Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

D.10.7 Smart Cities connection to FIWARE Lab

**Project acronym:** FI-Core  
**Project full title:** Future Internet - Core  
**Contract No.:** 632893  
**Strategic Objective:** FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform  
**Project Document Number:** ICT-2013-FI-632893-WP10-D.10.7  
**Project Document Date:** 2015-02-20  
**Deliverable Type and Security:** PU  
**Author:** Sergio García Gómez (Telefónica I+D)  
**Contributors:** Waag Society, Bismart, VTT, Forum Virium, ULPGC, Alfamiro, DRI, FULLIT, IPN, PCITAL, JMP, JIG, Soidemer, UCAN, US, Adevice, Naevatec, URJC, LCU, FBK, POLITO, TOWIRELESS, Inndea, Talleria, Gradiant
1 Summary

1.1 Executive Summary

In order to get requirements, test the technology and prepare the platform to make it ready for adoption as a smart city platform, a number of activities have been carried out together with several cities throughout Europe. This deliverable gathers all the activities carried out by local third party stakeholders, explaining how the data from the municipality has been integrated, the prototypes built and the feedback from the development and validation processes carried out in each city.

1.2 About This Document

This document reports on the activities carried out in Task 10.7, about integration and activities related to connection of Smart Cities to FIWARE Lab.

1.3 Intended Audience

The document targets cities, integrators, GE owners and FIWARE architects in order to understand the application of FIWARE Lab technologies in the smart cities domain.

1.4 Structure of this Document

The document lists the activities carried out per city.

1.5 Acknowledgements

The current document has been elaborated using a number of collaborative tools, with the participation of all the representatives from all the cities involved, including the direct participation of the municipalities themselves, although they do not participate in the activities directly.

1.6 Keyword list

FIWARE, FI-PPP, Smart Cities, Open Data, IoT, Context information.

1.7 Changes History

<table>
<thead>
<tr>
<th>Release</th>
<th>Major changes description</th>
<th>Date</th>
<th>Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>Final version</td>
<td>2015-02-20</td>
<td>Telefonica I+D</td>
</tr>
</tbody>
</table>
# Table of Contents

1  Summary  
1.1  Executive Summary  
1.2  About This Document  
1.3  Intended Audience  
1.4  Structure of this Document  
1.5  Acknowledgements  
1.6  Keyword list  
1.7  Changes History  
1.8  Table of Contents  

2  Introduction  

3  Amsterdam (Netherlands): FIWARE CitySDK-LD Context Broker Interface  
3.1  Scope & Requirements  
3.2  Approach & Architecture  
3.3  Data Sets & Data Model  
3.4  Validation & Findings  
3.5  Appendix  

4  Barcelona (Spain): Open Data  
4.1  Scope & Requirements  
4.2  Open Data  
4.3  Trials and Validation  

5  Espoo (Finland): Energy consumption monitoring  
5.1  Scope & Requirements  
5.2  GEs used and Architecture  
5.3  Data  
5.4  Guidelines for Security  
5.5  Trials and Validation  

6  Helsinki (Finland): Open311 Smart City pilot  
6.1  Scope & Requirements  
6.2  GEs used and Architecture  

D.10.7 Smart Cities connection to FIWARE Lab
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>Opendata</td>
<td>36</td>
</tr>
<tr>
<td>6.4</td>
<td>Trials and Validation</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>Las Palmas (Spain): Smart Port</td>
<td>44</td>
</tr>
<tr>
<td>7.1</td>
<td>Application 1: Smart Port - Wirecloud</td>
<td>44</td>
</tr>
<tr>
<td>7.2</td>
<td>Application 2: Smart Port version 2</td>
<td>46</td>
</tr>
<tr>
<td>8</td>
<td>Lisbon (Portugal): Social and mobility applications</td>
<td>55</td>
</tr>
<tr>
<td>8.1</td>
<td>Introduction</td>
<td>55</td>
</tr>
<tr>
<td>8.2</td>
<td>Scope &amp; Requirements</td>
<td>55</td>
</tr>
<tr>
<td>8.3</td>
<td>Architecture and GEs used</td>
<td>59</td>
</tr>
<tr>
<td>8.4</td>
<td>Infrastructure and equipment</td>
<td>62</td>
</tr>
<tr>
<td>8.5</td>
<td>Open Data</td>
<td>63</td>
</tr>
<tr>
<td>8.6</td>
<td>Trials and Validation</td>
<td>65</td>
</tr>
<tr>
<td>8.7</td>
<td>Feedback on the FIWARE platform</td>
<td>75</td>
</tr>
<tr>
<td>8.8</td>
<td>Conclusions</td>
<td>76</td>
</tr>
<tr>
<td>9</td>
<td>Lleida (Spain): Accessibility and public transport</td>
<td>77</td>
</tr>
<tr>
<td>9.1</td>
<td>Scope &amp; Requirements</td>
<td>77</td>
</tr>
<tr>
<td>9.2</td>
<td>GEs used and Architecture</td>
<td>77</td>
</tr>
<tr>
<td>9.3</td>
<td>Open Data sets</td>
<td>77</td>
</tr>
<tr>
<td>9.4</td>
<td>Trials and Validation</td>
<td>82</td>
</tr>
<tr>
<td>10</td>
<td>Logroño (Spain): Open Data and Irrigation management</td>
<td>83</td>
</tr>
<tr>
<td>10.1</td>
<td>Scope &amp; Requirements</td>
<td>83</td>
</tr>
<tr>
<td>10.2</td>
<td>GEs Used And Architecture</td>
<td>83</td>
</tr>
<tr>
<td>10.3</td>
<td>Open Data</td>
<td>84</td>
</tr>
<tr>
<td>10.4</td>
<td>Trials and Validation</td>
<td>91</td>
</tr>
<tr>
<td>10.5</td>
<td>Screenshots</td>
<td>93</td>
</tr>
<tr>
<td>10.6</td>
<td>Conclusions</td>
<td>95</td>
</tr>
<tr>
<td>11</td>
<td>Logroño(Spain): Open Data &amp; SmartAppCity</td>
<td>96</td>
</tr>
<tr>
<td>11.1</td>
<td>Scope &amp; Requirements</td>
<td>96</td>
</tr>
<tr>
<td>11.2</td>
<td>GEs used and Architecture</td>
<td>96</td>
</tr>
<tr>
<td>11.3</td>
<td>Infrastructure and equipment</td>
<td>102</td>
</tr>
<tr>
<td>D.10.7</td>
<td>Smart Cities connection to FIWARE Lab</td>
<td>4</td>
</tr>
</tbody>
</table>
Future Internet Core Platform

11.4  (Open) Data 102
11.5  Trials and Validation 106

12  Malaga (Spain): Citizen as a sensor 111
   12.1  Scope & Requirements 111
   12.2  GEs used and Architecture 112
   12.3  Infrastructure and equipment 113
   12.4  Data feeds 121
   12.5  Trials and Validation 122
   12.6  Screenshots 126
   12.7  Demo 127

13  Rome (Italy): Integration of mobility open data sets 127
   13.1  Scope & Requirements 127
   13.2  GEs used and Architecture 127
   13.3  Infrastructure and equipment 127
   13.4  (Open) Data 128
   13.5  Trials and Validation 128

14  Santander (Spain): Integration of the SmartSantander IoT facility in the FIWARE framework 133
   14.1  Scope & Requirements 133
   14.2  GEs used and Architecture 133
   14.3  Infrastructure and equipment 139
   14.4  Parking sensors 141
   14.5  Data 146
   14.6  Trials and Validation 150
   14.7  Tutorials 177
   14.8  Dictionaries 178

15  Sevilla (Spain): Smart Fountains 186
   15.1  Scope & Requirements 186
   15.2  GEs used and Architecture 187
   15.3  Infrastructure and equipment 187
   15.4  Data 192
15.5 Trials and Validation 195
16 Sevilla (Spain): A smart-city application for detection of crows through video cameras. 200
16.1 Scope & Requirements 200
16.2 GEs used and architecture 200
16.3 Kurento Media Server and the Crowd Detector Filter 201
16.4 The Application Server: integrating with the context broker 203
16.5 Demonstrating and validating the application 204
16.6 References 207
17 Torino (Italy): Monitoring Non-Emergency Calls in Torino Smart City 208
17.1 Scope & Requirements 208
17.2 GEs used and Architecture 208
17.3 Open data 211
17.4 Trials and Validation 213
17.5 Screenshots 215
18 Trento (Italy): Smart Campus 218
18.1 Scope & Requirements 218
18.2 GEs used and Architecture 218
18.3 Infrastructure and equipment 225
18.4 (Open) Data 226
18.5 Trials and Validation 228
19 Valencia (Spain): Smart Taxi 231
19.1 Scope & Requirements 231
19.2 GEs used and Architecture 231
19.3 Infrastructure and equipment 232
19.4 Open Data 232
19.5 Trials and Validation 234
20 Valencia (Spain): Open data migration 237
20.1 Scope and Requirements 237
20.2 GEs used and Architecture 237
20.3 Infrastructure and Equipment 238
## 21 Vigo (Spain): Integration of Vigo data sources into the FIWARE architecture

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.1 Scope &amp; Requirements</td>
<td>241</td>
</tr>
<tr>
<td>21.2 GEs used and Architecture</td>
<td>241</td>
</tr>
<tr>
<td>21.3 Open Data</td>
<td>242</td>
</tr>
<tr>
<td>21.4 Test and Validation</td>
<td>263</td>
</tr>
</tbody>
</table>
2 Introduction

The FIWARE cloud and software platform is the perfect catalyst for an open ecosystem of entrepreneurs aiming at developing state-of-the-art data-driven applications. This ecosystem is formed by application developers, technology and infrastructure providers and entities who aim to leverage the impact of developing new applications based on the data they produce and publish. In this context Cities will play a unique role, especially those implementing a “Smart City” strategy, who open up their data to facilitate the creation of applications built by developers that form part of this ecosystem. Building applications based on FIWARE is quick and easy because they make use of pre-fabricated components in its cloud, sharing their own data as well as accessing open data from cities. This saves time and money for developers when creating their application and offers them the opportunity to test their application with real data from cities.

During the last months, a number of cities, in cooperation with several local stakeholders, have been working on developing smart city applications on top of city data which was made available using the FIWARE technology, namely using the FIWARE Lab infrastructure and some Generic Enablers. In this context, the majority of projects were developed using a mix of the following three types of approaches: deploying datasets through the open data platform, based on CKAN [CKAN]; integrating city sensors with the FIWARE IoT platform and their consumption through the NGSI context API (Next Generation Service Interfaces - FIWARE Open RESTful API Specification); and finally, integrating dynamic and/or structured information through the context API. The current status of this work is summarized in the table below.

<table>
<thead>
<tr>
<th>City</th>
<th>Country</th>
<th>IoT</th>
<th>Open Data</th>
<th>Context</th>
<th>Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amsterdam</td>
<td>Netherlands</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>CitySDK-LD integration</td>
</tr>
<tr>
<td>Barcelona</td>
<td>Spain</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Espoo</td>
<td>Finland</td>
<td></td>
<td>✔</td>
<td></td>
<td>Energy consumption dashboard</td>
</tr>
<tr>
<td>Helsinki</td>
<td>Finland</td>
<td></td>
<td>✔</td>
<td></td>
<td>Participation dashboard (CitySDK-Open311)</td>
</tr>
<tr>
<td>Las Palmas</td>
<td>Spain</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>Port management dashboard</td>
</tr>
<tr>
<td>Lisbon</td>
<td>Portugal</td>
<td>✔</td>
<td></td>
<td></td>
<td>Mobility and social networks</td>
</tr>
<tr>
<td>Lleida</td>
<td>Spain</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>Public transport and accessibility</td>
</tr>
<tr>
<td>Logroño</td>
<td>Spain</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>Smart watering, City App</td>
</tr>
<tr>
<td>Malaga</td>
<td>Spain</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Citizen as a sensor</td>
</tr>
</tbody>
</table>
The following sections of the document describe in detail the activities and outcomes carried out in all the aforementioned cities.
3 Amsterdam (Netherlands): FIWARE CitySDK-LD Context Broker Interface

3.1 Scope & Requirements

The CitySDK Linked Data API offers unified and direct access to open transport, mobility and general data from government, commercial and crowd sources alike. It is designed to work closely with other open source projects as OpenTripPlanner, Analyst, Open311 and OpenStreetMap. Based on objects linked to a geometry, the API provides for a linked-data (JSON-LD) view of the added data; one query about one object provides results from multiple datasets, annotated using semantic web technologies.

For CitySDK-LD version 1.0 we’ve aligned this API with the NGSI10 interface of the Orion Context Broker, making the CitySDK-LD API an alternative GE to The Orion CB for certain applications. Data can be added and results can be queried across both interfaces, providing an extended capability without compromising compatibility.

Source code, documentation and other components of the CitySDK ecosystem are available through Github https://github.com/waagsociety/citysdk-ld

3.2 Approach & Architecture

The Orion Context Broker implements update and query interfaces, as well as a subscription API on entities with types and attributes. In the CitySDK-LD API we speak of objects with data on typed layers. The parallels are not hard to see, and the implementation of the NGSI10 interface did not require extensive surgery on the CitySDK-LD API.

We’ve decided to use the CitySDK data model as this is capable of conveniently encapsulating the NGSI model, with the exception of the subscription interface, which we’ve added, and is available only through the NGSI interface.

The advantage of this approach becomes apparent when you realize that all data in the CitySDK-LD API becomes available through the NGSI interface and vice versa. The interface through which the data was originally added is irrelevant.

There are, however, some differences. The geography the NGSI10 supports for entities is limited to single Points, which means that in some cases (Multi) Polygons or (Multi) LineStrings need to be replaced by their centroid in the NGSI output. Another small difference is that the CitySDK-LD API does not support data layers without type. This means that context updates adding entities without an entity-type are not supported.
3.3 Data Sets & Data Model

Although the original project required the CitySDK-LD API to be an API for mobility applications, it soon became apparent that in order to support rich enough mobility applications the data needed required functionality to cater for much broader categories of data than strictly mobility related sets.

Loops in the road measuring traffic throughput and cameras measuring trajectory timings are clearly mobility related. For the administrative geometry of a city this is less clear, until you add parking tariffs into the equation. This has led to an API that caters to many types of open data (soon also less open variations), making it suitable for crowd-sourced information, official registry information, real-time sensor data, public transport, physical infrastructure etc.

The CitySDK goal of harmonization of APIs in Europe is rather less based on standardization than on thorough description of (machine readable) annotation of data sets and API’s. In the CitySDK-LD API we are adopting a pragmatic approach using semantic web technologies and modern, proven, web-based API practices. These technologies are used to describe data so that there’s enough information to align data sets from different sources even without a standardized data model. Anticipating more processing power, more data and better algorithms we believe there is going to be less need for formal standards. Machines will be increasingly able to process structured data even without total standardization.

The use of semantic web technology suggests unique URIs for objects that are referenced. CitySDK-LD implements just such a scheme, which we’ve also adopted in the NGSI interface. The entity_id is an URI that uniquely identifies a particular entity across all (registered) CitySDK-LD endpoints. In the example output in the appendix we see the id for the province of Gelderland in the Netherlands is ‘http://rdf.citysdk.eu/asd/admr.nl.prov.gelderland’. The Smart Citizen Kit at Waag Society (physical sensing device) has the id: ‘http://rdf.citysdk.eu/asd/sck.462’.

The ‘asd’ part of the URI points to a specific instance of the API, running in Amsterdam, in this case. This mechanism makes it possible to tie data sets together across all of Europe. Much remains yet to be done (typing policies, for instance), but this approach will accommodate many diverse functional
requirements in many use cases. As you can see from the output as shown in the appendix, the results mapping is, even when the data model maps quite conveniently onto the NGSI entities, not very straightforward. Below we discuss some of the nomenclature in both interfaces and their mapping.

<table>
<thead>
<tr>
<th>CitySDK</th>
<th>NGSI</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
<td>-</td>
<td>Entry point of an instance of the API. This is relevant insofar as the NGSI entity ID encodes the particular endpoint that the actual data frame is accessible through. In the example above, the id is <a href="http://rdf.citysdk.eu/asd/admr.nl.prov.gelderland">http://rdf.citysdk.eu/asd/admr.nl.prov.gelderland</a>, encoding the Amsterdam endpoint (through the ‘ams’ uri part). In this manner the entities are represented as unique URI’s across different endpoint instances.</td>
</tr>
<tr>
<td>object</td>
<td>entity</td>
<td>In CitySDK an object has an ID, an optional name and a geography. Attribute data is linked to an object in layers. In NGSI the attributes of an entity are linked to the individual entity whereas in CitySDK attributes are organized in ‘layers’ (data sets); this is also where the attributes are described and typed. An NGSI entity is a CitySDK object plus the data (attributes) from a single layer. Multiple NGSI entities can link to the same CitySDK object, but with a different set of attributes each time.</td>
</tr>
<tr>
<td>layer</td>
<td>entity_type / entity</td>
<td>Layers are the main organizational concept in the CitySDK platform. A layer has a type, not necessarily unique across layers. The layer type is what defines the entity type in an NGSI context query. A layer is basically a data set that is linked to ‘underlying’ objects. So when a building is an object, data about this building is organized in layers of historical data, official (cadastre) data, data about building usage, data of sensors attached to the building, data about the sales history, etc. Each of these data frames (a set of key/value attributes belonging to one layer), together with the building itself, constitutes an NGSI entity.</td>
</tr>
<tr>
<td>field</td>
<td>attribute</td>
<td>The CitySDK field describes an individual attribute within a layer. The metadata available is richer than the NGSI interface support. In CitySDK-LD we support name, type, unit, language, equivalent property and description metadata. This metadata applies to all data frames (‘entities’) in the layer, as opposed to NGSI,</td>
</tr>
</tbody>
</table>
3.4 Validation & Findings

The development of the NGSI implementation was done in parallel with the development of the 1.0 version of the CitySDK-LD api. The 1.0 release will be deployed in the coming weeks, including the NGSI functionality, in an Amsterdam based instance, and will be available on github, of course.

Throughout development we’ve been running tests that confirm the functional correctness of the implementation (see appendix), but we have not implemented the API in a large scale production environment yet.

The process of implementing the NGSI10 interface was somewhat of a mixed blessing. For one, the expressiveness of the data model is less than that of the native API (polygon becomes point, for instance), which ‘feels’ like it is a shame. Also, the provenance of the data is unclear, the attribute metadata less expressive, the JSON output non-standard (i.e. not GeoJSON or JSON-LD). On the other hand, there’s the subscription interface, which has clear use cases, as well as the fact that, NGSI being a more established interface model, we increase the potential audience.
Figure 1: The Smart Citizen Kit; one of the sensor platforms in CitySDK-LD during a pilot in 2014
### Figure 2: output of CitySDK-LD NGSI test run giving an overview of the implemented functionalities.
JSON result of convenience call “/ngsi10/contextEntityTypes/csdk:borders” (only first result entity shown)

```json
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "gid",
                        "type": "rdfs:label",
                        "value": "9"
                    },
                    {
                        "name": "admn_level",
                        "type": "rdfs:label",
                        "value": "1"
                    },
                    {
                        "name": "provincien",
                        "type": "xsd:string",
                        "value": "Gelderland"
                    },
                    {
                        "metadatas": [
                            {
                                "name": "location",
                                "type": "string",
                                "value": "WSG84"
                            }
                        ],
                        "name": "geography",
                        "type": "coords",
                        "value": "52.0617370131056, 5.9391067002224"
                    }
                ],
                "id": "http://rdf.citysdk.eu/ams/admr.nl.prov.gelderland",
                "isPattern": false,
                "type": "csdk:borders"
            },
            "statusCode": {
                "id": "http://rdf.citysdk.eu/ams/admr.nl.prov.gelderland",
                "isPattern": false,
                "type": "csdk:borders"
            }
        }
    ]
}
```
"code": "200",
"reasonPhrase": "OK"
}
D.10.7 Smart Cities connection to FIWARE Lab

JSON result of the equivalent call "/layers/admr/objects" (only first result feature shown); the ‘admr’ layer has type ‘csdk:borders’

```json
{
  "@context": {
    "@vocab": "http://rdf.citysdk.eu/ams/",
    "cdk_id": "cdk_id",
    "data": "layerData",
    "date_created": "dc:date",
    "dc": "http://purl.org/dc/elements/1.1/",
    "dcat": "http://www.w3.org/ns/dcat#",
    "features": "apiResult",
    "foaf": "http://xmlns.com/foaf/0.1/",
    "geometry": null,
    "geos": "http://www.opengeospatial.org/ont/geosparql#",
    "layer": {
      "@id": "createdOnLayer",
      "@type": "@id"
    },
    "layers": {
      "@container": "@index",
      "@id": "layerOnObject"
    },
    "org": "http://www.w3.org/ns/org#",
    "owl": "http://www.w3.org/2002/07/owl#",
    "properties": "_:properties",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "title": "dc:title",
    "type": null,
    "xsd": "http://www.w3.org/2001/XMLSchema#"
  },
  "@type": "APIResult",
  "features": [
    {
      "@id": "objects/admr.nl.prov.gelderland",
      "@type": "Object",
      "geometry": {
        "coordinates": [ [ [ 5.839891, 50.950191 ], [ ... ], [ ... ] ] ]
      },
      "type": "MultiPolygon"
    },
    "properties": {
      "@id": "objects/admr.nl.prov.gelderland",
      "cdk_id": "admr.nl.prov.gelderland",
      "layer": "layers/admr",
      "layers": {
```
"admr": {
    "@id": "layers/admr/objects/admr.nl.prov.gelderland",
    "@type": "LayerOnObject",
    "data": {
        "@context": "http://0.0.0.0:9292/layers/admr/@context",
        "@id": "objects/admr.nl.prov.gelderland/layers/admr",
        "@type": ["LayerData", "csdk:borders"],
        "admn_level": "1",
        "provincien": "Gelderland",
        "gid": "3"
    },
    "layer": "layers/admr"
},

"title": "Gelderland"
},
"type": "Feature"
,...}
4 Barcelona (Spain): Open Data

4.1 Scope & Requirements

Publish and maintenance of a list of datasets from the city of Barcelona Open Data

4.2 Open Data

1. List of Datasets that have been published and maintained in the Barcelona Open Data portal until now

I. Second hand housing. Sale
II. Registered unemployment. By gender
III. Registered unemployment. By duration
IV. Registered unemployment. Evolution of the estimate registered unemployed in Barcelona neighborhoods and months
V. Registered unemployment. Weight of the registered unemployment by neighborhoods on the population aged 16 to 64. Percentages
VI. Number of disability pensions by neighborhoods
VII. Number of pensions for retirement by districts
VIII. Calendar of sports activities
IX. Total number of buildings and housing properties in the building
X. Properties in buildings used primarily for housing according to the number of floors above ground
XI. Properties in buildings used primarily for housing according to the number of floors below ground
XII. Properties in buildings used primarily for housing according year of construction
XIII. Properties in buildings used primarily as housing according the condition of the building
XIV. Properties in buildings used primarily as housing according their facilities
XV. Total number of buildings with housing according number of floors
XVI. Total number of buildings with housing according to floors above ground
XVII. Total number of buildings with housing according the number of floors below ground
XVIII. Number of buildings used primarily as housing by construction year
XIX. Number of buildings used primarily as housing by condition of the building
XX. Number of buildings used primarily as housing by their facilities
XXI. Number of buildings used primarily for housing by number of properties
XXII. Number of buildings used primarily for housing according to the number of parking spaces
XXIII. Number of buildings used primarily for housing according to the number of floors above ground
XXIV. Number of buildings used primarily for housing according to the number of floors below ground
XXV. Total number of buildings with housing according number of properties
XXVI. Total number of buildings with housing by building type
XXVII. Number of animal inputs and outputs at CAACB (Shelter Pet of Barcelona)
XXVIII. Family dwellings located in buildings used primarily for housing according construction year
XXIX. Main homes by year built
XXX. Main homes according to condition of the building
XXXI. Main homes according to facilities I: heating
XXXII. Main homes according to facilities II: home water, toilet and bathroom
XXXIII. Main homes according to facilities III: telephone and internet
XXXIV. Main homes by number of rooms
XXXV. Main homes according to the number of people living
XXXVI. Main homes as useful surface
XXXVII. Main homes by tenure status
XXXVIII. Family dwellings located in buildings used primarily for housing according to type
XXXIX. Street trees in the city of Barcelona
XL. The cadastral homes. Number of local housing by surface
XLI. The cadastral homes. Number of local housing according to construction year
XLII. The cadastral homes. Owner type and nationality of the owner

2. List of Datasets that will be published and maintained in the Barcelona Open Data portal from now on

I. Local entities File
II. Trees with local interest
III. Game areas
IV. Areas for dogs
V. Parks and gardens
VI. Parking slots for handicapped people
VII. Points of electrical charging stations
4.3 Trials and Validation

Please include several screen shots and a link to an online video with the demo.

