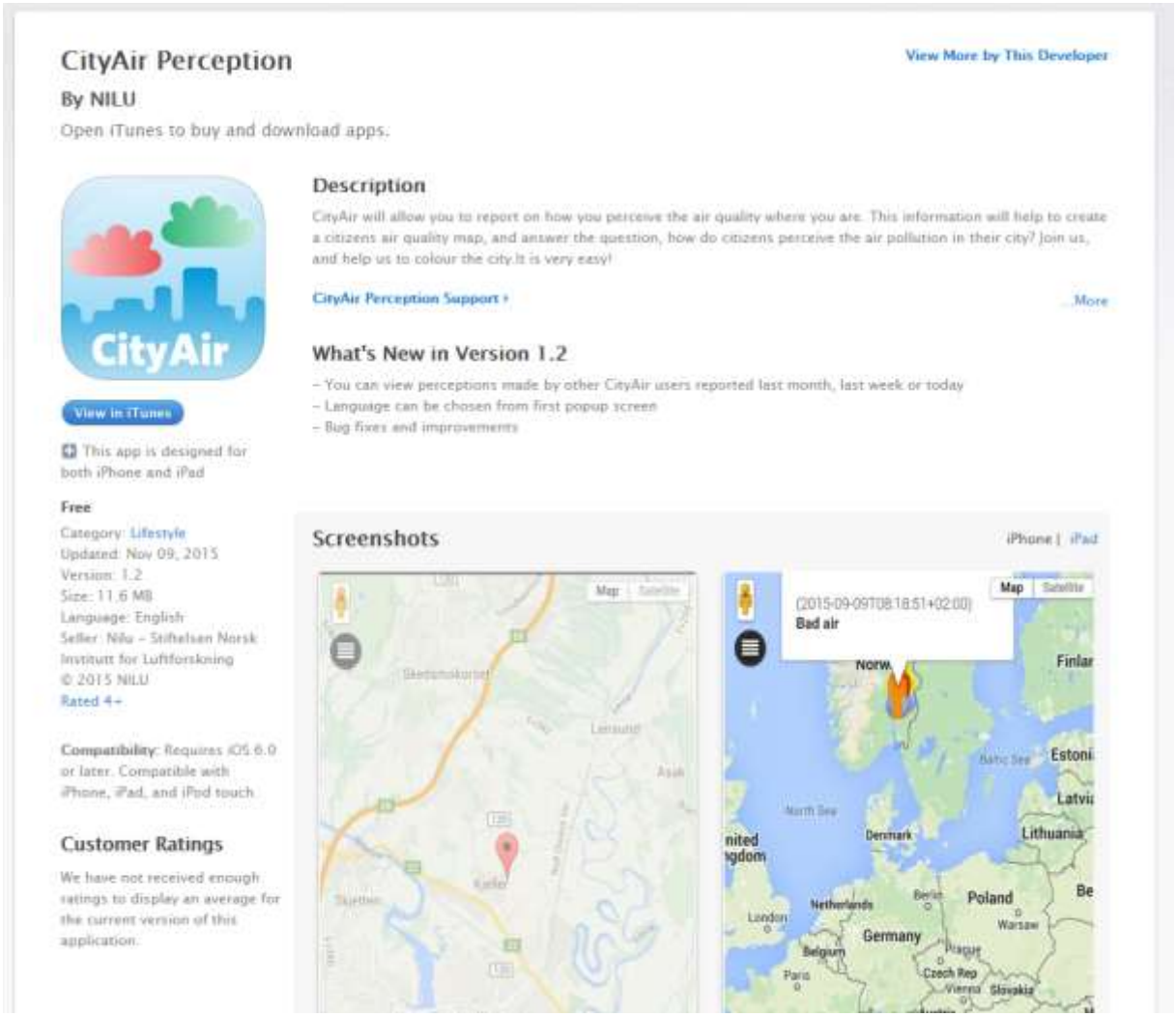

Figure 6-8 CityAir description from Apple iTunes App Store

The following Figure 6-9 shows the current architectural picture of the data flow related to the CityAir application and CITI-SENSE's overall architecture.