The link to the demo:


Screenshots:

Main screen
Dataset description

Dataset information and formats available

D.10.7 Smart Cities connection to FIWARE Lab
## Dataset xls data

<table>
<thead>
<tr>
<th>Python code</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#include &lt;iostream&gt;</td>
<td></td>
</tr>
<tr>
<td>int main() {</td>
<td></td>
</tr>
<tr>
<td>std::cout &lt;&lt; &quot;Hello World!&quot;;</td>
<td></td>
</tr>
<tr>
<td>return 0;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

This is a sample dataset in Excel format, demonstrating the integration of Python code for data manipulation.

### Example of Excel Table

<table>
<thead>
<tr>
<th>Column A</th>
<th>Column B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 1</td>
<td>Data 2</td>
</tr>
<tr>
<td>Data 3</td>
<td>Data 4</td>
</tr>
</tbody>
</table>

### Example of Excel Figure

![Excel Figure](image_url)
5 Espoo (Finland): Energy consumption monitoring

5.1 Scope & Requirements

In the Espoo FIWARE pilot the objective was to demonstrate how energy consumption measurements can be gathered and visualized. The demonstrator collects building level information from Espoo Otaniemi researcher hotel\(^1\) and provides the data to FIWARE platform for processing and visualization. The data is sent to FIWARE in real-time as a standard NGSI update context messages. VTT also analyzed the security and privacy requirements. Figure 1 shows the conceptual architecture of the Espoo pilot.

VTT’s focus was on the research hotel electricity consumption data. The researcher hotel (built 2012) is owned by city of Espoo and it includes 53 apartments. The available data from researcher hotel includes 1) hourly consumption information for electricity, district heat and water, 2) detailed real-time electricity consumption information from building level, 53 apartments, and main HVAC loads, and 3) over 4000 real-time building automation data points including different parameters for energy, indoor conditions and technical equipment.

The demonstration utilizes VTT’s earlier developed an experimental application called “service platform”. The service platform collects measurement data from buildings and other open data sources. The service platform provided the protocol or rules for data transmission between different places as well.

To demonstrate the utilization of the real-time energy consumption data, and as a proof-of-concept for the integration of VTT’s service platform and FIWARE components, we have developed a showcase application visualizing the energy data. The showcase application is hosted in the FIWARE Lab cloud service that enables free experimentation and development with the FIWARE technology. The data were reported to the users by using visualization tools and user interface solutions.

\(^1\) [http://www.unihome.fi/aalto-inn/aalto-inn-home.html](http://www.unihome.fi/aalto-inn/aalto-inn-home.html)
Future Internet Core Platform

Figure 3: The conceptual architecture of Espoo pilot

5.2 GEs used and Architecture

The FIWARE generic enablers used in the pilot were:

- Orion Context Broker
- FIWARE Lab cloud environment
- Proton Complex Event Processing (test)

The architecture of developed service platform is shown in Figure 4. On the VTT side, there is an ESB (Enterprise Service Bus) based system that handles retrieval the energy data from the researcher hotel and forwarding data to the FIWARE. In the VTT platform a poller has been configured to request new data in one minute interval. After processing and storing data locally, the data is transformed to NGSI messages that are sent to the FIWARE Context Broker as updateContext-requests. The data is visualized to the users through a web application that is installed to the FIWARE Lab cloud service instance (Ubuntu 12.04 LTS). The web application has subscribed the data from the Context Broker to get frequent updates automatically and updated information is presented to the user.
5.3 Data

Data is retrieved from the Hotel’s building automation system. The data interface is secured and opened only by request for external activities.

NGSI messages are created from the data received from the researcher hotel. An example of updateContext-request is shown in Appendix 1. An example message showing an update message for weather data is shown in Appendix 2.
5.4 Guidelines for Security

The demonstration does not currently integrate any access control features. The web interface and collected data are openly available for all users. The following mechanisms are recommended when the prototype is applied in larger trials.

1. Authenticate and authorize end-users accessing web interface
   a. The FIWARE identity manager GE (KeyRock) can be utilized to manage users and users roles.
   b. Access to web UI should be secured using HTTPS.
   c. The broker should allow only our web interface component to subscribe our information. Proxy-based Policy Enforcement Point (PEP) architecture\(^2\) may be utilized to control who can subscribe information. In this case the context broker should be available only through PEP proxy. Currently, FIWARE LAB broker instance is open for all direct communication. Consequently, a dedicated broker instance, which is available only for authorized applications, is recommended.

2. Authenticate information sources
   a. HTTPS based authentication should be utilized. Authentication can be based e.g. on username-password combination or on the use of certificates. The latter may be more scalable approach.
   b. A solution for registering new information easily sources to the system is needed. This registration process should provide sufficient authentication data (e.g. password or client-certificate) to devices. It may also be possible to utilize Identity Manager (OAuth protocol). In this case, the context broker (or reverse-proxy in front of broker) has a role of OAuth resource.
   c. FIWARE context broker does not relay authentication information to the visualization application (authenticated https session is only between information provider and broker). Consequently, authentication control must be enforced in the broker (or in PEP proxy in front of the broker).

3. Privacy critical information should be protected in fine-grained manner. For instance, room specific information should be available only for authorized parties - an occupant of one room should not be able to see information of other rooms.
   a. Web interface should authorize use of different information and views. Access control roles (e.g. building maintenance, system administrator, room occupant) may be utilized to ease configuration in case the user amount becomes large. Identity Manager can be used in end-user authentication but enforcement of fine-grained access control must be integrated to

\(^2\) https://github.com/telefónica-fiware/fiware-orion-pep
the web GUI component. Proxy-based PEP architecture provides one alternative for integration but it is unclear whether the PEP access control component can provide sufficient granularity.

b. The FIWARE Identity Manager, KeyRock, provides user’s roles when user information is queried. The roles are given as textual names and ID numbers. Each application (registered to Identity Manager) can have own roles.

We identified potential issues with the FIWARE Policy Enforcement Point (PEP) architecture:

- PEP proxy\(^3\) intercepts NGSI service request and acquires further user information from the identity manager (IdM) by using access token that is part of the request. The client is assumed to know the access token (and all parameters required to acquire it from IdM including application specific secret). This makes solution a bit inflexible for cases with large amount of information providers (clients must be delivered this configuration information each time it changes). Alternatively, the proxy could provide an interface for redirecting requester to IdM to acquire access token. Redirecting unauthenticated requester to IdM is a standard practice in OAuth.

- Orion PEP access control architecture\(^4\) provides single control point for allowing or denying NGSI requests. The authentication information (which proxy acquires from IdM) is not available anymore after context data leave from Orion context broker. Consequently, the architecture does not enable application specific security control (e.g. applications cannot assess reliability of information based on sender’s identity).

5.5 Trials and Validation

Using Orion Context Broker was easy after getting understanding of its features and capabilities. The documentation of the Context Broker was also good and easy to follow. A missing functionality was to query data JSON format after submitting data in XML format to the Context Broker. In addition, it would beneficial that Context Broker could provide also historical data from some selected period. This would have been enough for Espoo demonstration.

The original plan was to utilize also generic enablers providing storage services, but it was not succeeded as the status of the options for storage was uncertain. The big data platform was not the best option for us, because we wanted to query historical data from the web application. It was discussed about the planned short-term storage service to be added to FIWARE, but that did not happen. Then there were discussion about sending data to the FIWARE CKAN, but the

\(^3\) https://github.com/ging/fiware-peg-proxy

\(^4\) https://github.com/telefonicaid/fiware-orion-pep
instructions how to do this weren’t available. We ended up solution that the web application queries data directly from VTT’s storage.

During the trials it became clear that aggregate data should be calculated from the events for performance reasons and Proton CEP was tested for this task. The documentation of the CEP was unfortunately not very clear. For installation the main option was an image for a virtual server but in our case we would have needed more public IP numbers than we had so we installed Proton on our own Tomcat server. War files for this were found but were of different version than the ones that would have been available on the virtual server instance. We created a processing chain that calculated hourly consumption values for the parameters of interest. In our case the base event contained 235 measurements of which 53 were such that an aggregate calculation was required. This kind of event is not the most suitable for Proton’s definition language as it requires defining each parameter and the actions for it resulting in a long definition. The testing was conducted with file based consumers and producers. Integration to the Context Broker appeared to be possible but was not implemented, partly due to minimal documentation. To conclude, Proton appears to be a versatile CEP but more thorough documentation would make it easier and faster to utilize this GE.

The FIWARE Lab’s cloud service was easy to use. We used image of Ubuntu 12.04 and created instance from that.

The organization of FIWARE documentation is not clear; basically it’s very difficult to find needed information. At the beginning of the Espoo pilot implementation, the documentation of the generic enablers was poor as well. However, the removal of poorly implemented and documented generic enablers made use of the catalogue service easier.

The demonstration can be found online from the address: [http://130.206.82.223/](http://130.206.82.223/).
Figure 5: The building energy consumption between selected time window
Figure 6: Graph showing building energy consumption
**Figure 7: Graph showing outdoor temperature**
Figure 8: Graph showing energy produced by the solar panels
6 Helsinki (Finland): Open311 Smart City pilot

6.1 Scope & Requirements

The main objective of the pilot was to prepare a working prototype software to a) integrate Helsinki Open311 issue reporting API to FIWARE platform b) store Helsinki issue reports (based on Open311 standard) to FIWARE Orion Context Broker c) display Helsinki issue reporting API data on FIWARE Wirecloud environment. For this purpose a dedicated Orion Context Broker instance was setup using FIWARE Cloud environment. In addition to this Helsinki Service registry API was used to receive additional information to be displayed in mashup environment.

6.2 GEs used and Architecture

The FIWARE generic enablers are depicted in the used in the pilot were:

- Orion Context Broker
- Wirecloud mashup platform

![Architecture diagram](image-url)
More detailed documentation and widget(s) / server source code and documentation is available on Github with MIT license https://github.com/ForumViriumHelsinki/fi-ware-open311

6.3 Opendata

The opendata source used in the pilot is Open311 standard based Helsinki issue reporting API which is integrated to City of Helsinki internal feedback system.

The Helsinki issue reporting API is based on GeoReporting version 2 (http://wiki.open311.org/GeoReport_v2), which is better known as Open311 specification. The interface is designed in such a way that any GeoReporting version 2 compatible client is able to use the interface. Helsinki Open311 interface is compliant with CitySDK specific enhancements to Open311 and parameters specific to it are marked CitySDK specific. The interface is accessed via HTTP(S) and the full specification is available here: http://dev.hel.fi/apis/issuereporting

Helsinki issue reporting interface allows both reading and writing (reporting) issues via the interface. Issue reporting is only possible using an API key but reading the issues can be done without an API key. Below is JSON example response without CitySDK specific extensions.

```json
[
    {
        "service_request_id": "8fmht6gl470b3qk8pthg",
        "service_code": "171",
        "description": "Itäkeskuksen uimahallin edessä kadulla on monttuja ajotiessä.",
        "service_notice": "",
        "requested_datetime": "2013-05-02T11:02:07+03:00",
        "updated_datetime": "2013-05-02T15:32:29+03:00",
        "status": "open",
        "status_notes": "",
        "agency_responsible": "",
        "service_name": "",
        "address": "",
        "address_id": "",
        "zip_code": "",
        "lat": "60.21263634325148",
        "long": "25.077090230550745"
    },
    {
        "service_request_id": "1ju4p6v3e04pcpobfv7u",
```
The NGSI entity is created from an Open311 object using attributes with similar names. The attributes are copied one-to-one, except "lat" and "long", which are translated to a "position" attribute expected by NGSI. Also four extended attributes from Helsinki/CitySDK are used: "service_object_id", "service_object_type", "title" and "detailed_status".

The attributes are as follows:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_request_id</td>
<td>Unique id identifying the Open311 request</td>
</tr>
<tr>
<td>status_notes</td>
<td>Explanation of why status was changed</td>
</tr>
<tr>
<td>status</td>
<td>open/closed</td>
</tr>
<tr>
<td>service_code</td>
<td>service request type</td>
</tr>
<tr>
<td>service_name</td>
<td>human readable service request type</td>
</tr>
<tr>
<td>description</td>
<td>verbose description of the request</td>
</tr>
<tr>
<td>agency_responsible</td>
<td>Name of agency responsible for fulfilling the request</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>service_notice</td>
<td>Information about the action supposed to fill the request.</td>
</tr>
<tr>
<td>requested_datetime</td>
<td>Date and time the request was created</td>
</tr>
<tr>
<td>updated_datetime</td>
<td>Date and time the request was last updated</td>
</tr>
<tr>
<td>expected_datetime</td>
<td>Date and time the request is expected to be fulfilled</td>
</tr>
<tr>
<td>address</td>
<td>Street address</td>
</tr>
<tr>
<td>media_url</td>
<td>Link to a photo if available</td>
</tr>
<tr>
<td>position</td>
<td>NGSI-style position attribute. Aggregated from &quot;lat&quot; and &quot;long&quot; Open311 attributes</td>
</tr>
<tr>
<td>service_object_id</td>
<td>Identifier for querying related information from Helsinki service registry. Extended Open311</td>
</tr>
<tr>
<td>service_object_type</td>
<td>Object type in Helsinki service registry. Extended Open311</td>
</tr>
<tr>
<td>title</td>
<td>First few words of description. Extended Open311</td>
</tr>
<tr>
<td>detailed_status</td>
<td>RECEIVED / IN_PROCESS / PROCESSED / ARCHIVED / REJECTED. Extended Open311</td>
</tr>
</tbody>
</table>

An example XML transmitted describing a request:

```xml
<queryContextResponse>
  <contextResponseList>
    <contextElementResponse>
      <contextElement>
        <entityId type="Open311" isPattern="false">
          <id>2vi33ihsde5p2u53s53f</id>
        </entityId>
        <contextAttributeList>
          <contextAttribute>
            <name>service_request_id</name>
            <type>string</type>
            <contextValue>2vi33ihsde5p2u53s53f</contextValue>
          </contextAttribute>
        </contextAttributeList>
      </contextElement>
    </contextElementResponse>
  </contextResponseList>
</queryContextResponse>
```
<contextAttribute>
  <name>status_notes</name>
  <type>string</type>
  <contextValue>Kiitos ilmoituksestanne. Olemme välittäneet korjauspyynnön alueen katujen kunnossapitoon. Terveisin Rakennusviraston asiakaspalvelu</contextValue>
</contextAttribute>
<contextAttribute>
  <name>status</name>
  <type>string</type>
  <contextValue>closed</contextValue>
</contextAttribute>
<contextAttribute>
  <name>service_code</name>
  <type>integer</type>
  <contextValue>171</contextValue>
</contextAttribute>
<contextAttribute>
  <name>service_name</name>
  <type>string</type>
  <contextValue>Potholes</contextValue>
</contextAttribute>
<contextAttribute>
  <name>description</name>
  <type>string</type>
  <contextValue>Olisikohan jo korkea aika korjata oheinen terävä kynnys ulkoilutiliellä, johon väistämättä törmää laskettaessa pyörällä Sinisen Huvilan mäkeä alas kohti Töölölähteä. Pienellä määrällä massaa saa ongelman korjattua ja sadat fillaristit tyytyväisiksi.</contextValue>
</contextAttribute>
<contextAttribute>
  <name>agency_responsible</name>
  <type>string</type>
  <contextValue>Rakennusvirasto</contextValue>
</contextAttribute>
<contextAttribute>
  <name>requested_datetime</name>
  <type>datetime</type>
  <contextValue>2014-09-25T14:50:22+03:00</contextValue>
</contextAttribute>
<contextAttribute>
  <name>updated_datetime</name>
  <type>datetime</type>
  <contextValue>2014-09-26T06:45:01+03:00</contextValue>
</contextAttribute>
6.4 Trials and Validation

End-to-end integration and testing was performed in FIWARE Lab environment using two different FIWARE Lab accounts and installing widgets and wiring them manually.
Using Orion Context Broker was quite straightforward. Developing against it was easy despite some encountered bugs. Small problems were caused by unclear API semantics. It is not a REST API and the semantics of the methods (append/update) are not very intuitive.

Developing the Wirecloud widgets proved more problematic. Deployment of a plugin through web interface requires many manual steps, which makes development really hard. There is an eclipse plugin that handles automatic deployment, but we couldn’t get it working easily.

In practice we ended up developing the widgets as stand-alone web pages and only at the last minute integrating it to the wirecloud environment. The architecture of our prototype should have had a separate datasource operator for widgets, but due to the complexity of edit-test cycle in wirecloud, we didn’t implement that and instead made map widget act as a datasource as well.

The following specific issues and difficulties were found with the software/platform during the project:

<table>
<thead>
<tr>
<th>Issue</th>
<th>FIWARE Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocating public IP address for a host doesn’t work.</td>
<td>Cloud</td>
</tr>
<tr>
<td>MashupPlatform.http.makeRequest is a security threat. Essentially removes ajax same origin policy.</td>
<td>Wirecloud</td>
</tr>
<tr>
<td>Mashup platform as a development environment is painful. Making a change to a widget requires full widget upload</td>
<td>Wirecloud</td>
</tr>
<tr>
<td>Wirecloud Eclipse plugin fails on authentication while creating a wirecloud server: Invalid Authorization Request bad_request</td>
<td>Wirecloud</td>
</tr>
<tr>
<td>Tried to buy MapViewer widget, but got an error: Error ord() expected a character, but string of length</td>
<td>Wirecloud</td>
</tr>
<tr>
<td>Issue</td>
<td>Resolution</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Orion context broker always returns HTTP 200 also in case of an error. It is confusing. The documentation claims it to be RESTful, which is also confusing, It is not RESTful in any way.</td>
<td>Orion Context Broker</td>
</tr>
<tr>
<td>The NGSI-library provided with mashup lab wirecloud deployment didn’t respect alternative proxy settings so we needed to ship a custom NGSI-library with the product</td>
<td>NGSI Javascript library</td>
</tr>
<tr>
<td>Orion Context Broker JSON encoder does not work. It inserts newline-characters inside string values. E.g. { “attr” : “first line second line” }</td>
<td>Orion Context Broker</td>
</tr>
<tr>
<td>Orion context broker doesn’t handle updates with non-ASCII JSON correctly. JSON default encoding is UTF-8, it should work. Also, Non-ASCII XML won’t work if encoding is not explicitly declared. That should be fine, but no error is reported, which is confusing</td>
<td>Orion Context Broker</td>
</tr>
<tr>
<td>Sometimes remove widget -&gt; add widget In Wirecloud marketplace doesn’t actually change the running widget code. This happens randomly, often it works. When this happens, it is possible to download widget and check that the new widget code is actually in wirecloud, but the code that is run in mashup is for some reason the old removed version still.</td>
<td>Wirecloud</td>
</tr>
</tbody>
</table>

**Screenshots**
7 Las Palmas (Spain): Smart Port

7.1 Application 1: Smart Port - Wirecloud

7.1.1 Scope & Requirements
Smart Port was originally an application that was intended to provide an easy mechanism to monitor and retrieve historical data from geolocated sensors of the Port Authority of Las Palmas. Displaying information visually aids to better understand a system, so we decided to represent the sensors on a 3D map.

To accomplish the application objectives, we needed a supporting infrastructure that allows us manage the context of the elements. Orion Context Broker meets perfectly this requirement. The next step was to save historical data and Cosmos provides an easy integration with Orion Context Broker, furthermore this component allows us to work with big data easily.

We choose Wirecloud Platform as development environment because it allows us develop the application as components and later connect it straightforwardly, also provides an API to interact with the Orion Context Broker.

Finally we used the FIWARE Cloud Platform to host customs services because it gives us the possibility of create custom virtual machines without any complication and using this component our development was fully in the FIWARE ecosystem.

7.1.2 GEs used and Architecture
The generic enables used to develop this application were:

- Orion Context Broker
- Wirecloud Mashup Platform
- Cosmos Big Data
- FIWARE Cloud Platform

7.1.3 Infrastructure and equipment
In order to deploy our services, we use in this phase of the project the infrastructures provided by Wirecloud Platform and by FIWARE Cloud as seen in the diagram in the previous section.

Regarding sensors, we do not have direct access to them, so we needed to read the values from a database provided by the Port Authority of Las Palmas. At that time we only had access to information about current meters and meteorological stations.
7.1.4 Trials and Validation

The front end of the application was tested using Chrome and Firefox web browsers installed in Ubuntu Desktop, Windows 8 and OSX 10.5. Since some WebGL features are not supported neither IE nor Safari web browser, our application does not offer support to them.

To test the back end services, we also developed a series of scripts written in python that simulate all the data process.

While developing the application we found the following difficulties:

- NGSI API changed when Wirecloud was updated, so some fixes had to be made to have the application operative.
- Hard debug process: if a developer does not use Eclipse IDE as his environment for any reason, trying to debug our functionality required to create a new version of the widget every single time we wanted to test an error, and that was a little annoying and unproductive.
- Sometimes Wirecloud did not recognize the new version of a widget and, instead of the new one, the previous version was shown, so we lost time trying to guess if you had the last one in the workspace.
- The Ajax-based APIs have a lot of advantages, but also have disadvantages at debugging time: breakpoints did not work inside an onSuccess() / onFail() function within the Ajax petition, so we had to debug its contents by the old way, using prints.
- The Wirecloud Mashup Platform does not allow dynamic widget creation.
- Lack of advanced documentation about Wirecloud Platform.
- An API to make queries to Cosmos is not provided by Wirecloud Platform.
- In order to access to data stored in Cosmos from machines located outside the trusted network, we had to develop an intermediate service allocated in a FIWARE Cloud virtual machine.
- We had issues with Cygnus storing data inside Cosmos as JSON documents. The problem was that our Python services that access Cosmos remotely were only able to read CSV format. In order to surpass this problem, we modified Cygnus source code to adapt to our case and store in CSV format with an extra column to allow the identification of the records as a block.

7.1.5 Screenshots

7.2 Application 2: Smart Port version 2
7.2.1 Scope & Requirements

Once the first version of Smart Port was developed, we discovered several potential features so we decided to go one step further. This time we obtained much more data from the Port Authority of Las Palmas allowing us to really improve the functionality of our application. Now the application is not only a monitored tool of some sensors, it integrates some interesting features like:

- To provide a system to manage alerts and notifications based on triggered values of the sensors.
- Better representation of the data.
- To display where are located all nearby vessels and information about them.
- To know which vessels are going to arrive or depart today and information about them.
- To show POI on the map.

To perform these enhancements we have new requirements, which caused us to change our architecture and infrastructure. The first and biggest change was to leave the Wirecloud Platform to avoid its limitations about design and lack of dynamic behaviour. The alerts and notifications management system uses the Publish/Subscribe API provided by Orion Context Broker.

7.2.2 GEs used and Architecture

The generic enables used to develop this application were:

- Orion Context Broker
- Cosmos
- Big Data
7.2.3 Infrastructure and equipment

Due to new needs arising in our application, like a more dynamic behaviour or a better design, we decided to leave Wirecloud Platform. Currently the deployment of our application is as described in the previous section. The infrastructure consists of a dedicated server where all our services are located.

The Port Authority of Las Palmas provided us new sources of data with different mechanisms to access to them:

- Information about which vessel will depart or arrive each day, hosted in a FTP server.
- A vessel stream with updated information.
- New sensors like mareographs and buoys.

7.2.4 Trials and Validation

Like the previous version of the application, it was tested using Chrome and Firefox web browsers installed in Ubuntu Desktop, Windows 8 and OSX 10.5. Since some WebGL features are not supported neither IE nor Safari web browser, our application does not offer support to them.

To test the new back end services, we also developed more scripts written in python that simulate the new process. Although we have these test scripts, we are using real flow data in our production environment.

While developing the new version of the application we found the following difficulties:

- Once we had access to historical data, we tried to push them to Cosmos through Orion, but we found out that we were quickly overcoming Cygnus queue memory which leaded to discontinuity in the recovery. In order to avoid the memory overflow, we needed to slow down the data push from the script, which made the process longer (up to four days to upload months of data). On the other side, we also started uploading data directly to Cosmos, in order to avoid the overhead of Cygnus, since we were not interested in Orion format for non contextualized data.
- As we were receiving new data from the Port Authority, we soon started to get limited by the memory and the availability of the virtual machine, which motivated us to start migrating the services to a local physical machine.
- All data uploaded to Cosmos HDFS is readable by any user of the system and accessible without credentials through webHDFS protocol. As most of our data inside Cosmos cannot be directly published for security reasons, this became an issue we wanted to control with our own installation.
- We needed to implement a custom security access for access data stored in Orion with an easy solution. Furthermore, we needed to abstract to other developers how Orion works, so we developed an API that allows us to meet these requirements easily.
- While we were using our Orion API Smart Port application, we found some issues about caching data when a proxy is used for an AJAX petition, so we had to add a randomly generated parameter to the url. In our case, we use a timestamp.
- To replace the wiring features provided by the Wirecloud Platform we developed a JavaScript API that allows the communication among components.
- To avoid cross-domain reference problems we used a new proxy.
- Since we removed any limitation about the size of the historical data provided by Cosmos, the JSON response, which now contains all the historical data of a sensor, could generate a really big transaction. That produces a network overhead, so we are working in a new architecture to avoid this problem.
7.2.5 Open Data

Currently in CKAN instance at the University of Las Palmas de Gran Canaria we have the following organizations providing data:

- **Puertos de Las Palmas**, the Port Authority of Las Palmas.
- **Emalsa**, company in charge of part of the water cycle in the island of Gran Canaria.
- **University of Las Palmas de Gran Canaria**.

The datasets are compiled into five main groups, which are directly accessible from the welcome page.

For every group, we will list the datasets published:

<table>
<thead>
<tr>
<th>Group</th>
<th>Dataset Name</th>
<th>Data Source Names</th>
<th>Formats available</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>University</td>
<td>Photovoltaic ULPGC</td>
<td>2014/10/08 08:00</td>
<td>Excel file(.xls)</td>
<td>Hourly document with the readings from the photovoltaic panels installed at the ULPGC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2014/10/08 23:00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Water</td>
<td>Water abstraction Gran Canaria</td>
<td>Captaciones</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of Emalsa's water abstractions points in the island.</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Sewage pipes Gran Canaria</td>
<td>Tramos emisarios</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of Emalsa's sewage pipe network in the island.</td>
</tr>
<tr>
<td></td>
<td>Water treatment stations of Gran Canaria</td>
<td>Depuradoras</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of Emalsa's water treatment stations in the islands.</td>
</tr>
<tr>
<td></td>
<td>Sanitation pipes Gran Canaria</td>
<td>Ramales saneamiento</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of Emalsa's sanitation pipes network in the island.</td>
</tr>
<tr>
<td></td>
<td>Network pipes Gran Canaria</td>
<td>Tramos conducción</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of Emalsa's network pipes in the island.</td>
</tr>
<tr>
<td></td>
<td>Deposits Gran Canaria</td>
<td>Depósitos</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of Emalsa's water deposits in the island.</td>
</tr>
<tr>
<td></td>
<td>Sewer pipes Gran Canaria</td>
<td>Tramos colectores</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of Emalsa's sewer pipes network in the island.</td>
</tr>
<tr>
<td></td>
<td>Water supply</td>
<td>Datos de abastecimiento por zonas en Las Palmas de Gran Canaria</td>
<td>Excel file (.xls)</td>
<td>Summary of Emalsa's water supply for the city of Las Palmas de Gran Canaria.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tabla durezas Las Palmas de Gran Canaria</td>
<td>Excel file (.xls)</td>
<td>Summary of Emalsa's water hardness readings for the municipality of Las Palmas de Gran Canaria.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tabla durezas Santa Brígida</td>
<td>Excel file (.xls)</td>
<td>Summary of Emalsa's water hardness readings for the municipality of Santa Brígida.</td>
</tr>
<tr>
<td>Port</td>
<td>Data Source</td>
<td>File Format (s)</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Las Palmas</td>
<td>Weather stations Las Palmas</td>
<td>CSV (.csv)</td>
<td>Readings from weather stations installed in the city of Las Palmas de Gran Canaria.</td>
<td></td>
</tr>
<tr>
<td>Fuerteventura</td>
<td>CSV (.csv)</td>
<td></td>
<td>Readings from weather stations installed in the island of Fuerteventura.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Environment</th>
<th>Data Source</th>
<th>File Format (s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Las Palmas</td>
<td>Education Centres Gran Canaria</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of education centres in the island.</td>
</tr>
<tr>
<td>Fuerteventura</td>
<td>Morgues Gran Canaria</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of morgue centres in the island.</td>
</tr>
<tr>
<td>Fuerteventura</td>
<td>Markets Gran Canaria</td>
<td>Shapefile (.dbf, .shp and .shx) and geojson</td>
<td>Location of markets in the island.</td>
</tr>
</tbody>
</table>

In order to have all the information up to date in our CKAN platform, we developed a procedure which allows us to use the FileStore API provided by CKAN to update a resource each time a public register that must be upload to CKAN is inserted in our systems. Once it has been implemented, the behavior is something like:

1. New data is inserted in some of our databases.
2. A trigger is executed and it creates a new file from the table.

3. The trigger uploads the created file to CKAN using the FileStore API.

7.2.6 Trials and Validation

To test the procedure that automatically updates a CKAN resource when new data is available, we use a dummy dataset that has a correspondence with a dummy PostgreSQL database.

Problems with CKAN

The first problems we found were related to CKAN dependencies. We wanted to install the system in an Ubuntu Server 14.04 but the documentation only supported the installation in 12.04. The problem consisted that in the documentation there was no specification on the versions for the packages Solr and Jetty, in order to overcome it we needed to install CKAN in a 12.04 distribution, check the packages versions and then compile and install in our production server.

Once the base installation was done, we started customizing the site. The problem we found in this stage was that the documentation seemed to be a bit dispersed to get things done.
The official documentation about APIs provided for CKAN is incomplete, disorganized and hard to follow.
8 Lisbon (Portugal): Social and mobility applications

8.1 Introduction

The FIWARE Lisbon Use case Pilot is framed in the FIWARE FP7 project coordinated by Telefónica. The Lisbon consortium is composed by Alfamicro, DRI, Full IT and IPN and started its activities in May 2014 which were technically completed in September 2014 followed by refinements and the completion of the present document. This report aims at reporting the work done during the baseline of the project as also the results and dissemination activities undertaken by the partners\textsuperscript{5}.

In terms of roles and responsibilities, Alfamicro led the local coordination of partners, the liaison with the FIWARE team and the Lisbon Municipality, as also the dissemination activities. Likewise, led the definition of the vision and architectural design of the solutions for the Lisbon Pilot, data analysis and preparation and platforms integration with FIWARE.

DRI worked on the integration of the MyNeighbourhood platform with FIWARE as also on the production of dissemination material.

FULL IT analysed and prepared the data sets from Lisbon Municipality to publish in the FIWARE Lab cloud, working at the same time in the integration of the Lisbon Open Data platform with the FIWARE. Full IT also worked on the production of the dissemination material.

IPN worked towards integrating the Mobility Platform (OST) with FIWARE, namely by developing data connectors that would export/import Lisbon Municipality open data from and towards the common CKAN instance in FIWARE Lab. On top of having all data aggregated and centralized in FIWARE, IPN developed a web-based demo app to test and validate the integration mechanisms.

The structure of this report follows the recommendations of Telefónica and the inputs were collected and discussed with all partners of the consortium.

8.2 Scope & Requirements

The scenarios implemented in Lisbon took advantage on the integration of the solutions that already exist and are available in the local ecosystem, as a result of the development and implementation of different initiatives and projects, in particular, the CIP projects CitySDK, Citadel and MyNeighbourhood, as well as the Open Data platform from the Lisbon Municipality and the Mobility Platform from IPN. The figure below depicts the architecture the ecosystem integrated solution.

\textsuperscript{5} The partners’ description is presented in the Annexe 1.
This envisioned integrated solution was reflected in two scenarios which were experimented in Lisbon. The first one focused on the integration of the Open Data sets from several data sources available in Lisbon with the FIWARE platform, making use of the Generic Enablers (GEs) and the results of CitySDK\textsuperscript{6} and Citadel\textsuperscript{7} projects. Thus, this scenario aimed at integrating the Open Data of the Municipality of Lisbon with FIWARE Lab. To achieve this goal, the consortium has analyzed and prepared the datasets available in different formats and domains, such as Lisbon Participa Open Data Platform\textsuperscript{8}, CitySDK Tourism Open Data API\textsuperscript{9}, Lisbon ArcGIS API, places information from MyNeighbourhood platform and also Comboios de Portugal and Carris public transportation data available in the One Stop. Transport platform (OST)\textsuperscript{10}. The different data sets coming from these platforms were analyzed and prepared to be published in the FIWARE Lab cloud, enabling developers to re-use such data and create new applications and services.

The second scenario worked in the integration in FIWARE of different platforms, namely, the Open Data Platform, the generic platform for Open Data from Lisbon; the OST Mobility Platform developed and implemented by IPN which aims to bring together data providers, developers and end users in order to enable the development of innovative solutions so that a vibrant ecosystem is born and nurtured around open mobility data; and the MyNeighbourhood Platform build to support the implementation of Human Smart Cities vision framed in the MyNeighbourhood\textsuperscript{11} ongoing EC CIP project.

\textsuperscript{6}http://www.citysdk.eu/
\textsuperscript{7}http://www.citadelonthemove.eu/
\textsuperscript{8}http://www.lisboaparticipa.pt/pages/newApps.php
\textsuperscript{9}http://tourism.citysdk.eu/endpoints/lisbon/
\textsuperscript{10}https://www.ost.pt/
\textsuperscript{11}http://my-neighbourhood.eu/
MyNeighbourhood is a project integrated in the European Union’s CIP Programme, which involves 16 partners from seven European countries. MyNeighbourhood aims to create a new and ‘smarter’ conception of the ‘Smart City’ that focuses on people and their wellbeing rather than just Information and Communications Technology infrastructures. Technology enables Relationships, so MyNeighbourhood is a web platform - the MyN platform\(^{12}\) - that emerged to provide a technological environment that helps to restore a sense of belonging to the neighbourhood, where people share the same space, interests and needs.

Using the Living Labs approach, Gamification and Design Thinking tools, the MyN platform was built combining several technological solutions that respond to different users and their needs, in the context of a neighbourhood, especially through features related to the creation of an online environment where communities can emerge, gather and thrive, finding their common WINs.

The integration of existent platforms with FIWARE was the core scenario of the Lisbon Use case. To achieve this goal, partners focused on the development of solutions to enable the integration of the mentioned platforms which host data and services for the city, however with no connection between them. The proof of concept in this scenario consisted in enabling the communication between the platforms through FIWARE, as well as enhancing the interoperability to allow developers to create applications and services that can access to the data and features provided by each platform.

An example of a use case for this scenario would be to integrate the data from Mobility Platform with the Open Data from the city to create a demo application to visualize data from both platforms in the MyN platform. The solution was created using the selected Generic Enabler from the FIWARE Lab Catalogue along with the tools and APIs already created by each platform. All these activities were coordinated by Alfamicro with the support of Full IT, DRI and IPN.

To achieve the goals of the FIWARE Lisbon Case, regarding the integration with the MyN Platform four major steps were followed:

4. **Scope Definition** - identify the challenge: how can MyN platform benefit with the integration with existing solutions in the local ecosystem (like CitySDK, OST, Citadel)? What can MyN offer that can be useful for other platforms?

5. **Design technical solution** – identify what are the available solutions to integrate MyN Platform with FIWARE, and design the technical solution taking in consideration the scenario goals, nature of the information, etc.

6. **Trials and validation** – implement the agreed-upon requirements, in order to test and demonstrate the integration scenarios;

7. **Demo materials** – develop demo materials to explain and demonstrate the Lisbon use case: demo applications, presentations, scripts and video. Powerful tools – brochures, concept video and demo

\(^{12}\) [http://www.my-n.eu/](http://www.my-n.eu/)
video were produced to disseminate the project not only in Europe but also worldwide, particularly in Brazil.

Accordingly, the MyN platform was used to understand how FIWARE could help to overcome the challenge of empowering the emerging communities as also technically supports the Human Smart Cities ecosystem. It is very important to highlight that the MyN platform is the core part of the Human Smart City movement and vision and it was on this context that FIWARE was chosen to be the host and the cloud to support all the Human Smart City technological answer. Therefore, it was used one of MyN platform features to this integration scenario: the MyN Places and a testing neighbourhood: the Belém Neighbourhood in the City of Lisbon.

MyN Places is a special feature dedicated to a directory of the neighbourhood points of interest (POI) enriched with information generated by citizens. It aims to promote local business, gain knowledge from users feedback and share stories about these places. This component aims to transport the physical space to the digital environment.

**Identify the Challenge:**

- MyN is a user-generated content platform. This means that the platform lives on the content created and inserted by the users;
- The MyN Places module aims to solve day-to-day problems, get users to know their neighbourhood and provide access to existing places and services.
- As a new neighbourhood is created, little information is there. How can we transform a “new born” platform neighbourhood in a reliable and valid source of information for the community?
- How can MyN give back to other platforms for the cities and developers? MyN could enrich cities open data, as it provides mechanisms that incentive user to share stories and experiences. The application of gamified logics on the platform helps to enrich data by the stimulation of the user experience.

**Solution – FIWARE and Open data – Why?**

- Open data already exist in the city, spread in different platforms: data that can be freely available to everyone to use and republish as they wish;
- Many projects and initiatives are emerging that promote the creation of web and mobile civic applications to enable better services, lower costs and improved transparency.
- Open data play an important role in the Smart Cities and Communities Initiatives.

**Setting the Goals**

For the Lisbon Case, **two goals** were set:

1. Enable the communication between existing open data platforms that provide information in the context of the cities, and that are relevant at the neighbourhood level, USING FIWARE. The platform that centralizes all the described architecture is the MyN platform, supported by the FIWARE cloud.
2. Give back: open MyN data, for cities and developers.

8.3 Architecture and GEs used

The FIWARE Lisbon Case architecture is presented in the figure below:

Figure 11 - FIWARE Lisbon Case Architecture
Summarizing the technical architecture of the FIWARE Lisbon use, it is based on the following points:

3. MyN dataset reader from FIWARE common instance, incorporating into the MyN platform front end (MyN places tool) visualizations of FIWARE datasets.
4. Data integration in MyN features, using the inserted data from City SDK;
5. Using FIWARE common dataset format;
6. Data is imported in bulk in a regular fashion, entire data are copied from FIWARE to MyN;
7. Connector from FIWARE to MyN assembled and working in a local instance of CKAN.
8. Allowing data deployment into FIWARE in a common CKAN instance, therefore transferring data between instances;
9. Use of the Lisbon open data from the City SDK API.
10. City SDK API data publication into FIWARE
11. With the help of Citadel converter tool from W3C POI format to JSON CKAN data store.
12. OST data publication into FIWARE.
13. OST demo app dataset reader from FIWARE of the inserted mobility and transport data, allowing calculation engine of the OST demo application to perform distance, path and proximity calculation, using FIWARE common dataset format.
14. Insertion in MyN of a calling link to the OST demo app to trigger the path and proximity calculation.

As shown in the above picture, OST publishes datasets into the CKAN instance. For this procedure, a worker (FIWARE Lisbon Connector) was created to periodically fetch data from OST APIs and import it using CKAN’s Data Store APIs. This worker is open-source, available on GitHub, and is also responsible for the data import procedure back to OST, in order to demonstrate the platform’s full integration.

Part of the data published in CKAN was not available in OST databases before this project started. Before making it available at OST, some work had to be done thus data could be normalized, validated, and converted to the desired format (GTFS – General Transit Feed Specification). Only after those steps, the previously described connector could fetch it from OST and publish it in CKAN.

The following table, schematizes the design of the technical solution emerged from the presented diagram.
<table>
<thead>
<tr>
<th>#</th>
<th>Goal</th>
<th>Implementation Description</th>
<th>Activities identified</th>
<th>GE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Enable the communication between existing open data platforms</td>
<td>Use data from FIWARE in MyN – <strong>CitySDK (Tourism domain)</strong>&lt;br&gt;Assumptions: an open data dataset from CitySDK is available in FIWARE</td>
<td>• Connect MyN with FIWARE to collect open data&lt;br&gt;• Develop MyN Connector to CKAN (Reader)&lt;br&gt;• Present data in MyN</td>
<td>CKAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use data from FIWARE in MyN – <strong>OST (Mobility Domain)</strong>&lt;br&gt;Assumptions: an open data dataset from OST is available in FIWARE</td>
<td>• Connect MyN with FIWARE to collect open data&lt;br&gt;• Develop MyN Connector to CKAN (Reader)&lt;br&gt;• Present data in MyN</td>
<td>CKAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Connect POI in FIWARE with OST application – connect each POI to <strong>OST Demo Application</strong> (passing query string variables of a geographical point and metadata information), that offers an engine to perform distance, path and proximity calculation</td>
<td>• Connect POI to OST Demo application;</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Make MyN data available in FIWARE</td>
<td>Publish data from MyN platform to CKAN, one of the components of FIWARE architecture, so that it can be used by other systems.&lt;br&gt;In this scenario, MyN will be providing data to FIWARE.</td>
<td>• Define MyN Dataset&lt;br&gt;• Develop MyN Connector to CKAN (Publisher)&lt;br&gt;• Integrate MyN with CKAN – publishing data</td>
<td>CKAN</td>
</tr>
</tbody>
</table>

*Table – Designing the FIWARE Lisbon use case technical solution*
8.3.1 FIWARE Generic Enablers Research

Although in the beginning of the project some work was done around Orion Context Broker, the Generic Enabler used on the FIWARE Lisbon Case scenario was a CKAN instance, deployed on FIWARE LAB. From the performed analysis and the work done with the project technical coordination, CKAN was selected as the suitable solution for the proposed goal. This selection was discussed in the workshop held in Paris in the 12th June 2014 with all Lisbon Use Case partners and Telefónica where the use of CKAN was agreed with the support of Telefónica.

CKAN is part of the FIWARE architecture and is a powerful data management system that makes data accessible – by providing tools to streamline publishing, sharing, finding and using data. CKAN is aimed at data publishers, wanting to make their data open and available. The several datasets were published there, with most of them being converted to a common JSON format in order to be easily integrated into MyN platform. For proof of concept validation, a web application was developed to fetch data from the CKAN instance, to display it on a map and to use it for the journey planning operations.