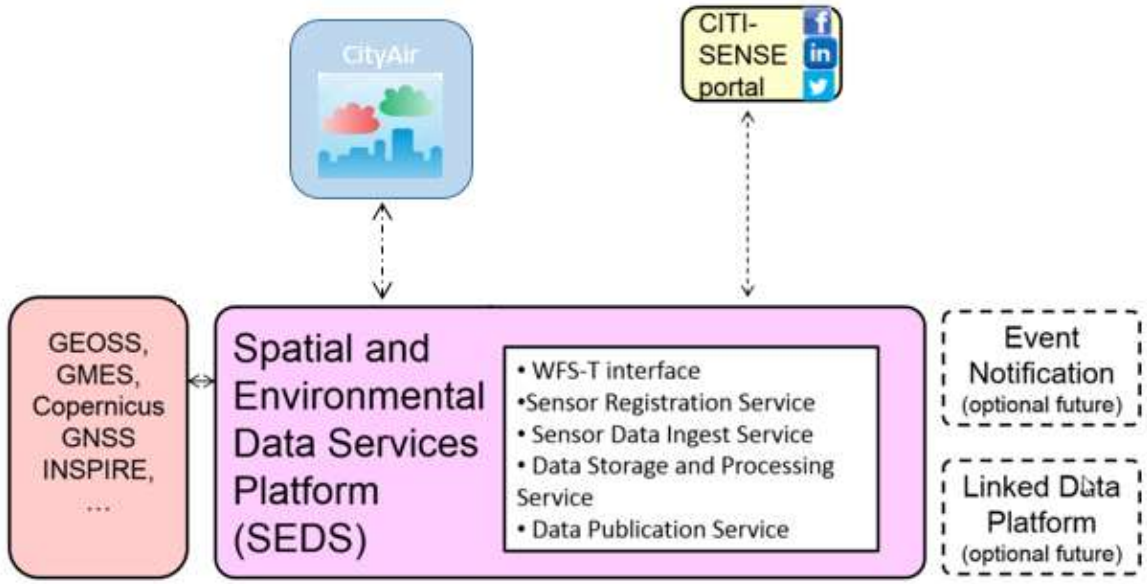

Figure 6-9 Data Flows for the CityAir architecture

CityAir stores all information in a local database on the smart phone. When network is available, it will store the perceptions and the comments from the user on the SEDS server hosted by Snowflake using HTTPS (a secure HTTP protocol that requires username and password).

CityAir can also download and visualize perceptions and comments made by other users.

CityAir's characteristics is typical among Citizens' Observatory applications. It records an observation with data from a sensor (in this case, the users' perception of air quality is the sensor), stores the observation in a data storage, and offer some kind of visualisation of previous observations.

CityAir is a hybrid application, developed with Cordova (earlier known as PhoneGap). Cordova provides a framework for "write once, run anywhere", meaning that the application is written in JavaScript, HTML and CSS, and then built by Cordova to run on iOS and Android. One of the benefits of this approach is that you only need to maintain a single codebase. Another benefit is that by using JavaScript it is likely that some of the business logic can be shared between the Citizens' Observatory application and the optional companion web portal.

Parts of the code written for CityAir could be generalized and used in other Citizens' Observatory projects. This includes code to insert observations into a WFS storage, query a WFS storage for observations, localizing the application to support multiple languages and how to use Google Maps API with observation data.

Further information about the CityAir app can be found in D6.4 and D6.5 and further information about the details of the architecture described here can be found in D7.6 part 2 of this deliverable.