It is important to state that during this activity the consortium has made a research on the Generic Enablers available on the FIWARE platform, namely the context related generic enablers, extracting information about how they work and detailing the requirements to integrate with such components of the FIWARE ecosystem.

For the further developments of the MyN platform, it is expected the exploitation and also the use of more FIWARE Generic and Specific Enablers, in order to fulfil the functionalities and ambitious features of the platform specially in what concerns the services for citizens and urban city management provided by the MyN platform.

8.4 Infrastructure and equipment

As mentioned before, the central FIWARE component is a CKAN instance, which was deployed on FIWARE LAB and can be accessed at the following URL: http://lisboncase-ckan.fullit.pt. The open-source project FIWARE Lisbon Worker, which performs the data-publishing tasks, was not deployed on a specific machine. However, it can be virtually run on any UNIX machine with Internet access. Regarding the OST platform, it was essential in the project’s architecture for two things besides data provision. The journey planning service could not be detached from the OST platform, since it is a complex application that cannot be updated in real-time (it needs to process several GTFS archives and a large OpenStreetMap file, containing street and road information about Portugal) and currently has some hardware requirements, which are not very easy to maintain (16GB of RAM).
The other service that was very useful for the project was the One.Stop.Transport’s app marketplace, which was used to host the demonstration application. This way, the demo application can benefit from some of the OST’s Javascript APIs and a separate machine just for the application hosting is not necessary. Given the feedback provided in Munich, IPN has since updated OST’s marketplace, in order to avoid authentication when running applications. Without forcing authentication, anyone can use the applications (including the FIWARE Lisbon Pilot) but will still need an account to install or rate them.

Moreover, no sensors or devices were used in the FIWARE Lisbon Use Case. Nevertheless, in the future, the MyN platform will integrate services focused for example in the city sensorization, namely environmental as also traffic monitoring sensors.

### 8.5 Open Data

Different types of Open Data sets were worked on the FIWARE Lisbon case as described below.

Lisbon Open Data is available from two sources: the municipality Open Data itself and from the CitySDK project.

- **Open Data from the Lisbon Municipality** - The chosen datasets cover a wide broad type of information about Lisbon, focused on data containing relevant information in the tourism field as also public equipment and facilities, commercial and public services available in each parish and its characteristics. As examples: sports facilities, hotels and accommodation, restaurants, public libraries, museums, monuments among others. This information is available for all the city but for the proof of concept, as stated, the Belém neighbourhood was used.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Field type</th>
</tr>
</thead>
<tbody>
<tr>
<td>field_poi_id</td>
<td>Text</td>
</tr>
<tr>
<td>field_neighbourhood</td>
<td>Text</td>
</tr>
<tr>
<td>field_title</td>
<td>Text</td>
</tr>
<tr>
<td>field_category_places</td>
<td>Text</td>
</tr>
<tr>
<td>field_body</td>
<td>Text</td>
</tr>
<tr>
<td>field_photographs</td>
<td>JSON</td>
</tr>
<tr>
<td>field_website</td>
<td>Text</td>
</tr>
<tr>
<td>field_email</td>
<td>Text</td>
</tr>
<tr>
<td>field_phone</td>
<td>Text</td>
</tr>
<tr>
<td>field_location_latitude</td>
<td>Text</td>
</tr>
<tr>
<td>field_location_longitude</td>
<td>Text</td>
</tr>
</tbody>
</table>
The fields used were POI ID, Neighbourhood - to identify which parish POI belong to, title, place’s category, body information, photographs, website of POI, contact information - email, phone and address, location information - latitude and longitude. All the field types are text, except photos which are JSON, in order to accommodate an unlimited number of referenced images.

CitySDK API - It was decided to load the data from CitySDK POI’s, once the information was already enriched, namely with pictures. The dataset was imported from CitySDK using their API and stored in a CKAN instance hosted on FIWARE servers, and used commonly by all partners to store the datasets to be integrated in MyN. It was developed a CKAN harvester that grabs the information from CitySDK through its API, convert it to the formats agreed with the consortium, and store it in CKAN. Once we are working with static data - not being updated regularly – the harvester is run manually. The CitySDK dataset is published in JSON and CSV formats.

Regarding the OST platform, the datasets integrated in the project were the following:

- **GTFS data (Comboios de Portugal)** - Public transportation information about CP (nationwide railroad agency) regarding train stations, routes, trip schedules and the agency’s contact details. Following the globally used *de facto* standard for public transportation (*General Transit Feed Specification*), it was published as a set of CSV (*comma-separated values*) files, currently available on the [FIWARE Lab portal](http://lisboncaseckan.fullit.pt/dataset/fiware-citysdk-lisbon-v3).

- **GTFS data (Carris)** - Public transportation data about Carris (the Lisbon bus agency) regarding its contact details, bus stops, routes, shapes and trip schedules. Following the same standard as the previous dataset (CP), it was published as a set of CSV files, currently available on the [FIWARE Lab portal](http://lisboncaseckan.fullit.pt/dataset/fiware-citysdk-lisbon-v3). This dataset was not available in OST before this project, and it took some effort to get access to, in order to parse and validate it.

- **Lisbon Places from OST (bus and train stops)** - On a JSON format established by the whole consortium, the bus and train stations (from Carris and CP) provided by the OST APIs, are available on the [FIWARE Lab portal](http://lisboncaseckan.fullit.pt/dataset/fiware-citysdk-lisbon-v3) and are one of the places’ categories used by the demonstration application. The original data was in the GTFS format (*stops.txt*) but has been converted to this specific JSON for easier integration with the MyN platform.

All these datasets are exported from OST to CKAN by the [FIWARE Lisbon Connector](http://lisboncaseckan.fullit.pt/dataset/fiware-citysdk-lisbon-v3), as already mentioned in this report, which can be run periodically (hourly, daily, weekly, etc.), keeping CKAN with the most up-to-date information. This same worker can import the first two datasets (GTFS data) back to One.Stop.Transport. During the pilot’s development, datasets were published in a shared CKAN instance.
between the consortium partners. Telefonica then suggested moving the data to a global instance instead, which IPN eventually made during the last month of the project.

For the publication in the MyN platform, a common dataset structure was defined for the open data integration in the Lisbon Use case. MyNeighbourhood, CitySDK and Citadel from other sources are publishing data using the same structure:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field_poi_id</td>
<td>Int</td>
<td>Yes</td>
<td>POI unique identifier</td>
</tr>
<tr>
<td>field_neighbourhood</td>
<td>Text</td>
<td>Yes</td>
<td>Neighbourhood</td>
</tr>
<tr>
<td>field_title</td>
<td>Text</td>
<td>Yes</td>
<td>Place’s Name (POI)</td>
</tr>
<tr>
<td>field_category_places</td>
<td>Text</td>
<td>Yes</td>
<td>Place’s category - see Category List</td>
</tr>
<tr>
<td>field_body</td>
<td>long text</td>
<td>Yes</td>
<td>Place’s Description</td>
</tr>
<tr>
<td>field_photographs</td>
<td>Text (json encode)</td>
<td>No</td>
<td>Place’s pictures. 'public’ url’s</td>
</tr>
<tr>
<td>field_website</td>
<td>Text (json encode)</td>
<td>No</td>
<td>Place’s URL. Up to 3</td>
</tr>
<tr>
<td>field_email</td>
<td>Text (json encode)</td>
<td>No</td>
<td>Place’s E-mail. Up to 3</td>
</tr>
<tr>
<td>field_phone</td>
<td>Text (json encode)</td>
<td>No</td>
<td>Place’s Phone. Up to 3</td>
</tr>
<tr>
<td>field_location_latitude</td>
<td>Decimal</td>
<td>Yes</td>
<td>Place’s Latitude latitude (decimal degrees)</td>
</tr>
<tr>
<td>field_location_longitude</td>
<td>Decimal</td>
<td>Yes</td>
<td>Place’s Location latitude (decimal degrees)</td>
</tr>
<tr>
<td>field_location_address_first_line</td>
<td>Text</td>
<td>Yes</td>
<td>Place’s address - line 1</td>
</tr>
<tr>
<td>field_location_address_second_line</td>
<td>Text</td>
<td>No</td>
<td>Place’s address - line 2</td>
</tr>
<tr>
<td>field_location_city</td>
<td>Text</td>
<td>Yes</td>
<td>Place’s City/town</td>
</tr>
<tr>
<td>field_location_country</td>
<td>Text</td>
<td>Yes</td>
<td>Place’s Country</td>
</tr>
</tbody>
</table>

*Table: Common dataset structure*

8.6 Trials and Validation

*Implementation of Scenario 2.1: Using data from FIWARE in MyN*

In this scenario, MyN is using data available in FIWARE from other open data applications:

- **Goal:** Integrate with MyN Places data collected from FIWARE.
- **Selected datasets:**
To achieve this goal, the following activities were performed:

- Prepare development environment and configuration;
- Determine dataset format to be consumed from CKAN (Places Dataset);
- Update MyN Vagrant project to use last version of MyN;
- Extend CKAN FIWARE Connector to support data extraction from CKAN;
- Perform integrated testing.

The following diagram shows the components identified:

*Figure 12 - MyN reading open data from FIWARE*
**Implementation of Scenario 2.2: Making MyN data available in CKAN**

In this scenario, the goal was to implement the integration between MyN and CKAN, allowing the publication of public geo-referenced data, namely the places that exists on the platform to the FIWARE platform.

Having this information published through the FIWARE platform allows any developer to develop applications or other solutions that will consume this data through FIWARE using a generic API.

The following diagram shows the components identified:

![Diagram of MyN data publishing in FIWARE](image)

**Figure 13 - MyN data publishing in FIWARE**

The activities performed were:

- **Setup CKAN virtual environment**: Vagrant was used to allow the project team developers to recreate a virtual environment, with a CKAN installation. Vagrant is an open-source software for creating and configuring virtual development environments.
- **Setup MyN virtual environment**: Vagrant was used to allow the project team developers to recreate a virtual environment, with a MyN platform installation. A copy of the current database of MyN is completely loaded in this installation.
- **MyN Connector Development (CKAN Publisher)**: During this activity we developed the MyN connector for FIWARE, using the existing specifications for FIWARE context aware producers. This activity included:
  - MyN Dataset specification
  - MyN Dataset creation
Future Internet Core Platform

- Create front-end interface for MyN Connector configuration
- Create base module to be executed using Cron
- Map fields from MyN dataset to CKAN
- Integration with CKAN.
- Testing
- Create installation files
- Document deployment guidelines.

- **Connect MyN to common CKAN instance**

Test the integration with the common instance of CKAN, to be used by the *consortium* of the Lisbon Case ([http://lxcase-test.fullit.pt/](http://lxcase-test.fullit.pt/)).

- **Integrated tests**

After the implementation of the described activities, we were able to test the following scenarios:

**Test scenario 1:**

- Add information to MyN Places (ex: place 1, place2);
- Activate the MyN Connector and configure CKAN connection;
- Execute MyConnector publishing function;
- Check CKAN and validate that the MyN Dataset is published in CKAN;

**Test scenario 2:**

- Change information to MyN Places (ex: place 1);
- Execute MyConnector publishing function;
- Check CKAN and validate that the MyN Dataset is published in CKAN, and place 1 was updated;

The information displayed in CKAN is displayed in tabular form, but also in a map, as the dataset contains geo-referenced information.

As previously stated, a web application was created and published on the *OST marketplace* for concept validation. The application was based on an *open-source template* for building simple and elegant web mapping applications with Bootstrap and Leaflet and provides the following features:

- Fetch JSON datasets from CKAN and display them on a map, allowing filtering and search on a sidebar;
- Display details about each place when one’s marker is clicked;
- Provide a “*How to get there?*” link for each place that invokes OST’s journey planner and draw the best itinerary from the user’s position to the place’s location;
- Display the planned journeys’ step-by-step information on a modal window, for more details about the itinerary;
- Allow a given coordinates tuple as an URL parameter, to perform the journey planning operation but with the received latitude, longitude tuple as a destination point, instead of a given place’s position. This URL parameter is needed for the MyN integration with the demo application;
The consortium partners provided feedback in order to improve the application’s usability, functionality and to guarantee all requirements were fulfilled after testing and validating the prototype. In the following pages, several screenshots about this prototype can be found.

Figure 14 - Demo Application hosted on the OST marketplace
Figure 15 - Points of Interest displayed on the Demo App

Figure 16 - Point of Interest details
Figure 17 - Journey planning results displayed on the map

Figure 18 - Journey planning results displayed as step-by-step instructions
Future Internet Core Platform

Figure 19 - Demonstration Application about and credits page
8.6.1 Demo materials

- MyN Demo Neighbourhood: for the Lisbon Use case, a new neighbourhood was created for demo purposes: Belém – Lisbon.
  - Create content to populate neighbourhood;
  - Collect members to join the neighbourhood;
Collect information relevant for the neighbourhood;

- **MyN Demo Script:** a step-by-step demonstration was documented, in order to perform a live demonstration of DRI activities in the Lisbon Case.
- A dissemination video was prepared to explain the concept of the FIWARE Lisbon Case – see [https://vimeo.com/108239117](https://vimeo.com/108239117);
- The Lisbon Case was present in the FIWARE event in Munich (17-09-2014). The concept of the Lisbon Case was presented using presentation.
- Preparation of video with a screen recording of the Lisbon Case demonstration

![MyN Belém Neighbourhood](https://vimeo.com/114122840)

*Figure 21 - MyN Belém Neighbourhood*
8.7 Feedback on the FIWARE platform

It is common opinion of the Lisbon consortium members that the FIWARE Lab and Cloud have performed as strong platforms for the project implementation. The various and efficient tools really speed away the tasks involved in setting up server virtual machines and software services. The security system is also remarkable and works efficiently. The integration of CKAN in the FIWARE Cloud was made without major questions, performing perfectly as a collaborative data hub between the members of the FIWARE Lisbon Case team.

Regarding suggestions and questions as well on the FIWARE platform, the Lisbon consortium has come up with some points as follows:

- CKAN is (still) too hard to install, especially with relevant plugins such as DataStore. There ought to be a better way of creating CKAN instances using Vagrant / Docker / Packer images and make them easy to deploy.
- CKAN APIs usage documentation could also be improved in order to ease the application development, since there seems to be some ambiguity amongst procedures with the same goal (e.g. data insertion and update).
- It would be useful to have mechanisms for forking, adapting and deploying modified codebases of FIWARE enablers.
- Some of the generic enabler implementations are not open source, and nowadays no entrepreneur wants to have his/her business rely on closed sourced, proprietary software. Also, could there be more open-source components available as images in the FIWARE Lab cloud? The open-source community often flocks organically to certain frameworks / products and it is very hard not to have dependency on those. Therefore, it could be interesting to create images as well as an easy mechanism for launching them, the way Heroku or DigitalOcean do.
- Is there going to be Docker support? This would be very attractive to developers because it's emerging as the standard de facto for publishing apps/components.
- From what we have been informed, TestBed access needs a fixed IP to be accessed. Since our team is composed by several members working in different places, it was never useful for us to engage in that access.

Regarding this project specifically, the difficulties faced with FIWARE Software and Platform occurred mainly in the initial phase, during CKAN’s installation and configuration procedures. Another complication occurred near the prototype’s final deadline due to FIWARE’s downtime.
8.8 Conclusions

The work reported on this document is the outcome of a strong collaboration between all the members of the Lisbon team which met weekly since the beginning until the end of the project activities. The workshop held on 12th June 2014 in Paris as also the presence in the 2nd European Conference on the Future Internet during the 17th and 18th September 2014 in Munich were very important steps regarding the maturity of the Lisbon Use Case Vision as also its acceptance and dissemination.

It was made clear that the ecosystem envisioned FIWARE Lisbon Pilot can be the technical answer for the Human Smart Cities Vision. The next steps should be the further exploitation of the use of other generic enablers such as context broker and authentication to integrate in the MyN Platform as well as a deeper integration of FIWARE and MyN as part of the strategy for the Human Smart Cities not only at the European level but also in Latin American countries, namely in Brazil, where the MyN platform and the Human Smart Vision are being already tested and implemented.

For this further approaches, the FIWARE Lisbon Use case consortium is at the disposal of the FIWARE team in order to achieve a wider dissemination of the FIWARE vision and most important its real adoption by cities and its citizens.

As stated, the work described in this deliverable was the result of a collaborative team dealing with different experiences, guided and coordinated by the clear vision and strategy of the coordinator, to the Future Internet platform, methodologies and tools as the ICT platform for the European municipalities. Coherence and consequent vision was achieved through technical and coordination meetings. It was also very important the excellent collaboration with the FIWARE team at Telefónica. Alfamicro also represented Telefónica’s position in several meetings with the Lisbon Municipality and the Portuguese Government.
9 Lleida (Spain): Accessibility and public transport

9.1 Scope & Requirements

The main objective of the project was to introduce and connect mobility information from the Open Data available on the city council portal to the FIWARE platform, including several datasets and the information of the municipal bus network with real time information of the buses.

9.2 GE used and Architecture

For the Open Data datasets a connection from the City Council Open Data database was created to FIWARE LAB Orion Context Broker instance.

All 12 datasets can be found in the url: https://data.lab.fiware.org/organization/lleida

9.3 Open Data sets

Some of the following information is in Spanish, as the name of the fields are in this language, facilitating the developer’s comprehension of the data. A boldface translation in the most important parts is included.

9.3.1 Public parking slots reserved for people with reduced mobility.

This dataset contains a list of the different parking slots for people with reduced mobility on the streets of Lleida.

Contains two fields

- Address, with name and number of the street.
- Number of slots, with the number of slots in this place.

9.3.2 Cycle Parking

This dataset contains a list of the different places with public cycle mounts.

Fields

- Street, name of the street
• Number, number of the street
• Number of elements, number of cycles allowed on this place.
• Signal: if there’s a public sign for the cycles park.
• Equipment: notes in what public equipment is located the cycle parking
• Observations.

9.3.3 Accessibility dataset from public local in Lleida city.
This open dataset contains the accessibility information from many public locals in Lleida. This information is compiled with a crowdsocial effort through the eAccesible project from Lleida City Council.

http://aplicacionsweb.paeria.es/eAccessiblePublicWeb/faces/portada.jsp?lang=es&local=0

Actually there’re 6 different forms, each one corresponding to a kind of local.

- Bars and restaurants
- Pharmacies
- Hotels
- Banks
- Public locals
- Shops

The forms have the following blocs:

- Kind and name of the local.
- Location
- Accessibility questionnaire upon the kind of local.
9.3.4 Streets and public transport

- Street name database from: Lleida city with all the streets in Lleida
- Bus lines: Map of the different bus lines of the city
- Bus stop map: Map with the different stops the bus line mesh has in Lleida.
- Traffic lights with acoustic signals or buttons: Listing of all traffic lights that have request button or acoustic signals.
- List of accessible dumpsters (public trashcans on the street)
- Bus lines stops: This open data set contains the listing of all the different bus lines and stops in the city.
- Bus lines: This dataset contains the listing of all bus lines of the city.
- Bus stops: This dataset contains the listing of all bus stops of the city.
- Bus line transfers: This dataset contains the listing of all bus stops transfers of the city
9.3.5 Real time bus information

Currently, there is a web service with the local city bus contractor to reach the real time bus information. The web service has a number of different interconnected services to reach this info, as many actors are involved, both from PPAA and the contractors.

There are 3 operators: TMB, ATM and Tugsal, and one integrator, JCDecaux. Tugsal is the operator that has the Lleida local bus public contract.

On our actual system we have been integrating with FIWARE those 13 different calls we have on the web service: getAllArrivalTimes, getArrivalTimesByStopCode, getArrivalTimesByStopCodeAndLineCode, getArrivalTimesByMultiStopCode, getAllRoutes, getRoutesByStopCode, getRoutesByMultiStopCode, getPlannedMessages, getOnlineMessages, getPlannedMessagesPublication, getOnlineMessagesPublication,getAllStopDescriptions, getStopDescriptionsByMultiStopCode. The current system is being replaced, being the integration developments stalled until the new system is accessible.
9.4 Trials and Validation

The datasets on the FIWARE LAB were tested first internally and then by exposing them to developers in our dissemination activities.

Among other we had a Hackathon where 5 teams developed applications, 3 of them finished on time the projects and were awarded, as we show up in the next “Dissemination activities report” document.
10 Logroño (Spain): Open Data and Irrigation management

10.1 Scope & Requirements.

The main objective of this project is to integrate the updated and provided information by the Major of Logroño to the FIWARE platform, so this information will be accessible by the citizens from the Major’s web platform. For this purpose, the APIs of the CKAN will be used in order to store and give the set of data. At the same time, the Orion Context Broker’s global instance will be used to integrate the data of the weather station of the City in real time. Later, this instance will be integrated in CKAN using a walkway with the Orion Context Broker.

Also, the data published from the weather Station has been integrated into the irrigation’s control services, allowing to these services integrate any future weather station.

10.2 GEs Used And Architecture

The FIWARE generic enablers are depicted in the used in this project were:

- Orion Context Broker
- CKAN
10.3 Open Data

The Major has a number of set of data which they want to make accessible to the citizens. For this reason, they have decided to implement a system that lets incorporate these data in the FIWARE platform in order to put it available to the citizens and/or developers.

These data come from different departments of the Town Council, such as, environmental, strategic, operating and support. (For example: weather stations, public security services, sport services, etcetera)

In addition, these set of public data are available in XLS and XML format. Then these files are processed and converter to CSV format in order to integrate it in the Generic Enabler called CKAN.

Example of the integration of the data from the GIS (Geographic Information System) in CKAN.

A fragment of a XML file which contains information from the potable sources of the City of Logroño, is showed:

```xml
<?xml version="1.0" encoding="windows-1252"?>
<Explorador size="338">
  <FeatureMember typeName="fuente_jardines">
    <FeatureMember typeName="fuente_jardines" identifier="keysZ8234149">
      <boundedBy>
        <Box>
          <coordinates>546099067,700534598 546099067,700534598</coordinates>
        </Box>
      </boundedBy>
      <swidy:relationshipProperty typeName="zona_verde" count="1" typeNameExternal="Zona Verde">
        <FeatureMember typeName="zona_verde" typeNameExternal="Zona Verde" identifier="keysZ8232718"/>
      </swidy:relationshipProperty>
      <property typeName="codigo" type="string" typeNameExternal="Código de fuente">189001</property>
      <property typeName="tipo_fuente" type="string" typeNameExternal="Tipo de fuente">Bebedera</property>
      <property typeName="modelo_fuente" type="string" typeNameExternal="Modelo">Urbana</property>
      <property typeName="marca_fuente" type="string" typeNameExternal="Marca de fuente"/>
      <property typeName="numero_grifos" type="string" typeNameExternal="Número de grifos">1 grifo</property>
      <property typeName="fecha_instalacion" type="date" typeNameExternal="Fecha de instalacion">2000-01-01</property>
      <property typeName="cartucho" type="string" typeNameExternal="Cartucho ?">False</property>
      <property typeName="incidencia" type="string" typeNameExternal="Incidencia?">False</property>
      <property typeName="observaciones" type="string" typeNameExternal="Observaciones"/>
      <position type="Position">Point ID:keysZ1138732X47217772X8234150" swidy:world="01worldZ0" src="spain_etrs89_utm30_mnn">
        <coordinates>546099067,700534598</coordinates>
      </position>
    </FeatureMember>
  </FeatureMember>
</Explorador>
```
Here are the attributes which are going to be integrated:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>codigo</td>
<td>Unique identifier of the source.</td>
</tr>
<tr>
<td>tipo_fuente</td>
<td>Kind of source: potable or ornamental.</td>
</tr>
<tr>
<td>modelo_fuente</td>
<td>Model of source: urban, traditional, Georgian,</td>
</tr>
<tr>
<td>marca_fuente</td>
<td>Manufacturer.</td>
</tr>
<tr>
<td>numero_grifos</td>
<td>Number of water taps.</td>
</tr>
<tr>
<td>fecha_instalacion</td>
<td>Installation Date.</td>
</tr>
<tr>
<td>cartucho</td>
<td>Boolean value which indicates if the source has water filter.</td>
</tr>
<tr>
<td>incidencia</td>
<td>Boolean value which indicates if the source has some incident.</td>
</tr>
<tr>
<td>observaciones</td>
<td>Description of the possible incident.</td>
</tr>
<tr>
<td>latitud</td>
<td>Geographic coordinates.</td>
</tr>
<tr>
<td>longitud</td>
<td>Geographic coordinates.</td>
</tr>
</tbody>
</table>

An example of JSON content used to upload the information to the CKAN, is showed:

```json
{
    "resource_id": "RESOURCE-ID",
    "records": [
        {
            "zona_verde": "",
            "codigo": "189001",
            ...
        }
    ]
}
```
"tipo_fuente":"Bebedera",
"modelo_fuente":"Urbana",
"marca_fuente":"",
"numero_grifos":"1 grifo",
"fecha_instalacion":"2000-01-01",
"cartucho":"False",
"incidencia":"False",
"observaciones":"",
"posicion":"546099057,700534598",
"drawing":"",
"LAT":"42.4557743730193",
"LON":"-2.43934880573283"
}
"fields": [  
  
  {"id":"zona_verde",
  "type":"text"
  },  
  
  {"id":"codigo",
  "type":"text"
  },  
  
  {"id":"tipo_fuente",
  "type":"text"
  },  
  
  {"id":"modelo_fuente",
  "type":"text"
  },  
  
  {"id":"marca_fuente",
  "type":"text"
  },  
  
  {"id":"numero_grifos",
  "type":"text"
  },  
  
  {"id":"fecha_instalacion",
  "type":"text"
  },  
  
  {"id":"cartucho",
  "type":"text"
  },
One of the weather stations has data which are updated in real time, using Orion Context Broker GE.

An example of the information before update it to Orion Context Broker is showed:
The attributes are as follows:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp</td>
<td>Measurement Date.</td>
</tr>
<tr>
<td>lluvia_actual</td>
<td>Litres per square meter of rain registered.</td>
</tr>
<tr>
<td>temperatura</td>
<td>Temperature in degree centigrade registered.</td>
</tr>
<tr>
<td>ETO</td>
<td>Value of the evapotranspiration equation.</td>
</tr>
<tr>
<td>presion</td>
<td>Atmospheric pressure in Millibar registered.</td>
</tr>
<tr>
<td>humedad</td>
<td>Percentage of humidity registered.</td>
</tr>
<tr>
<td>radiacion</td>
<td>Solar radiation registered.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>velocidad_viento</td>
<td>Maximum wind speed.</td>
</tr>
<tr>
<td>velocidad_viento_media</td>
<td>Average wind speed.</td>
</tr>
<tr>
<td>direccion_viento</td>
<td>Direction in wind degrees.</td>
</tr>
<tr>
<td>lluvia_24h</td>
<td>Accumulated water in the last 24 hours.</td>
</tr>
<tr>
<td>porcentaje_calculado</td>
<td>Percentage of calculated irrigation water.</td>
</tr>
<tr>
<td>porcentaje_real</td>
<td>Percentage of real irrigation water,</td>
</tr>
</tbody>
</table>

Here is an example of the JSON content used in order to upload the information to the Context Broker.

```json
{
  "contextElements": [
    {
      "type": "MeteoLo",
      "isPattern": "false",
      "id": "MeteoLo",
      "attributes": [
        {
          "name": "timestamp",
          "type": "string",
          "value": "02/12/2014 17:10:00"
        },
        {
          "name": "lluvia_actual",
          "type": "liter",
          "value": "0"
        },
        {
          "name": "temperatura",
          "type": "centigrade",
          "value": "11.407407"
        },
        {
          "name": "ETO",
          "type": "float",
          "value": "1.391361"
        }
      ]
    }
  ]
}```
"name": "presion",
"type": "mmHg",
"value": "962.819093"
},
{
"name": "humedad",
"type": "percentage",
"value": "74.666667"
},
{
"name": "radiacion",
"type": "float",
"value": "7.666667"
},
{
"name": "velocidad_viento",
"type": "float",
"value": "5.06835"
},
{
"name": "velocidad_viento_media",
"type": "float",
"value": "4.773367"
},
{
"name": "direccion_viento",
"type": "float",
"value": "109.95"
},
{
"name": "lluvia_24h",
"type": "liter",
"value": "0.78011725"
},
{
"name": "lluvia_24h",
"type": "liter",
"value": "0.78011725"
},
{
"name": "porcentaje_calculado",
"type": "percentage",
"value": "0"
},
{
"name": "porcentaje_real",
"type": "percentage",
"value": "0"}
}]
}
],
"updateAction":"APPEND"
}

### 10.4 Trials and Validation

In order to put to use ORION/CKAN GE’s, we employ an account in the FIWARE LAB environment. The use of Orion was easy and dynamic; however the connectivity to CKAN was complicated due to the problems with the communication protocols / TLS1.2 certificates.

Moreover, the development of the web application to the data access from the client side was not too complicated. It has been developed entirely in PHP and using CURL to the connectivity with CKAN.

The following specific issues and difficulties were found with software/platform during the project,

<table>
<thead>
<tr>
<th>Issue</th>
<th>FIWARE Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectivity error with the CKAN</td>
<td>CKAN</td>
</tr>
<tr>
<td>=&gt;Error: The client and server cannot communicate, because they don’t possess a common algorithm. =&gt;Cause: The “Microsoft Server 2008 Standard” hasn’t got available the TLS 1.1 and TLS 1.2 protocols. =&gt;Solution: To change the security protocols of the CKAN website.</td>
<td>CKAN</td>
</tr>
<tr>
<td>Connectivity error with the CKAN</td>
<td>CKAN</td>
</tr>
<tr>
<td>=&gt;Error: Could not create SSL/TLS secure channel. =&gt;Solution: To specify TLS 1.2 as security protocol.</td>
<td>CKAN</td>
</tr>
</tbody>
</table>
In the use examples, it would be fine the incorporation of the Microsoft .NET programming language.

<table>
<thead>
<tr>
<th>The management of the “X-AUTH-TOKEN” extraction is a little tedious. It’s necessary to ask this password to a url different to the ORION global entity. In addition, it’s necessary to administrate the validation date of the “X-AUTH-TOKEN”, so the application will be more complicated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIWARE LAB</td>
</tr>
</tbody>
</table>

Impossibility of using Windows virtual machines in the instances section.
10.5 Screenshots

Figure 1. Data Sets options
### Context Resources View

**Recursos GPS**

<table>
<thead>
<tr>
<th>Titulo</th>
<th>Categoría</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventos</td>
<td>eventos</td>
</tr>
<tr>
<td>Farmacias</td>
<td>farmacias</td>
</tr>
<tr>
<td>diario</td>
<td>diary</td>
</tr>
<tr>
<td>area_juego</td>
<td>datos_gis</td>
</tr>
<tr>
<td>fuente_bebedora</td>
<td>datos_gis</td>
</tr>
<tr>
<td>juego</td>
<td>datos_gis</td>
</tr>
<tr>
<td>juego_deportivo</td>
<td>datos_gis</td>
</tr>
<tr>
<td>zona_verde</td>
<td>datos_gis</td>
</tr>
<tr>
<td>Logroño Fuel Station</td>
<td>fuel-stations</td>
</tr>
</tbody>
</table>

¡Recursoc **fuente_bebedora** seleccionado!
10.6 Conclusions

In this project we have set the bases to build a solid an integrated collaborative environment between the major, citizens and developers at Logroño. Several enterprises are beginning developments based on this new torrent of raw data that Majors is providing. For JMP Ingenieros, this project has allowed us to start the migration of several services to this new and exciting platform and begin a migration to new services and market.
11 Logroño (Spain): Open Data & SmartAppCity

11.1 Scope & Requirements

The proof of concept of the project consists on using the FIWARE platform to store the current Open Data of the city of Logroño (in Spain) and serve this data to the citizens and visitor through a mobile application.

The project verifies that FIWARE is a solid tool to store data on cloud. All the institutional data of the city (transport information, alerts, publications, events, news, suggestions, gas stations, points of interest, and traffic cameras) will be available directly for Logroño’s citizens and visitors thanks to FIWARE.

The project is divided into three sections:

- Automatic Rollover of open data the City of Logroño to FIWARE.
- Adaptation of the current City Hall App to connect it to FIWARE (and read those "open data" already available).
- As last part of the project, Logroño Data Sets publication on the https://data.lab.fiware.org/es/organization/logrono web portal.

The FIWARE app created will be easily transferred to other FIWARE cities and see that the model is replicable or transferable to other municipalities.

11.2 GEs used and Architecture

The following section shows how the App Logroño.es it is implemented on top of the FIWARE architecture.

We have created a database into a virtual machine in FIWARE. The app reads directly from this FIWARE database. This database is fed from different institutional data sources or webservices. Synchronization is made by a cron and we receive data from different sources and with different natures (dynamic data, static data, etc.).

One of the GEs used on this project is “Publish/Subscribe Context Broker - Orion Context Broker”. Orion Context Broker is an implementation of NGSI9 and NGIS10 with persistence storage based in MongoDB.

During this project, we have automatized the upload and update of the entities PoI (“Points of Interest”) type in the virtual machine.

Example of the structure of an entity with Pol type:
<contextElement>
    <entityId type="Poi" isPattern="false">
        <id>Riojaforum</id>
    </entityId>
    <contextAttributeList>
        <contextAttribute>
            <name>id</name>
            <type>String</type>
            <contextValue>48</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>place_type_id</name>
            <type>String</type>
            <contextValue>14</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>address</name>
            <type>String</type>
            <contextValue>C/ San Millán 25</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>description</name>
            <type>String</type>
            <contextValue>Para compartir su esencia. Riojaforum evoca, ya desde su nombre (…)</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>image</name>
            <type>String</type>
            <contextValue>http://comercios.get-app.es/app/moviles/contenidos/imagenes/lugares/Riojaforum.png</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>phone</name>
            <type>String</type>
            <contextValue>941 276 200</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>email</name>
            <type>String</type>
            <contextValue>riojaforum@riojaforum.com</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>web</name>
            <type>String</type>
            <contextValue>http://www.riojaforum.com</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>latitude</name>
            <type>String</type>
            <contextValue>42.471931</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>longitude</name>
            <type>String</type>
            <contextValue>-2.430984</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>tourism</name>
            <type>String</type>
        </contextAttribute>
    </contextAttributeList>
</contextElement>
- Architecture

The following tables implemented in FIWARE show the Data Model used for this project.
### Future Internet Core Platform

#### D.10.7 Smart Cities connection to FIWARE Lab

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Suggestions</strong></td>
<td>id VARCHAR(255) &lt;br&gt; date DATETIME &lt;br&gt; suggestion_type_id INT(11) &lt;br&gt; subject VARCHAR(255) &lt;br&gt; suggestion_address VARCHAR(255) &lt;br&gt; suggestion_city VARCHAR(100) &lt;br&gt; description VARCHAR(500) &lt;br&gt; response_method VARCHAR(255) &lt;br&gt; name VARCHAR(100) &lt;br&gt; surname VARCHAR(100) &lt;br&gt; nif VARCHAR(15) &lt;br&gt; phone VARCHAR(255) &lt;br&gt; email VARCHAR(100) &lt;br&gt; fax VARCHAR(255) &lt;br&gt; address VARCHAR(255) &lt;br&gt; city VARCHAR(100) &lt;br&gt; postal_code VARCHAR(15) &lt;br&gt; province VARCHAR(255) &lt;br&gt; file VARCHAR(255) &lt;br&gt; city_hall_id VARCHAR(15) &lt;br&gt; device_id VARCHAR(255) &lt;br&gt; response VARCHAR(500)</td>
</tr>
<tr>
<td><strong>Notifications</strong></td>
<td>id INT(11) &lt;br&gt; notification_type_id VARCHAR(255) &lt;br&gt; element_id VARCHAR(100) &lt;br&gt; sent_date DATETIME &lt;br&gt; dispatched_amount INT(11) &lt;br&gt; message VARCHAR(255) &lt;br&gt; start_date DATETIME &lt;br&gt; finish_date DATETIME &lt;br&gt; url VARCHAR(255)</td>
</tr>
<tr>
<td><strong>Menus</strong></td>
<td>id INT(11) &lt;br&gt; title VARCHAR(255) &lt;br&gt; icon BLOB &lt;br&gt; order INT(11) &lt;br&gt; module VARCHAR(255) &lt;br&gt; active TINYINT(4) &lt;br&gt; shortcut_icon BLOB &lt;br&gt; operation TEXT &lt;br&gt; classic_cover TINYINT(1) &lt;br&gt; accessible_cover TINYINT(1) &lt;br&gt; accessible_icon BLOB</td>
</tr>
<tr>
<td><strong>Fuel Stations</strong></td>
<td>id INT(11) &lt;br&gt; name VARCHAR(150) &lt;br&gt; address VARCHAR(255) &lt;br&gt; schedule VARCHAR(150) &lt;br&gt; latitude DOUBLE &lt;br&gt; longitude DOUBLE</td>
</tr>
<tr>
<td><strong>Events Route Geolocation</strong></td>
<td>id INT(10) &lt;br&gt; event_route_id INT(11) &lt;br&gt; position TINYINT(4) &lt;br&gt; latitude DOUBLE &lt;br&gt; longitude DOUBLE &lt;br&gt; date DATETIME</td>
</tr>
<tr>
<td><strong>Events Places</strong></td>
<td>id INT(11) &lt;br&gt; event_id VARCHAR(255) &lt;br&gt; place_id INT(11)</td>
</tr>
<tr>
<td><strong>Place Place Types</strong></td>
<td>place_id INT(11) &lt;br&gt; place_type_id INT(11)</td>
</tr>
<tr>
<td><strong>Bus Day Types</strong></td>
<td>id INT(11) &lt;br&gt; code VARCHAR(45) &lt;br&gt; description VARCHAR(45)</td>
</tr>
<tr>
<td><strong>Tokens</strong></td>
<td>device VARCHAR(255) &lt;br&gt; token VARCHAR(255) &lt;br&gt; date DATETIME &lt;br&gt; news TINYINT(4) &lt;br&gt; events TINYINT(4) &lt;br&gt; publications TINYINT(4) &lt;br&gt; commerce TINYINT(4) &lt;br&gt; origin VARCHAR(20) &lt;br&gt; version VARCHAR(255)</td>
</tr>
<tr>
<td><strong>Places</strong></td>
<td>id INT(11) &lt;br&gt; place_type_id INT(11) &lt;br&gt; name VARCHAR(255) &lt;br&gt; address VARCHAR(255) &lt;br&gt; description TEXT &lt;br&gt; image VARCHAR(255) &lt;br&gt; phone VARCHAR(255) &lt;br&gt; schedule VARCHAR(255) &lt;br&gt; email VARCHAR(255) &lt;br&gt; web VARCHAR(255) &lt;br&gt; latitude DOUBLE &lt;br&gt; longitude DOUBLE &lt;br&gt; tourism TINYINT(4) &lt;br&gt; active TINYINT(4) &lt;br&gt; audio TINYINT(4) &lt;br&gt; video VARCHAR(255)</td>
</tr>
<tr>
<td><strong>Bus Line Routes</strong></td>
<td>id INT(11) &lt;br&gt; operator_code VARCHAR(45) &lt;br&gt; line_code VARCHAR(45) &lt;br&gt; route_code VARCHAR(45) &lt;br&gt; long_description VARCHAR(45) &lt;br&gt; short_description VARCHAR(45) &lt;br&gt; direction VARCHAR(45)</td>
</tr>
<tr>
<td><strong>Bus Line Route Points</strong></td>
<td>id INT(11) &lt;br&gt; operator_code VARCHAR(45) &lt;br&gt; line_code VARCHAR(45) &lt;br&gt; route_code VARCHAR(45) &lt;br&gt; order INT(11) &lt;br&gt; stop_code VARCHAR(45) &lt;br&gt; stop_distance INT(11) &lt;br&gt; stop_time INT(11)</td>
</tr>
<tr>
<td><strong>Publications</strong></td>
<td>id VARCHAR(255) &lt;br&gt; title VARCHAR(255) &lt;br&gt; creation_date DATE &lt;br&gt; publish_date DATE &lt;br&gt; publish_limit_date DATE &lt;br&gt; url VARCHAR(255) &lt;br&gt; short_url VARCHAR(45) &lt;br&gt; image VARCHAR(255) &lt;br&gt; file VARCHAR(255) &lt;br&gt; notice INT(11) &lt;br&gt; active TINYINT(4) &lt;br&gt; address VARCHAR(255) &lt;br&gt; latitude DOUBLE &lt;br&gt; longitude DOUBLE</td>
</tr>
<tr>
<td><strong>Pharmacies</strong></td>
<td>id INT(11) &lt;br&gt; name VARCHAR(150) &lt;br&gt; address VARCHAR(255) &lt;br&gt; schedule VARCHAR(255) &lt;br&gt; phone VARCHAR(255) &lt;br&gt; web VARCHAR(255) &lt;br&gt; email VARCHAR(150) &lt;br&gt; image VARCHAR(255) &lt;br&gt; latitude DOUBLE &lt;br&gt; longitude DOUBLE</td>
</tr>
<tr>
<td><strong>Menu Sections</strong></td>
<td>menu_section_id INT(11) &lt;br&gt; menu_id INT(11)</td>
</tr>
</tbody>
</table>
### Future Internet Core Platform

**D.10.7 Smart Cities connection to FIWARE Lab**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
</table>
| own_images | id VARCHAR(255)  
| | type VARCHAR(45)  
| | image VARCHAR(45)  |
| news | id VARCHAR(255)  
| | locale_id INT(11)  
| | title VARCHAR(255)  
| | short_description TEXT  
| | long_description TEXT  
| | small_image VARCHAR(255)  
| | image VARCHAR(255)  
| | source VARCHAR(255)  
| | date DATE  
| | url TEXT  
| | short_url VARCHAR(45)  
| | notification INT(11)  
| | address VARCHAR(255)  
| | latitude DOUBLE  
| | longitude DOUBLE  |
| pharmacy_pharmacy_schedules | id INT(11)  
| | pharmacy_id INT(11)  
| | pharmacy_schedule_id INT(11)  
| | start_date DATE  
| | finish_date DATE  |
| comments | id INT(11)  
| | element_id VARCHAR(255)  
| | comment_type_id INT(11)  
| | nick VARCHAR(50)  
| | score INT(11)  
| | date DATE  
| | text TEXT  
| | email VARCHAR(50)  
| | approved TINYINT(4)  
| | active TINYINT(4)  |
| parameters | name VARCHAR(45)  
| | value VARCHAR(255)  |
| event_routes | id INT(11)  
| | event_id VARCHAR(255)  
| | color VARCHAR(45)  
| | description VARCHAR(255)  
| | positions INT(11)  |
| event_categories | id INT(11)  
| | name VARCHAR(50)  
| | icon LONGBLOB  
| | icon_map LONGBLOB  
| | icon_name VARCHAR(50)  |
| fuel_stations_prices | id INT(11)  
| | fuel_station_id INT(11)  
| | fuel_id VARCHAR(20)  
| | price DOUBLE  
| | updated DATE  |
| pharmacy_schedules | id INT(11)  
| | open_time TIME  
| | close_time TIME  |

---

**D.10.7 Smart Cities connection to FIWARE Lab**

100
**Future Internet Core Platform**