The code for CityAir is open source and available through the following GIT service by NILU:
https://git.nilu.no/citi-sense/cityair-smartphone-app

7 Summary and Conclusions

The deliverable, D7.6 "Platform and Architecture – version 4.0" part 1, 2 and 3, provides the final version 4.0 of the specification and description for the CITI-SENSE architecture and platform. This document D7.6 CITI-SENSE Platform and architecture Version 4 - Part 3: Citizens' Observatory Toolbox - Developer perspective has presented the CITI-SENSE project results from a Citizens' Observatory Toolbox (COT) perspective – with a focus on how the software and tools from the CITI-SENSE project might be used in future Citizens' Observatory projects.

Methods has provided an overview of methodologies and methods developed in the project with references to the relevant deliverables. This part 3 document has presented in particular a proposed Citizens' Observatory Interoperability method focusing on various interoperability areas that should be considered by any Citizens' Observatory project. It combines interoperability perspectives from INSPIRE information value chain, the European Interoperability Framework and the ISO 19119 Service areas.

Data is a main focus in this D7.6 part 3 document – with a particular focus on how to set up a WFS server and the actual data that has been collected through the CITI-SENSE project. The CITI-SENSE architecture aims at the support of multiple types of sensors and mobile apps for collecting data, and the support of multiple ways of providing data for further use and processing, with the use of a common data model and WFS storage support for the Citizen Observation data.

Web Portals represents the most visible interface to the CITI-SENSE result and data. The deliverable D4.5 Citizens' Observatory provides a comprehensive description of the final co.citi-sense.eu web portal. This document has presented a further description of a new Map Query portal which provides access to the final collected CITI-SENSE data through a map interface linked to a new relational database containing the data from the CITI-SENSE project based on the CSV data available at the end of the project.

Widgets are reusable user interface components that can be reused in future Citizen Observatories. The CITI-SENSE widgets are based on HTML5 to enhance the support for portability across platforms. The CITI-SENSE project has provided widgets for a number of areas including maps with sensor locations and index values, real-time and historical sensor values, physical activity level maps and graphs, thermal & acoustic measurement graphs and widgets for questionnaires.

Sensors and sensor platform development has been a main focus in the CITI-SENSE project.
The section in this document describes the two principal ways for sensor platforms to interact with the WFS server – either through a push (API-based) or a pull (file-based) interface

Surveys provide a good approach for interaction with citizens. CivicFlow is a service that can support surveys both from a smart phone and a web portal. The widget framework for Questionnaires by U-Hopper is the basis for interaction with the survey services.

Mobile Apps are important in order to support citizen's participation. CITI-SENSE has aimed at creating mobile apps that can run on multiple smart phone platforms through the use of frameworks like PhoneGap/Cordova and widgets supporting portability through HTML5. The SENSE-IT-NOW and CityAir App are presented as examples of the approach for the development of Mobile App apps. From a developer's point of view the code is available as open source through a GitHub respository. Further work to abstract and generalise the apps as a basis for a more reusable app framework is in progress.

This D7.6 part 3 document has presented the CITI-SENSE Observatory Toolbox results from the seven Toolbox areas of Methods, Data, Web Portals, Widgets, Sensors, Surveys and Mobile Apps – with the objective that elements from these can be useful when new Citizen Observatories are being created and also for any further analysis of the results including the data from the CITI-SENSE project.

8 References

The following CITI-SENSE public deliverables are refered to in the text – and these can all be found at the co.citi-sense.eu website:

http://co.citi-sense.eu/TheProject/Deliverables.aspx

D2.4 Evaluation of the performance of the user cases for outdoor air quality in cities
D3.4 Evaluation of indoor air quality in schools and environmental quality in public spaces
D4.4 Citizens' observatories - methodology assessment
D4.5 Citizens' observatories web portal description
D5.3 Citizens' engagement and empowerment- methodology and protocol
D5.4 Methodological study to support the representativeness of citizens' participation
D5.5 Co-ordinated analysis & Evaluation of empowerment initiatives
D6.4 Final report on methodology
D6.5 Report on implementation and demonstration
D6.6 Report on data fusion/data assimilation applications
D8.4 Sensor platforms enhancement
D8.5 Final sensor platform report