```
Future Internet Core Platform

```
11.3 Infrastructure and equipment

N/A (Not applicable): Neither sensors nor devices were installed on the city inside this project.

Nevertheless, thousand sensors or devices are used to obtain some of the data:

- City-Buses information: sensors installed in the public transportations vehicles among the 51 city buses that compose the 13 existing bus lines.
- Traffic Cameras: nowadays, 15 street cameras capture the traffic information and it is offered the general public.
- App users: thanks to the more than 26,000 smartphones of the current users of the Logroño.es application, the City Hall receives daily incidents and malfunctions of the city detected by the citizens. The users use their camera to generate images, videos, or sounds and the GPS to geo-position the incidence. They are sent in real time to the City Hall (stored in FIWARE) to correct or solve them.

11.4 (Open) Data

The App just read from institutional sources of information to feed FIWARE and the App.

These are the Datasets used in the project, integrated in FIWARE, and published in the app thanks to different access protocols and mechanisms:

11.4.1 Institutional News

The City Council approaches to neighbours and visitors all the breaking municipal news in the city to stay informed of what’s happening in the city.

Transparency and approach of institutions to citizens.

Access protocol/mechanisms: a call is made to a PHP that returns the information delivered by the City Hall. It is returned in xml format.

11.4.2 Municipal Events

The City Council approaches all the events in the city to neighbours and visitors.

The list of events can be viewed in the app by type of event or by geolocation. Citizens and tourists are able to add the event to their cellphone calendar.
Access protocol/mechanisms: a call is made to a Web Service and it returns the events organized by the City Hall including all the "Teatro Bretón" events.

11.4.3 Official Publications
Magazines, cultural programs, bulletins, and other publications published by the City Council are available in the App.

Users may download the publication without having to wait for the door-to-door delivery.

All the publications can also be read by another app available in the citizen’s device.

It contributes to the municipal transparency and approaches the institutions to the citizens.

Access protocol/mechanisms: PDF publishing system fully integrated with the portal of the City of Logroño. Here there are two web services: one to announce what to show and another one to download the PDFs.

The thumbnails of the PDFs arrive with publications information (from the City Hall, they send one more field with the base64 miniature that it is saved in FIWARE).

11.4.4 City Buses
The App indicates the nearby bus stops to the users and the time remaining that every line will expend in arrive to that stop.

It also indicates the lines you should take to get to any interest place like the City Hall, Hospitals, University...

Access protocol/mechanisms: it calls directly to two different web services: one that reports the position of all the buses (indicating the last stop by that each operating bus has passed) and other web service that informs the next arrivals in minutes for each of the stops, indicating to which line belong each of these buses.

To collect this information of the buses, a request is made to a PHP which is responsible for making the call to the buses’ web-service and for processing and formatting the response to fit it to what the devices expect.

The City Hall also provides the information of routes and stops for each route. The coordinates have to be calculated, because the service brings them in UTM and the application uses latitude and longitude.
11.4.5  Gas Stations
Thanks to this service, the user is able to find the nearest gas station to his/her position and is able to compare prices. It also indicates the cheapest gas station of the city in real time.

Access protocol/mechanisms: Data are downloaded daily straight from the Spanish Ministry of Industry with the prices of gasoline at the different stations in the country. This data is stored in FIWARE and seeped into the application to show only stations that city (by zip code). Through a PHP a call is done that returns the data in JSON.

11.4.6  Traffic
All the traffic cameras of city are available from every Smartphone. So the citizen or tourist can check the traffic conditions before leaving home or the hotel.

Access protocol/mechanisms: Web service that returns the cameras images (which are refreshed every x minutes).

The web service of the traffic cameras is called directly from the application. The position of all traffic cameras is saved in the FIWARE database to initially display them on the map and be able to be chosen by the citizen.

11.4.7  Points of Interest
All the local attractions and highlight sights appear geo-positioned on a map or shortlisted to be easily visualized through the smartphones.

All the Polis appear with contact details, description, image and driving directions to there from their location, choosing the best route.

Audio-guides and factsheets: from each of the sites that are worth not to be missed, it offers geo-positioned description about what you will see and route directions.

Access protocol/mechanisms: The City Council provides all the information of the points geolocated and categorized, with information of each point. These points of interest are in FIWARE and it is accessed directly from the App. In each point can be integrated different podcast systems (audio tours) and videos. Currently there are no videos or audio guides attached to the points in the application.

11.4.8  Announcements
The City Council uses the App to warn and notify urgent notices and announcements to the citizens.
Access protocol/mechanisms: the announcements are inserted via a private web manager provided by JIG. From the City Hall, they fill the information (category, date from and date to be shown, attachments, etc. with the possibility of geo-positioning warnings if they occur at a particular point in the city) and shows announcements directly in the App.

Web Manager where notices are created: is accessed privately from the manager.

The announcements are stored in FIWARE; and the App through a JSON obtains and displays all active messages discharged.

Possibility that such notices will be also reported and reach all users without opening the application. Notifications are also stored in the FIWARE database. These notices are entered by the Municipality directly or introduced by JIG if the City Hall requests it.

11.4.9 Complaints & Suggestions

Citizens may submit to the City Council, all the complaints and suggestions geo-positioned, attaching video, audio or images taken with their mobiles; anonymously or personally if they want to receive an answer.

With the positioning system, response times to incidents from the City Hall are substantially improved.

Citizen participation. Open Government

Access protocol/mechanisms: The data generated by each citizen or tourist through the application are received and stored in FIWARE. Every time a new suggestion is generated, it calls an own web service with JSON which calls the City-Council PHP (010) that also inserts the incident in the management system of the City Council.

Suggestions sent by each user are also saved in the application itself, in internal databases. The shipping to the City-Council web service is done by SOAP.

11.4.10 Pharmacies

On-Duty Pharmacies is one of the institutional open data available on the app; it is offered by the official board of pharmaceutics.

Thanks to the App and the data stored in FIWARE, users are able to find the closest pharmacies to their position (they can visualize the position of all the pharmacies of the city). Also, these pharmacies appear with the symbol in red or green (depending if they are open or closed in that moment). Clicking on each pharmacy, the app will show (in addition to the name, telephone number and address of the pharmacy) the remaining opening time (for the open ones) or next opening (for the closed ones).
11.4.11 DataSets published (CSV/CKAN)

As commented at the introduction, and as last part of the project, a series of Data Sets has been published on the FIWARE data webpage created for the City Hall of Logroño (the actual institution name is “Ayuntamiento de Logroño”):

https://data.lab.fiware.org/es/organization/logrono

We have selected the most interesting ones to the general public and the nature of the FIWARE open data platform. The Open Data published are the following:

- Agenda
- Events
- Pharmacies
- Gas stations
- News
- Publications

This datasets has been uploaded with CSV and they are ready to be processed with CKAN. These datasets are publics and they are used to feed the official App Logroño.es

11.5 Trials and Validation

11.5.1 App prototype

The app prototype created for this project it is an iOS App for smartphones iPhone and for tablets iPad and iPad Mini.

Thanks to our in-house store, we are able to install the app created in every iPhone or iPad for trials and validations without publishing the prototype app at the Apple Store avoiding it to be available for general public.

The instructions to install the App in your iPhone or iPad are as follow:

1. Open a web browser (Safari, Chrome, Opera...) in your terminal (iPhone or iPad).
2. Type the following address on the navigation bar: http://inhouse.get-app.es/
3. Write down the following user ID: fiware (lower case) and password: fiware2014 (lower case) and push the “login” button.
4. Select the Logroño.es App FIWARE prototype and click on the “download” button.
5. Your device will install the app automatically.
6. Once downloaded, the first time you open it, Apple will remind you that the app it is created by JIG. Click OK and start to use the app.
7. You must allow the app to know your position and location. Thanks to this, you will be able to find all the services and points of interest close to your position. You will be able to send geo-located complains and suggestions to the City Hall.

The app was tested and validated in different devices (iPhone 4, iPhone 4S, iPhone 5, iPhone 5S, iPhone 5C, iPad, iPad Mini, iPad Air...) using different data connections (3G, WiFi, GPRS... and also, without data connection) testing all the services and features including geo-location ones.

Also, the prototype was tested with different Operation Systems as iOS 6, iOS 7 and the new iOS 8.0 (deployed in September 17th 2014).

All the tests and validation confirmed that the App works within the required functionality. The app is also accompanied by this descriptions document.

11.5.2 Open Data published

Here, find attached a set of CKAN queries for each of the datasets published:

- Agenda: https://data.lab.fi-ware.org/api/action/datastore_search?resource_id=8e901c6d-2e30-431c-b7ee-8f654cb78ab5
- Events: https://data.lab.fi-ware.org/api/action/datastore_search?resource_id=9e4a69e8-c068-492a-bc13-abda2317891e
- Pharmacies: https://data.lab.fi-ware.org/api/action/datastore_search?resource_id=1dfc3e9d-8b5d-4edc-ab2c-58fd2c8a9547
- Gas stations: https://data.lab.fi-ware.org/api/action/datastore_search?resource_id=b6242848-5370-4719-8656-9b4d20a2498c
- News: https://data.lab.fi-ware.org/api/action/datastore_search?resource_id=ca0628ec-6db-4557-9073-3623dda6bdf
- Publications: https://data.lab.fi-ware.org/api/action/datastore_search?resource_id=5f464fd6-ef8-44de-976-8f4b5312f6fb

11.5.3 Orion Context Broker

We have done several trials and validations for queries and updates of entries:

- Query: http://130.206.83.52:1026/ngsi10/contextEntities/RiojaForum
- Returns all the information of the entity with id="RiojaForum"
- Update: It has been modified the value of one of the attributes of an entity (in this case, the attribute “phone”): http://130.206.83.52:1026/ngsi10/updateContext Method: POST
  
  Payload: 
  ```xml
  <updateContextRequest>
  <contextElementList>
    <contextElement>
      <entityId type="Poi" isPattern="false">
        id="RiojaForum"
      </entityId>
      <contextElement>
        phone="+34 605 123 456"
      </contextElement>
    </contextElement>
  </contextElementList>
  </updateContextRequest>
  ```
Future Internet Core Platform

11.5.4 App Screen shots:

- Images of app screens displaying various features.
- One screen shows the menu with options like "Noticias," "Eventos," and "Publicaciones.
- Another screen displays news categories such as "La VENUS DE LAS PIELES."
D.10.7 Smart Cities connection to FIWARE Lab
11.5.5 Link to an online video with the demo:

- Youtube: [https://www.youtube.com/watch?v=kK_jzCQwivI&feature=youtu.be](https://www.youtube.com/watch?v=kK_jzCQwivI&feature=youtu.be)
12 Malaga (Spain): Citizen as a sensor

12.1 Scope & Requirements

The Malaga CitySense project aims to innovate in data generation for Smart City through the citizen cooperation using use of their Smartphone and his sensors. The objective of the integration in FIWARE was to build interesting applications for the citizens, acting as an applet for the ecosystem.

What we needed at first:

- A dedicated Orion Context Broker instance in order to provide important information to FIWARE. This was setup through the Cloud infrastructure in FIWARE.

- Displaying every field of information generated by the smartphone in an innovative, user-friendly visual way. Significant data have to be presented, giving an overview of user’s information. Furthermore, FIWARE have to provide a strong interactivity between users when new added events appear. Different applications (widgets and operator) were developed for FIWARE Application Mashup (Wirecloud) with this purpose.

12.2 GEs used and Architecture

FIWARE Generic Enablers used are:

- Publish/Suscribe Context Broker GE (Orion), taking into account the Data/Context scenario we are considering. Using the NGSI 10 REST API we were able to communicate with the GE binding XML over HTTP POST requests. This behaviour is just an implementation of the standard.

- Application Mashup GE. Using the global instance of Wirecloud we offer different applications such as a styled map viewer or responsive graphs with historical data.

The general architecture for the application is shown below.
XML binding and HTTP requests are made by a web service after getting smartphones data using a SOAP call. Historical data are stored in an already deployed MySQL database and would be consumed by widgets with Ajax queries. We needed to enable CORS to get this due to problems with Wirecloud proxy.

12.3 Infrastructure and equipment

50 beacons have been installed in the city for interaction with user smartphone, based in BLE technology. These beacons, strategically positioned in emblematic sites, are continuously sending signals which can be interpreted by smartphones, in concrete by the app, retrieving relevant information of points of interest.

On the other hand, each smartphone owned by a citizen represents a sensor node so not really much technology has to be installed to get information.
12.4 Data feeds

All the data is private, at least at the moment. Only information for emblematic sites of the city is released in http://datosabiertos.malaga.eu/, through the Open Government initiative of the City Hall. An own database is deployed to host every user information sent by the installed app in smartphones.

The information below represents the way we are communicating with Orion Context Broker so as inserted data, metadata and entities creation is defined here. Binding XML is used, but JSON could be a solution too, reducing verbosity. This would be developed in future updates.
<updateContextRequest>
  <contextElementList>
    <contextElement>
      <entityId type= dispositivo  isPattern= false >
        <id>  </id>
      </entityId>
      <contextAttributeList>
        <contextAttribute>
          <name>  </name>
          <type>  </type>
          <contextValue>  </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>  </name>
          <type>  </type>
          <contextValue>  </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>  </name>
          <type>  </type>
          <contextValue>  </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>  </name>
          <type>  </type>
          <contextValue>  </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>  </name>
          <type>  </type>
          <contextValue>  </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>  </name>
          <type>  </type>
          <contextValue>  </contextValue>
        </contextAttribute>
      </contextAttributeList>
    </contextElement>
  </contextElementList>
</updateContextRequest>
...<contextAttribute>
  <name> </name>
  <type> </type>
  <contextValue> </contextValue>
</contextAttribute>
<contextAttribute>
  <name> </name>
  <type> </type>
  <contextValue> </contextValue>
</contextAttribute>
<contextAttribute>
  <name> </name>
  <type> </type>
  <contextValue> </contextValue>
</contextAttribute>
<contextAttribute>
  <name> </name>
  <type> </type>
  <contextValue> </contextValue>
</contextAttribute>
<contextAttribute>
  <name> </name>
  <type> </type>
  <contextValue> </contextValue>
</contextAttribute>
<contextAttribute>
  <name> </name>
  <type> </type>
  <contextValue> </contextValue>
</contextAttribute>
<contextAttribute>
  <name> </name>
  <type> </type>
  <contextValue> </contextValue>
</contextAttribute>
<contextAttribute>
  <name> </name>
  <type> </type>
  <contextValue> </contextValue>
</contextAttribute>
<contextAttribute>
  <name> </name>
  <type> </type>
  <contextValue> </contextValue>
</contextAttribute>
**Device**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>temperature</td>
<td>Ambient temperature captured by the smartphone sensor</td>
</tr>
<tr>
<td>light</td>
<td>Ambient luminosity level captured by the smartphone sensor</td>
</tr>
<tr>
<td>humidity</td>
<td>Humidity level captured by the smartphone sensor</td>
</tr>
<tr>
<td>pressure</td>
<td>Pressure level captured by the smartphone sensor</td>
</tr>
<tr>
<td>magnetism</td>
<td>Magnetism level captured by the smartphone sensor</td>
</tr>
<tr>
<td>speed</td>
<td>Movement speed at the moment captured by the smartphone sensor</td>
</tr>
<tr>
<td>proximity</td>
<td>Measures the distance between the smartphone and the user</td>
</tr>
<tr>
<td>rotation</td>
<td>Rotation degrees</td>
</tr>
<tr>
<td>noise</td>
<td>Ambient noise captured by the smartphone sensor</td>
</tr>
<tr>
<td>latitude</td>
<td>Coordinates: latitude</td>
</tr>
<tr>
<td>longitude</td>
<td>Coordinates: longitude</td>
</tr>
<tr>
<td>altitude</td>
<td>Smartphone accelerometer parameter</td>
</tr>
<tr>
<td>acceleration</td>
<td>Smartphone accelerometer parameter</td>
</tr>
<tr>
<td>gravity</td>
<td>Smartphone accelerometer parameter</td>
</tr>
<tr>
<td>gsm_signal</td>
<td>GSM signal: threshold.</td>
</tr>
<tr>
<td>network_type</td>
<td>Network Type the smartphone is connected to.</td>
</tr>
<tr>
<td>operator</td>
<td>Mobile network operator</td>
</tr>
<tr>
<td>battery_level</td>
<td>Battery level of the smartphone</td>
</tr>
<tr>
<td>battery_temperature</td>
<td>Battery temperature of the smartphone</td>
</tr>
<tr>
<td>device_model</td>
<td>Smartphone model</td>
</tr>
<tr>
<td>android_version</td>
<td>Android system version</td>
</tr>
<tr>
<td>date</td>
<td>Update time</td>
</tr>
<tr>
<td>app_version</td>
<td>CitySense App version</td>
</tr>
</tbody>
</table>
Table 1: Main parameters retrieved from smartphone sensors.

This context entity refers to the smartphone and relevant context information collected by sensors.
<updateContextRequest>
  <contextElementList>
    <contextElement>
      <entityId type=mac isPattern=false>
        <id> </id>
      </entityId>
      <contextAttributeList>
        <contextAttribute>
          <name>monument_id</name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>name</name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>address</name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>phone</name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>web</name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>num_visits</name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
      </contextAttributeList>
    </contextElement>
  </contextElementList>
</updateContextRequest>
**Code:** NGSI 10 REST API - APPEND, UPDATE entities ‘mac’.

<table>
<thead>
<tr>
<th>MAC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>monument_id</td>
<td>Identifier of the point of interest in the MySQL database.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the point of interest</td>
</tr>
<tr>
<td>address</td>
<td>Postal address</td>
</tr>
<tr>
<td>web</td>
<td>Website for the point of interest</td>
</tr>
<tr>
<td>num_visits</td>
<td>Number of visits to the POI in the last year</td>
</tr>
<tr>
<td>num_notifications</td>
<td>Number of notifications generated by beacons</td>
</tr>
<tr>
<td>num_query</td>
<td>Number of accesses to the information page of a point of interest (through the App)</td>
</tr>
<tr>
<td>latitude</td>
<td>Coordinates: latitude</td>
</tr>
<tr>
<td>longitude</td>
<td>Coordinates: longitude</td>
</tr>
</tbody>
</table>

**Table:** Information parameters for POIs.

This context entity refers to POIs (Points of Interest), emblematic monuments and sites of the city with beacons interacting with the user through the smartphone app.
<updateContextRequest>
  <contextElementList>
    <contextElement>
      <entityId type='evento' isPattern='false'>
        <id> </id>
      </entityId>
      <contextAttributeList>
        <contextAttribute>
          <name> type </name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name> subcat </name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name> comment </name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name> latitude </name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
        <contextAttribute>
          <name> longitude </name>
          <type></type>
          <contextValue> </contextValue>
        </contextAttribute>
      </contextAttributeList>
    </contextElement>
  </contextElementList>
</updateContextRequest>

*Code*: NGSI 10 REST API - APPEND, UPDATE entities 'evento'.

### Table 3: Information parameters for ‘eventos’.

<table>
<thead>
<tr>
<th>EVENT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Event category, assigned by user in the app</td>
</tr>
<tr>
<td>subtype</td>
<td>Event subcategory, assigned by user in the app</td>
</tr>
<tr>
<td>comment</td>
<td>Comments related to the event, written by the user</td>
</tr>
<tr>
<td>latitude</td>
<td>Coordinates: latitude</td>
</tr>
<tr>
<td>longitude</td>
<td>Coordinates: longitude</td>
</tr>
<tr>
<td>img</td>
<td>Uploaded images related to events.</td>
</tr>
</tbody>
</table>

This context entity refers to city events added by users through the app: traffic jams, leisure events...

#### 12.5 Trials and Validation

Development experience has turned out to be a hard issue due to all the manual operations needed to upload, wire and prove widgets and operators at Mashup Platform. We tried to use Eclipse plugin to automatize the process with no success because of problems with connection between the application and FIWARE Lab. Moreover, a few problems have appeared related to stability and web cache of FIWARE Lab and machines deployed in Cloud environment, showing a state of inconsistency sometimes. Cygnus and Cosmos couldn’t be used at last due to complexity in installation, configuration and not really smooth learning curve. Reaching a solution for historical data tasks was easier using well-known web technologies like PHP/MySQL.

In contrast, documentation about Orion Context Broker is complete and well-defined and the way of exchanging information was easy to implement. Presentations at FIWARE events like “how to develop our first application in FIWARE”, webinars and information at forge made easier to reach our objectives.
12.6 Screenshots

Figure 25 An overview for the Malaga CitySense ecosystem in FIWARE.

Figure 26 Information pop-up shown when POI is selected.
Figure 27 Historical data: last registers and active devices.

Figure 28 Historical data: minimum, maximum and average sensor values.
Figure 29: Historical data: smartphone models, operators and android version.

Figure 30: Historical data: number of visits to POIs.
Future Internet Core Platform

Figure 31: Historical data: sensors and entities inspector.

Figure 32: Heatmap.
12.7 Demo

An application demo can be found in https://www.youtube.com/watch?v=2kkJb7VpwB8. Now is possible to find new information and download links for the smartphone app in http://citysense.es. Stay tuned for more news and updates.
13 Rome (Italy): Integration of mobility open data sets

13.1 Scope & Requirements

The proof of concept developed integrates different kind of mobility opendata in order to show a multilayer platform. Every layer is composed by a single set of open data, such as bus stops, real time traffic information, buses position, etc. The integration of the different layers shows the situation of the mobility in real time of the entire city.

13.2 GEs used and Architecture

We used the POI Data Provider GE which provides spatial search services and data on Points of Interest via RESTful web service API.

We loaded a part of our dataset over http://poi.webhop.org/poi_dp/ server, which resides over the FIWARE LAB cloud, a total of nearly 7000 POI were inserted.

For each POI we stored spatial coordinates and a brief description taken from the opendata dataset.

13.3 Infrastructure and equipment

The web application developed has different layer, on top of everything there is an open street map client plus some core functions developed in javascript to fetch and visualize the data, then there is a middleware level made in php which we used to create, manage and query the different dataset, at the bottom level we used an implementation of the GE.

We integrated two set of open data (bus stops and real time traffic) with the POI data provider generic enabler. The provided data are displayed on a map based on Openstreetmap. The platform developed is a GIS software which shows through different layers mobility information gathered from the chosen dataset. This work is available at: http://anesoftsolutions.com/ifiware/

13.4 (Open) Data

We used an integration of the Open Data mobility information from the Municipality of Rome (precisely from the mobility agency) to the CKAN FIWARE portal. This includes 10 different datasets, reachable at the url:

https://data.lab.fiware.org/dataset?q=rome&sort=score+desc%2C+metadata_modified+d
The main dataset used include a Real Time Traffic web API and a set of all the bus stops, plus bus information, of all Rome.

We use JSON as standard protocol for transferring data and as format for the dataset themselves.

### 13.5 Trials and Validation

This is the opening state of the application, where all the real time traffic lines are displayed and the user can choose the next action to do.

In this screenshot the user selected only one real time traffic line to watch it more in detail. The system allows any number of lines to be selected, in this way users can be more engaged being free to zoom in and out and observe a line in a different level of detail.
In this example user selected the box about public transport stops because the application is developed to mix the different set of opendata we selected, as each piece of information is stored in a different layer.
The next three screenshots show how users can interact with the different level of detail about the bus stops.

At a higher level of zoom the different markers are grouped in cluster which can be interacted by the user. Hovering a cluster shows the boundaries in which the markers are grouped.

Zooming in shows smaller clusters until we are in a level of zoom enough deep to show a single maker. Once a marker is shown the user can click on it to read some data about the stop.
D.10.7 Smart Cities connection to FIWARE Lab
14 Santander (Spain): Integration of the SmartSantander IoT facility in the FIWARE framework

14.1 Scope & Requirements

Integration of the IoT SmartSantander facility on top of the FIWARE platform

- Devices registration
- Observations generation
- Definition of new dictionaries to support sensor measurements as well as citizens observations (mainly social related events).
- Guidelines to access SmartSantander data (both historical and live data) by using Generic Enablers

14.2 GEs used and Architecture

One major strategic aspect of SmartSantander comes with the implementation of a large-scale experimental facility based on a real life IoT infrastructure deployed in the Santander urban landscape. Aligned with the Future Internet Research and Experimentation (FIRE) initiative, the facility offers to the research community the possibility of experimenting over the deployed devices, in several ways:

- **Native experimentation.** Most of deployed IoT Nodes (those with fewer constraints in terms of battery) can be flashed, as many times as required with different experiments, through OTAP (over-the-air programming) or MOTAP (Multihop OTAP), for nodes more than one hop away from the gateway. In this sense, researchers can test its own experiments, such as routing protocols, data mining techniques or network coding schemes.

- **Experimentation at service level.** Data generated by the different services implemented by the project, is also offered to the researchers for developing new services on top of it. Development of new added value services, as well as, correlation between information retrieved by different services, could be examples of this type of experimentation.

Figure 34 presents the overall architecture of SmartSantander, as well as its envisaged integration with the FIWARE platform. It is based on a three-tiered network approach, composed by the following tiers:

- The IoT node tier embraces the majority of the devices deployed in an iot infrastructure, composed of diverse heterogeneous devices, including miscellaneous sensor platforms, tailor-made devices for specific services as well as Radio-Frequency Identification (RFID) and Near Field Communications (NFC) tags. These devices are typically resource-constrained and host a range of sensors and, in some cases, actuators.

- The GW tier links the IoT devices on the edges of the capillary network to the core network infrastructure. IoT nodes are grouped in clusters that depend on a GW device. This node locally
Future Internet Core Platform

gathers and processes the information retrieved by IoT devices within its cluster. It also manages (transmission/reception of commands) them, thus scaling and easing the management of the whole network. The GW tier devices are typically more powerful than IoT nodes in terms of memory and processing capabilities, also providing faster and more robust communication interfaces. GW devices allow virtualisation of IoT devices. This enables the instantiation of emulated sensors or actuators that behave in all respects similar to the actual devices.

- The server tier provides more powerful computing platforms with high availability and directly connected to the core network. The servers are used to host IoT data repositories and application servers. Server tier devices receive data from all GW tier nodes. As a final step, the concept of federation is supported by the architecture. Servers managing networks located in different physical locations can connect among themselves to allow users of the platforms to transparently access IoT nodes that are deployed in different testbeds.

Note: Other devices such as mobile phones and purpose-built devices with reasonable computing power (e.g. mobile devices in vehicles), as well as providing wide area communication capabilities, behave as IoT nodes in terms of sensing capabilities and as GW nodes regarding processing and communication capabilities.

As a result of the integration carried out, data generated by SmartSantander IoT devices is directly sent to IDAS component, which is part of the FIWARE platform, providing external users (either hardware and/or software experimenters or service developers) standardized access to manage the information generated from the SmartSantander facility, which includes both the sensor deployments and the applications used by the citizens.
Every time a new device is registered within the SmartSantander IoT facility (or an existing one updates its capabilities), the Resource Configurator performs a registration/update process within FIWARE, sending a registration message to the IDAS including such capabilities. Once the registration is
completed, new sensor data collected from the SmartSantander facility is translated into new observations and sent to FIWARE. SmartSantander Service Storer is the component that sends the observations messages to FIWARE enablers.

Figure 35 depicts the overall architecture of the FIWARE platform, detailing how it is integrated on top of the SmartSantander IoT facility:

*Figure 35 FIWARE GE integration on top of the SmartSantander platform*
The set of FIWARE enablers used in order to get access to SmartSantander deployments are composed by:

- **Backend Device Management G.E. [IDAS].** IDAS is an implementation of the FIWARE Backend Device Management GE that collects all observations of devices and translates them into NGSI events, available at the corresponding FIWARE Context Broker GE instance. This way, application developers that are not IoT experts can still consume devices’ observations and context data through the ContextBroker enabler. IDAS provides a Device Gateway in the form of a Device Communication API for devices (sensor/actuators/gateways) communication that currently supports SensorML and Lightweight SensorML protocols. This API is used by the SmartSantander Service Storer components to upload observations.

  IDAS includes also its Admin REST API, provided to M2M application developers, that exposes the following functional blocks:
  
  o Provision: Configuration of internal components.
  o Data Model: Operations related to M2M entities (register/modify resources). Top level Object = Service. Part of this API is used by the SmartSantander Resource Configurator to register and update devices.
  o Sensor Data: Operations to retrieve the last/historical measures (observations) of devices.
  o Subscription: Configure apps to receive data published by devices.
  o Command Requests: Configure apps to receive data published by devices and receive their output. Commands are formatted in XML.

  Devices registered in IDAS are both able to send observations and receive commands.

- **Context Broker G.E. [ORION].** Orion Context Broker is an implementation of a context information broker with persistent storage that can play the role of two Generic Enablers (GE) within the FIWARE platform.
  
  o Pub/Sub Context Broker GE.
  o Configuration Management GE (it implements registration forwarding).

  It implements OMA NGSI9/10 specification.

  o NGSI-9 is about context information availability (registerContext, discoverContextAvailability, subscribeContextAvailability...).
  o NGSI-10 is about context information itself (updateContext, queryContext...).

  Using these interfaces, clients can do several operations:

  o Register context producer applications, e.g. a temperature sensor within a room
  o Update context information, e.g. send updates of temperature
  o Being notified when changes on context information take place (e.g. the temperature has changed) or with a given frequency (e.g. get the temperature each minute)
  o Query context information. The Orion Context Broker stores context information updated from applications, so queries are resolved based on that information.
Using this enabler instantiation, the external user can request context and data from SmartSantander registered devices, as well as perform subscriptions to these resources.

- **Big Data Analysis G.E. [COSMOS].** Cosmos is an implementation of the Big Data GE, allowing the deployment of private computing clusters based on Hadoop ecosystem. Current version of Cosmos allows users to:
  - I/O operations regarding Infinity, a persistent storage cluster based on HDFS.
  - Creation, usage and deletion of private computing clusters based on MapReduce and SQL-like querying systems such as Hive or Pig.
  - Manage the platform, in many aspects such as services, users, clusters, etc. from the Cosmos API or the Cosmos CLI.

  It uses:
  - HDFS as its distributed file system.
  - Hadoop core as its MapReduce engine.
  - HiveQL or PIG for querying data.
  - Oozie as remote MapReduce jobs and Hive launcher.
  - Cosmos is linked to Orion data through Cygnus connector, to store and create an archive of historical documentation.

  The available COSMOS API’s and tools provide access to the historical data from SmartSantander deployment.

- **Identity Management G.E. [KeyRock].** Identity Management covers a number of aspects involving users' access to networks, services and applications, including secure and private authentication from users to devices, networks and services, authorization & trust management, user profile management, privacy-preserving disposition of personal data, Single Sign-On (SSO) to service domains and Identity Federation towards applications. Identity Management is used for authorising foreign services to access personal data stored in a secure environment. Hereby usually the owner of the data must give consent to access the data; the consent-giving procedure also implies certain user authentication. IdM offers tools for administrators to support the handling of user life-cycle functions. The IdM offers hosted user profile storage with specific user profile attributes. The KeyRock Identity Management GEI complies with existing standards for user authentication and it provides access information to services acting as a Single Sign-On platform. The KeyRock IdM is a free/open source software which code can be found at Github: KeyRock source code.

  Through the KeyRock IdM, an external user can create his or her own FIWARE account, needed to be identified and authenticated previously to get access to the FIWARE Context broker.

- **Policy Enforcement Point (PEP) Proxy [Wilma].** The PEP Proxy Generic Enabler, together with Identity Management and Authorization PDP GEs, adds authentication and authorization security to the backend applications (currently, Context Broker). Thus, only FIWARE users will be able to access covered GEs or REST services. Wilma is the reference implementation of this Generic Enabler, completely integrated with the FIWARE ecosystem and specifically with FIWARE account. It is thought to work with OAuth2 and XACML protocols, the standards for authentication and authorization chosen in FIWARE. Furthermore, this is the component that
every GEIs are including on top of their REST APIs so it is tested and used in many different scenarios.

14.3 Infrastructure and equipment

SmartSantander IoT facility integrated in FIWARE is based on a real IoT deployment in an urban setting. The core of the facility will be located in the city of Santander and surroundings, encompassing IoT deployments (Figure 36), in different key areas of the city infrastructure, ranging from public transport, key logistics facilities, public places and buildings, work places and residential areas, thus creating the basis for development of a Smart City. This deployment exhibits the diversity, dynamics and scale that are essential for the evaluation of advanced protocol solutions.

![Figure 36. Devices deployment within the SmartSantander IoT facility](image)

**Environmental monitoring devices**

The city of Santander is trying to carry out an effective policy for environmental management through the signing of agreements that aid improvements in air quality and quality of life for its citizens. Key elements in undertaking this task are:

- Monitoring of pollutants
- Noise and temperature measurement
With the aim to develop this environmental monitoring policy, temperature, CO index, noise and luminosity sensors have been installed in street lamps and facades. All these devices send their information, in a multihop fashion (if needed), to the corresponding gateway that gathers all the received information making it available to the both SmartSantander backbone and therefore, sent to the FIWARE GEs. Figure 37 shows several examples of the installation of these devices, at different streetlamps and facades of the city:

![Figure 37. Examples of repeaters installation](image)

More than 1,000 of these fixed nodes have been installed at the Santander city centre. To extend this Environmental Monitoring service based on fixed infrastructure to other areas of the city is costly. Hence, instead of continuing with fixed deployment all over the city, the hardware was installed on municipal public buses, parks and gardens vehicles and taxis. This way we are be able to cover a much wider area on a much more efficient way. Mobile nodes send the collected information to the internet/intranet, and also, interact with the corresponding static nodes placed at streetlamps and facades.
14.4 Parking sensors

Numerous parking sensors have been deployed in the downtown Santander to indicate the parking spots that are available or occupied. In order to detect the occupancy state of a determined parking spot, ferromagnetic sensors buried under the asphalt have been installed, thus sending the corresponding information (free or occupied) associated to a determined parking space. This information is forwarded to the central device (gateway), through the corresponding repeaters (a set of the repeaters installed for environmental monitoring). In Figure 39, the installation process of the parking sensors is shown:

To be more precise, 350 parking sensors have been installed within the “30 Zone”, in order to inform the occupancy degree of determined parking lots. All the information gathered from these sensors is sent to...
the SmartSantander core platform, in which the Service Storer component forwards the information to the FIWARE GEs.

**Traffic Intensity sensors**

Nowadays, the measure and classification of vehicles in road traffic is accomplished by inductive loops placed under the pavement. These inductive loops allow monitoring vehicle passing by means of different configurations which provide us a number of data in order to control several parameters of the traffic (vehicle speed, traffic congestion and traffic accidents, between others).

![Traffic sensors installation in the main roads of the city](image)

However, these systems have several problems and disadvantages like their deployment, maintenance, high cost, and put into gear, among others. In this sense, within the SmartSantander project a solution based on Wireless Sensor Network was deployed, monitoring parameters such as cars speed, occupancy and count in the road lanes in the two main entrances of the city.

**Park and gardens Sensors**

Within the SmartSantander IoT facility, it has installed agricultural IoT devices and weather stations in three major parks of Santander: the Las Llamas Park, La Marga Park and Finca Altamira. A total number of 48 IoT sensor nodes, covering an area of 55000 m², were deployed at key positions inside these three areas, equipped with special agricultural sensors measuring parameters like: air temperature and
humidity, soil temperature and moisture, atmospheric pressure, solar radiation, wind speed/direction, rainfall etc. All these sensors transmit wirelessly the data acquired to a gateway device located within the park and then, this device will manage all the information sending it to the SmartSantander core platform with its inbuilt GPRS/3G module. Once the information is received on it, Service Storer component generates the corresponding observations message to the FIWARE platform.

Figure 41. Soil moisture sensors installation in parks and gardens

Presence and Luminosity Sensors (project OutSmart)

SmartSantander resources has been used as a testbed environment by other European initiatives that have increased its initial capabilities. One of these projects is OutSmart, which has provided SmartSantander with an application to control the street lighting of a located area of the city (Figure 42), deploying also a set of presence detectors and luminosity sensors. This control app keeps the streetlighting off while the captured ambient light level is high (e.g. daylight) and triggers, during the night, a control process based on the pedestrian detection: if presence reported by sensors is “0”, the streetlights of the controlled area are set to “Saving Mode”. When presence detected changes to “1”, the “Normal Mode” for the power supplied is set. This impacts directly over the light levels perceived by the citizens and the power consumption of the controlled power regulator (Figure 43), achieving considerable power savings.
This deployment involves not only capturing data from sensors and information sources, but also, through IDAS enabler and based on the captured data, send commands to the smart power controller in order to manage the supplied power to the streetlights within the OutSmart area. An example of this controlling process is shown in Figure 43.

**Figure 43. Supplied active power and reactive power, based on pedestrian presence detection**

**Participatory sensing service applications**

Participatory Sensing service exploits the use of citizens’ smartphones to make people to become active in contribution and generation data for the SmartSantander Platform. Citizens, Santander City Council
and the local newspaper “El Diario Montañes” are connected into a common platform where they can report, share and be notified of events happening in the city. Users also utilise their mobile phones to send physical sensing information, e.g. GPS coordinates, compass, environmental data such as noise, temperature, etc., feeding this information into the same platform. Both social related events and smartphone measurements are received in the SmartSantander platform and forwarded to the DCA component in FIWARE.

**Augmented Reality and information tags**

The AR service developed in the SmartSantander project includes information about more than 2700 places in the city of Santander, classified in different categories: beaches, parks and gardens, monuments, buildings, tourist information offices, shops, art galleries, libraries, bus stops, taxi ranks, bicycle hire points, parking lots and sports centres, etc. Furthermore, it allows real-time access to traffic and beach cameras, weather reports and forecasts, public bus information and bike hire service, generating a unique ecosystem for end users when moving around the city.

As an illustration of the type of service supported by the SmartSantanderRA application, it offers an interactive experience through its “stroll in the city” mode. With the application in this mode, visitors will receive information about specific Points Of Interest (POIs) taking into account their preferences as they stroll around the city. This, in general, enhances the serendipity effect for the application end user. In this sense, they can define their own preferences (language, places to visit, etc.) and have an interactive context-sensitive experience visiting the city, rather than using traditional standalone applications.

The deployment of stickers including Quick Response (QR) codes and NFC tags in strategic places in the urban landscape, see Figure 44. City guide functionality provided by stickers with NFC technology and QR codes, will provide location-sensitive information (transport service, the cultural agenda, shops, monuments, buildings). These stickers link visitors and citizens to the same information included in the AR Service. Additionally, it complements the SmartSantanderRA app, providing precise information about specific POIs.

*Figure 44. City guide functionality provided by stickers with NFC technology and QR codes*
The AR service has been designed to evaluate its usage, monitoring the devices end user’s behaviour, based on the observation data that are generated every time it is accessed from the application to particular information and also when reading any tag.

14.5 Data

Each one of the aforementioned devices represents a context information source, which is presented as an SmartSantander entity. The available data generated by the SmartSantander devices is the following:

- **1º Entity name “ID”.** A string always starting with "urn:x-iot:smartsantander:u7jcfa". It refers to the sensors located in the city of Santander. You can check them in this [map](#).
  - **Entity type.** Possible values: Santander:device.
  - **Attribute name:**
    - Environmental monitoring: Noise
      - sound
      - batteryCharge
      - acceleration
      - Model
      - Latitude
      - Longitude
    - Environmental monitoring: Luminosity.
      - temperature
      - luminousFlux
      - batteryCharge
      - acceleration
      - Latitude
      - Longitude
    - Environmental monitoring: CO
      - temperature
      - Co_index
      - batteryCharge
      - acceleration
      - Latitude
      - Longitude
    - Environmental monitoring: Temperature
      - temperature
      - batteryCharge
      - acceleration
      - Latitude
Future Internet Core Platform

- Longitude
- Environmental monitoring: Mobile node without CANBUS
  - temperature
  - Humidity
  - CO
  - O3
  - air_particles
  - NO2
  - airQAgr
  - altitude
  - course
  - Odometer
  - Latitude
  - longitude

- Environmental monitoring: Mobile node with CANBUS
  - timestamp
  - total_fuel_used
  - Tank_level_1
  - tank_level_2
  - rpm
  - weight
  - total_distance_traveled
  - distance
  - tacograph_speed
  - engine_temperature
  - instantaneous_consumption
  - axles_number
  - movement_sensor
  - driver_work_status_1
  - driver_work_status_2
  - driver_card_presence_1
  - driver_card_presence_2
  - speed_excess
  - direction_indicator
  - tacograph_behaviour
  - information_handling
  - driver_temporal_status_1
  - driver_temporal_status_2
  - temperature
- Humidity
- CO
- O3
- air_particles
- NO2
- airQAgr

- Parking:
  - presence
  - batteryCharge
  - Latitude
  - Longitude

- Park and gardens: Environmental Station
  - BatteryCharge
  - temperature
  - relativeHumidity
  - solarRadiation
  - atmosphericPressure
  - rainfall
  - windSpeed
  - windDirection
  - latitude
  - longitude

- Park and gardens: Agriculture
  - Battery
  - Temperature
  - relativeHumidity
  - soilMoisture
  - SoilTemperature
  - Latitude
  - longitude

- Park and gardens: Irrigation
  - Battery
  - Temperature
  - relativeHumidity
  - latitude
  - longitude

- Traffic: vehicle counter
  - trafficIntensity. Number of vehicles traversing per minute the measurement point.
- occupancy. The same than trafficIntensity, number of vehicles traversing per minute the measurement point.
- Latitude
- Longitude
- Traffic: vehicle speed
  - trafficIntensity. Number of vehicles traversing per minute the measurement point.
  - occupancy. The same than trafficIntensity, number of vehicles traversing per minute the measurement point.
  - median_speed
  - average_speed
  - Latitud
  - Longitud

2º Entity name “ID”. A string always starting with "OUTSMART". It refers to the sensors located in Parque de las Llamas.
- Entity type. Possible values: Node, AMMS or Regulator.
- Attribute name. It depends on the entity type:
  - Node: Entity ID: "OUTSMART.NODE_ID"
    TimeInstant, Latitud, Longitud, Presence, batteryCharge, Illuminance.
  - AMMS: Entity ID: "OUTSMART.AMMS_ID"
    TimeInstant, Latitud, Longitud, ActivePower, ReactivePower.
  - Regulator: Entity ID: "OUTSMART.RG_LAS_LLAMAS_ID"
    TimeInstant, Latitud, Longitud, ActivePower, ReactivePower, electricPotential, electricCurrent.
14.6 Trials and Validation

Here is described the different messages sent from the SmartSantander core, through the Resource Configurator (to register and update devices) and the Service Storer (for observations) that feeds the different FIWARE enablers as described above.

ENVIRONMENTAL MONITORING: CO NODE

<table>
<thead>
<tr>
<th>Registration message example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rs&gt;</td>
</tr>
<tr>
<td>&lt;id href=&quot;urn:x-iot:smartsantander:u7jcfa:fixed:t35&quot;&gt;M01.urn:x-iot:smartsantander:u7jcfa:fixed:t35&lt;/id&gt; &lt;param arcr=&quot;3:10&quot; href=&quot;8:27&quot; id=&quot;pgps&quot; name=&quot;Location&quot; role=&quot;3:15&quot;&gt; &lt;gps lat=&quot;43.46234&quot; lon=&quot;-3.79746&quot;&gt;&lt;/param&gt;</td>
</tr>
<tr>
<td>&lt;what href=&quot;urn:x-ogc:def:phenomenon:IDAS:1.0:COConcentration&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;what href=&quot;urn:x-ogc:def:phenomenon:IDAS:1.0:temperature&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;what href=&quot;urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;data&gt;&lt;quan uom=&quot;urn:x-ogc:def: uom:IDAS:1.0: dimensionless&quot;&gt;0&lt;/quan&gt;&lt;/data&gt;</td>
</tr>
<tr>
<td>&lt;data&gt;&lt;quan uom=&quot;urn:x-ogc:def: uom:IDAS:1.0: celsius&quot;&gt;0&lt;/quan&gt;&lt;/data&gt;</td>
</tr>
<tr>
<td>&lt;data&gt;&lt;quan uom=&quot;urn:x-ogc:def: uom:IDAS:1.0: percent&quot;&gt;0&lt;/quan&gt;&lt;/data&gt;</td>
</tr>
<tr>
<td>&lt;/rs&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Observation message example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;io&gt;</td>
</tr>
<tr>
<td>&lt;obs from=&quot;urn:x-iot:smartsantander:u7jcfa:fixed:t35&quot;&gt;</td>
</tr>
<tr>
<td>&lt;stm&gt;2013-08-29T02:59:32Z&lt;/stm&gt;</td>
</tr>
<tr>
<td>&lt;what href=&quot;urn:x-ogc:def:phenomenon:IDAS:1.0:temperature&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;param href=&quot;phenomenon:location&quot;&gt;</td>
</tr>
<tr>
<td>&lt;gps lat=&quot;43.46234&quot; lon=&quot;-3.79746&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;/param&gt;</td>
</tr>
<tr>
<td>&lt;param href=&quot;urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge&quot;&gt;</td>
</tr>
<tr>
<td>&lt;quan uom=&quot;urn:x-ogc:def: uom:IDAS:1.0: percent&quot;&gt;79.00&lt;/quan&gt;</td>
</tr>
<tr>
<td>&lt;/param&gt;</td>
</tr>
<tr>
<td>&lt;param href=&quot;urn:x-ogc:def:phenomenon:SmartS:1.0:coIndex&quot;&gt;</td>
</tr>
</tbody>
</table>
ENVIRONMENTAL MONITORING: TEMPERATURE NODE

Registration message example

```xml
<rs><id href="urn:x:iot:smartsantander:u7jcfa:fixed:t93"/><id href="urn:x:iot:smartsantander:u7jcfa:fixed:t93"/>
  <param arcr="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="43.46074" lon="-3.80812"/></param>
  <what href="urn:x:ogc:def:phenomenon:IDAS:1.0:temperature"/>
  <what href="urn:x:ogc:def:phenomenon:IDAS:1.0:batteryCharge"/>
  <data><quan uom="urn:x:ogc:def:uom:IDAS:1.0:celsius">0</quan></data>
  <data><quan uom="urn:x:ogc:def:uom:IDAS:1.0:percent">0</quan></data>
</rs>
```

Observation message example

```xml
<io>
  <obs from="urn:x:iot:smartsantander:u7jcfa:fixed:t68">
    <stm>2014-03-26T03:10:45Z</stm>
    <what href="urn:x:ogc:def:phenomenon:IDAS:1.0:temperature"/>
    <param href="phenomenon:location">
      <gps lat="43.46167" lon="-3.80591"/>
    </param>
    <param href="urn:x:ogc:def:phenomenon:IDAS:1.0:batteryCharge">
      <quan uom="urn:x:ogc:def:uom:IDAS:1.0:percent">82.00</quan>
    </param>
  </obs>
</io>
```
ENVIRONMENTAL MONITORING: NOISE NODE

Registration message example

```xml
<rs><id href="urn:x-iot:smartsantander:u7jcfa:fixed:t522"/><id href="M15.urn:x-iot:smartsantander:u7jcfa:fixed:t522"/>
<param arcr="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="43.46486" lon="-3.80939"/></param>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:sound"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge"/>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:decibel"/></data>
</rs>
```

Observation message example

```xml
<io>
  <obs from="urn:x-iot:smartsantander:u7jcfa:fixed:t377">
    <stm>2014-08-26T13:28:29Z</stm>
    <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:sound"/>
    <param href="phenomenon:location">
      <gps lat="43.46370" lon="-3.80979"/>
    </param>
  </obs>
</io>
```
ENVIRONMENTAL MONITORING: LUMINOSITY NODE

Registration message example

<rs><id href="urn:x-iot:smartsantander:u7jcfa:fixed:t582"/></rs>

<param arcr="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="43.45223" lon="-3.82481"/></param>

<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:luminousFlux"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:temperature"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge"/>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:lumen">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:celsius">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">0</quan></data>
</rs>

Observation message example

<io>

<obs from="urn:x-iot:smartsantander:u7jcfa:fixed:t582">
<stm>2014-10-20T11:56:01Z</stm>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:temperature"/>
<param href="phenomenon:location">
<gps lat="43.45223" lon="-3.82481"/>
</param>
</obs>
</io>
MOBILE ENVIRONMENTAL MONITORING: MOBILE NODE WITH CAN BUS

Registration message example

$rs:<id href="1:1">urn:x-iot:smartsantander:u7jcfa:mobile:fa3004</id><id href="1:8">TUSCAN.urn:x-iot:smartsantander:u7jcfa:mobile:fa3004</id>
<param arcr="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="" lon=""/></param>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:timeStamp"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:totalFuelUsed"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:totalDistanceTravelled"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:totalDistanceTravelled"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:totalDistanceTravelled"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:totalDistanceTravelled"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:distance"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:speed"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:engineTemperature"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:instConsumption"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:axlesNumber"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:movementSensor"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverWorkStatus1"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverWorkStatus2"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverCardPresence1"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverCardPresence2"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:speedExcess"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:direction"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:tacographBehaviour"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:informationHandling"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverTempStatus1"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverTempStatus2"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:temperature"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:relativeHumidity"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:COConcentration"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:O3Concentration"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:airParticles"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:NO2Concentration"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:airqAgregated"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:altitude"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:course"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:odometer"/>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:second">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:liter">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:liter">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:liter">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometersPerHour">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilogram">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometer">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometer">0</quan></data>
Observation message example: CANBUS data

```xml
<obs from="urn:x-iot:smartsantander:u7jcfa:mobile:fa3005">
  <stm>2014-11-04T13:11:54Z</stm>
</obs>
```
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:timeStamp"/>

<param href="phenomenon:location">
  <gps lat="43.4572000" lon="-3.8141500"/>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:totalFuelUsed">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:liter">329805.0</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:tankLevel1">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:liter">81.2</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:tankLevel2">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:liter"/>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:engineRPM">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:revolutionsPerMinute">1118.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:weight">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilogram">999.0</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:totalDistanceTravelled">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometer"/>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:distance">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometer">999999</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:speed">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometersPerHour">46</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:engineTemperature">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:celsius">85</quan>
</param>
<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:instConsumption">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless">-999.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:axlesNumber">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless">-999</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:movementSensor">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless">3</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverWorkStatus1">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless"/>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverWorkStatus2">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless"/>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverCardPresence1">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless"/>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:driverCardPresence2">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless"/>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:speedExcess">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless">3</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:direction">
  <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless">3</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:tacographBehaviour"/>
<quan uom="urn:o-ogc:def:uom:IDAS:1.0:dimensionless">3</quan>
</param>
<param href="urn:o-ogc:def:phenomenon:SmartS:1.0:informationHandling">
<quan uom="urn:o-ogc:def:uom:IDAS:1.0:dimensionless">3</quan>
</param>
</io>

**Observation message example: environmental data**

```xml
<io>
  <obs from="urn:x-iot:smartsantander:u7jcf:a:mobile:fa305">
    <stm>2014-11-04T12:17:48Z</stm>
    <what href="urn:o-ogc:def:phenomenon:IDAS:1.0:COConcentration"/>
    <param href="phenomenon:location">
      <gps lat="43.457200" lon="-3.814150"/>
    </param>
    <param href="urn:o-ogc:def:phenomenon:IDAS:1.0:temperature">
      <quan uom="urn:o-ogc:def:uom:IDAS:1.0:celsius">12.00</quan>
    </param>
    <param href="urn:o-ogc:def:phenomenon:IDAS:1.0:relativeHumidity">
      <quan uom="urn:o-ogc:def:uom:IDAS:1.0:percent">75.00</quan>
    </param>
  </obs>
</io>
```
<param href="urn:x-ogc:def:phenomenon:IDAS:1.0:O3Concentration">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:microgramPerCubicMeter">75.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:airParticles">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:microgramPerCubicMeter">115</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:IDAS:1.0:NO2Concentration">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:microgramPerCubicMeter">115.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:IDAS:1.0:altitude">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:meter">15.0</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:course">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless">250</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:odometer">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometer">968</quan>
</param>

<data>
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:milligramPerCubicMeter">0.10</quan>
</data>

</obs>

MOBILE ENVIRONMENTAL MONITORING: MOBILE NODE WITHOUT CAN BUS

Registration message example

<rs><id href="1:1">urn:x-iot:smartsantander:u7jcf:mobile:fa3026</id><id href="1:8">TUS.urn:x-iot:smartsantander:u7jcf:mobile:fa3026</id>
<param arc="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="" lon=""/></param>

<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:temperature"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:relativeHumidity"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:COConcentration"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:O3Concentration"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:airParticles"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:NO2Concentration"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:airqAgregated"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:altitude"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:course"/>
<what href="urn:x-ogc:def:phenomenon:SmartS:1.0:odometer"/>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:celsius">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:milligramPerCubicMeter">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:microgramPerCubicMeter">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:microgramPerCubicMeter">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:microgramPerCubicMeter">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:meter">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometer">0</quan></data>
</rs>

Observation message example

<io>
  <obs from="urn:x-iot:smartsantander:u7jcfa:mobile:fa3005">
    <stm>2014-11-04T12:17:48Z</stm>
    <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:COConcentration"/>
    <param href="phenomenon:location">
      <gps lat="43.457200" lon="-3.814150"/></param>
  </obs>
</io>
<param href="urn:x-ogc:def:phenomenon:IDAS:1.0:temperature">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:celsius">12.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:IDAS:1.0:relativeHumidity">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">75.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:IDAS:1.0:O3Concentration">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:microgramPerCubicMeter">75.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:airParticles">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:microgramPerCubicMeter">115.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:IDAS:1.0:NO2Concentration">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:microgramPerCubicMeter">115.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:IDAS:1.0:altitude">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:meter">15.00</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:course">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:dimensionless">250</quan>
</param>

<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:odometer">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometer">968</quan>
</param>

<data>
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:milligramPerCubicMeter">0.10</quan>
</data>

</obs>
PARK AND GARDENS: IRRIGATION NODE

**Registration message example**

```xml
<rs><id href="1:1">urn:x-iot:smartsantander:u7jcfa:fixed:t3209</id><id href="1:8">M21.urn:x-iot:smartsantander:u7jcfa:fixed:t3209</id>
  <param arcr="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="43.47486" lon="3.80163"/></param>
  <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:temperature"/>
  <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:relativeHumidity"/>
  <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge"/>
  <data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:celsius">0</quan></data>
  <data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">0</quan></data>
  <data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">0</quan></data>
</rs>

**Observation message example**

```xml
<io>
  <obs from="urn:x-iot:smartsantander:u7jcfa:fixed:t3218">
    <stm>2014-10-30T14:40:09Z</stm>
    <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:relativeHumidity"/>
    <param href="phenomenon:location">
      <gps lat="43.47461" lon="-3.80084"/>
    </param>
    <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:temperature">
      <quan uom="urn:x-ogc:def:uom:IDAS:1.0:celsius">26.93</quan>
    </param>
    <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:batteryCharge">
      <quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">78.00</quan>
    </param>
  </obs>
</io>
```
Future Internet Core Platform

D.10.7 Smart Cities connection to FIWARE Lab

PARK AND GARDENS: AGRICULTURE NODE OBSERVATION

Registration message example

```xml
<rs><id href="urn:x-iot:smartsantander:u7jcfafixed:t3202" id="urn:x-iot:smartsantander:u7jcfafixed:t3202"

<param arcr="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="43.47422" lon="3.79769"></param>

<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:temperature"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:relativeHumidity"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:SoilMoistureTension"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:soilTemperature"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge"/>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:celsius">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:centibar">0</quan></data>
</rs>
```

Observation message example

```xml
<io>
</io>
```
<obs from="urn:x-iot:smartsantander:u7jcfa:fixed:t3302">
  <stm>2014-10-17T11:11:07Z</stm>
  <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:trafficIntensity"/>
  <param href="phenomenon:location">
    <gps lat="43.44986" lon="-3.83014"/>
  </param>
  <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:median_speed">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometersPerHour">-1.00</quan>
  </param>
  <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:average_speed">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometersPerHour">-1.00</quan>
  </param>
  <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:occupancy">
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">0.00</quan>
  </param>
  <data>
    <quan uom="urn:x-ogc:def:uom:IDAS:1.0:vehiclesPerMinute">0</quan>
  </data>
</obs>

PARK AND GARDENS: ENVIRONMENTAL STATION NODE

Registration message example

<rs id="urn:x-iot:smartsantander:u7jcfa:fixed:t3224"><id href="1:8">M21.urn:x-iot:smartsantander:u7jcfa:fixed:t3224</id>
  <param arc="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="43.47556" lon="-3.79906"/></param>
</rs>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:relativeHumidity"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:solarRadiation"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:atmosphericPressure"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:rainfall"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:windSpeed"/>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge"/>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:hectoPascal">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:literPerSquareMeter">0</quan></data>
<data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometersPerHour">0</quan></data>
</rs>

**Observation message example**

```xml
<io>
  <obs from="urn:x-iot:smartsantander:u7jcfa:fixed:t3248">
    <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:relativeHumidity"/>
    <param href="phenomenon:location">
      <gps lat="43.46727" lon="-3.81165"/>
    </param>

    <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:temperature">
      <quan uom="urn:x-ogc:def:uom:IDAS:1.0:celsius">12.20</quan>
    </param>

    <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:solarRadiation">
      <quan uom="urn:x-ogc:def:uom:IDAS:1.0:wattsPerSquareMeter">7.32</quan>
    </param>

    <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:atmosphericPressure">
      <quan uom="urn:x-ogc:def:uom:IDAS:1.0:hectoPascal">0</quan>
    </param>
  </obs>
</io>
```
<quand uom="urn:x-ogc:def:uom:IDAS:1.0:hectoPascal">991.07</quand>
</param>
<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:rainfall">
<quand uom="urn:x-ogc:def:uom:IDAS:1.0:literPerSquareMeter">0.00</quand>
</param>
<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:windSpeed">
<quand uom="urn:x-ogc:def:uom:IDAS:1.0:kilometersPerHour">10.00</quand>
</param>
<param href="urn:x-ogc:def:phenomenon:SmartS:1.0:batteryCharge">
<quand uom="urn:x-ogc:def:uom:IDAS:1.0:percent">88.00</quand>
</param>
</data>
</io>

PARKING: PRESENCE NODES

Registration message example

<rs><id href="1:1">urn:x-iot:smartsantander:u7jcfax:fixed:np3609</id><id href="1:8">PLOT7.urn:x-iot:smartsantander:u7jcfax:fixed:np3609</id>

<param arcr="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="43.463096067" lon="-3.798491433" /></param>
<what href="urn:x-ogc:def:phenomenon:IDAS:1.0:presence"/>
<data><bln>false</bln></data>
</rs>

Observation message example

<io><obs from="urn:x-iot:smartsantander:u7jcfax:fixed:np3600"/>
TRAFFIC: VEHICLE COUNTER NODE

Registration message example

```xml
<rs><id href="urn:x-iot:smartsantander:u7jcfa:fixed:t3306"/><id href="urn:x-iot:smartsantander:u7jcfa:fixed:t3306"/>
  <param arcr="3:10" href="8:27" id="gps" name="Location" role="3:15"><gps lat="43.45135" lon="-3.82782"/></param>
  <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:trafficIntensity"/>
  <what href="urn:x-ogc:def:phenomenon:SmartS:1.0:occupancy"/>
  <data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:vehiclesPerMinute"/></data>
  <data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent"/></data>
</rs>
```

Observation message example

```xml
<io>
  <obs from="urn:x-iot:smartsantander:u7jcfa:fixed:t3307">
    <stm>2014-10-17T11:11:07Z</stm>
    <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:trafficIntensity"/>
    <param href="phenomenon:location">
      <gps lat="43.45138" lon="-3.82784"/>
    </param>
    <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:occupancy"/>
  </obs>
</io>
```
TRAFFIC: VEHICLE SPEED NODE

Registration message example

```xml
<rs>
  <id href="urn:x-iot:smartsantander:u7jcfa:fixed:t3305" id="urn:x-iot:smartsantander:u7jcfa:fixed:t3305"/>
  <param arc="3:10" href="8:27" id="pgps" name="Location" role="3:15"><gps lat="43.44991" lon="-3.83027"/></param>
  <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:trafficIntensity"/>
  <what href="urn:x-ogc:def:phenomenon:SmartS:1.0:occupancy"/>
  <what href="urn:x-ogc:def:phenomenon:SmartS:1.0:median_speed"/>
  <what href="urn:x-ogc:def:phenomenon:SmartS:1.0:average_speed"/>
  <data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent"/>0.00</quan>
  </data>
  <data><quan uom="urn:x-ogc:def:uom:IDAS:1.0:vehiclesPerMinute"/>0</quan>
  </data>
</rs>
```
Observation message example

```xml
<io>
  <obs from="urn:x-iot:smartsantander:u7jcfa:fixed:t3311">
    <stm>2014-10-17T11:11:07Z</stm>
    <what href="urn:x-ogc:def:phenomenon:IDAS:1.0:trafficIntensity"/>
    <param href="phenomenon:location">
      <gps lat="43.45214" lon="-3.82540"/>
    </param>
    <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:median_speed">
      <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometersPerHour">1.00</quan>
    </param>
    <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:average_speed">
      <quan uom="urn:x-ogc:def:uom:IDAS:1.0:kilometersPerHour">1.00</quan>
    </param>
    <param href="urn:x-ogc:def:phenomenon:SmartS:1.0:occupancy">
      <quan uom="urn:x-ogc:def:uom:IDAS:1.0:percent">0.00</quan>
    </param>
    <data>
      <quan uom="urn:x-ogc:def:uom:IDAS:1.0:vehiclesPerMinute">0</quan>
    </data>
  </obs>
</io>
```

OUTSMART Deployment: PRESENCE DETECTOR & LUMINOSITY NODE OBSERVATION

Registration message example

```xml
<rs>
  <id href="identifier:localIdentifier">NODE_3500</id>
</rs>
```
<id href="identifier:serialNumber">00:13:A2:00:40:8C:A8:89</id>

<id href="identifier:localName">LLAMAS_00</id>

<id href="identifier:manufacturer">UC</id>

<id href="identifier:UniversalIdentifierOfLogicalHub">OUTSMART</id>

<what href="phenomenon:presence"/>

<what href="phenomenon:batteryCharge"/>

<what href="phenomenon:illuminance"/>

<param href="phenomenon:location" id="gps" arcr="property:read" name="GPS" role="property:configurationProperty">
    <gps lon="-3.80517" lat="43.4722"/>
</param>

<data id="PD" name="Presence Detection">
    <quan uom="uom:dimensionless">3</quan>
</data>

<data id="BT" name="Battery Status">
    <quan uom="uom:percent">4</quan>
</data>

<data id="LT" name="Illuminance">
    <quan uom="uom:lux">5</quan>
</data>
</rs>

**Presence Detection message example**

<io>

    <obs from="NODE_3512">
        <stm>2013-02-19T09:42:28Z</stm>
        <what href="phenomenon:presence"/>
        <param href="identifier:UniversalIdentifierOfLogicalHub">
            <text>OUTSMART</text>
        </param>
    </obs>

</io>
Battery Status message example

<io>
  <obs from="NODE_3510">
    <stm>2013-04-04T17:10:21Z</stm>
    <what href="phenomenon:batteryCharge"/>
    <param href="identifier:UniversalIdentifierOfLogicalHub">
      <text>OUTSMART</text>
    </param>
    <param>
      <gps lon="-3.8058152198791504" lat="43.472164154052734"/>
    </param>
    <data id="BT" name="Battery Status">
      <quan uom="uom:percent">56</quan>
    </data>
  </obs>
</io>

Illuminance level message example

<io>
  <obs from="NODE_3505">
    <stm>2013-03-22T00:15:20Z</stm>
    <what href="phenomenon:illuminance"/>
    <param href="identifier:UniversalIdentifierOfLogicalHub"/>
OUTSMART Deployment: Advanced Metering and Management System (AMMS)

Registration message example

```xml
<rs>
  <id href="identifier:localIdentifier">AMMS_06E1E5B2100394784</id>
  <id href="identifier:serialNumber">ES0027700035628001TB0F</id>
  <id href="identifier:localName">Circuito_Mejico</id>
  <id href="identifier:manufacturer">EON_SP</id>
  <id href="identifier:UniversalIdentifierOfLogicalHub">OUTSMART</id>
  <what href="urn:x-ogc:def:phenomenon:Outsmart:1.0:ActivePower"/>
  <what href="urn:x-ogc:def:phenomenon:Outsmart:1.0:ReactivePower"/>
  <param href="phenomenon:location" id="gps" arc="property:read" name="GPS"
    role="property:configurationProperty">
    <gps lon="-3.82554" lat="43.4575"/>
  </param>
  <data id="AI" name="ActivePower">
    <quan uom="uom:wattPerHour">1</quan>
  </data>
  <data id="RI" name="ReactivePower">
    <text uom="urn:x-ogc:def:uom:Outsmart:1.0:VArPerHour">2</text>
  </data>
</rs>
```
Observation message example

<io>

<obs from="AMMS_06E1E5B2100394784">
  <stm>2012-10-15T01:30:00+02.00</stm>
  <what href="phenomenon:Active Power"/>
  <param href="identifier:UniversalIdentifierOfLogicalHub">
    <text>OUTSMART</text>
  </param>
  <data>
    <quan uom="uom:Watt per hour">2288</quan>
  </data>
</obs>

<obs from="AMMS_06E1E5B2100394784">
  <stm>2012-10-15T01:30:00+02.00</stm>
  <what href="phenomenon:Reactive Power"/>
  <param href="identifier:UniversalIdentifierOfLogicalHub">
    <text>OUTSMART</text>
  </param>
  <data>
    <quan uom="uom:Var/h">0</quan>
  </data>
</obs>
</io>

OUTSMART Deployment: POWER CONTROLLER Observation

Registration message example

<rs>
### Observation message example

```xml
<id href="identifier:localIdentifier">RG_LAS_LLAMAS_01</id>

<id href="identifier:serialNumber">628339895</id>

<id href="identifier:localName">Reg_Las Llamas_01</id>

<id href="identifier:manufacturer">ARELSA</id>

<id href="identifier:UniversalIdentifierOfLogicalHub">OUTSMART</id>

<what href="urn:x-ogc:def:phenomenon:Outsmart:1.0:ActivePower"/>

<what href="urn:x-ogc:def:phenomenon:Outsmart:1.0:ReactivePower"/>

<what href="phenomenon:electricPotential"/>

<what href="phenomenon:electricCurrent"/>

<param href="phenomenon:location" id="gps" arc="property:read" name="GPS" role="property:configurationProperty">
  <gps lon="-3.80692" lat="43.4717"/>
</param>

<param name="Command URL" role="property:operationProperty" arc="property:read" href="property:commandURL">
  <text>http://193.144.201.55:1122/OutSmart_SC/Reg_Rest_API/COMMAND</text>
</param>

<data id="AI" name="ActivePower">
  <quan uom="uom:wattPerHour">1</quan>
</data>

<data id="RI" name="ReactivePower">
  <text uom="urn:x-ogc:def:uom:Outsmart:1.0:VArPerHour">2</text>
</data>

<data id="V" name="Voltage">
  <quan uom="uom:volt">6</quan>
</data>

<data id="A" name="Intensity">
  <text uom="urn:x-ogc:def:uom:Outsmart:1.0:milliampere">7</text>
</data>
```
D.10.7 Smart Cities connection to FIWARE Lab

<io>
  <obs from="RG_LAS_LLAMAS_01">
    <stm>2013-05-13T09:33:34Z</stm>
    <what href="urn:x-ogc:def:phenomenon:Outsmart:1.0:ActivePower"/>
    <param href="identifier:UniversalIdentifierOfLogicalHub">
      <text>OUTSMART</text>
    </param>
    <param>
      <gps lon="-3.806917667388916" lat="43.471702575683594"/>
    </param>
    <data id="AI" name="ActivePower">
      <quan uom="urn:x-ogc:def:uom:Outsmart:1.0:wattPerHour">0</quan>
    </data>
  </obs>

  <obs from="RG_LAS_LLAMAS_01">
    <stm>2013-05-13T09:33:34Z</stm>
    <what href="urn:x-ogc:def:phenomenon:Outsmart:1.0:ReactivePower"/>
    <param href="identifier:UniversalIdentifierOfLogicalHub">
      <text>OUTSMART</text>
    </param>
    <param>
      <gps lon="-3.806917667388916" lat="43.471702575683594"/>
    </param>
    <data id="RI" name="ReactivePower">
      <text uom="urn:x-ogc:def:uom:Outsmart:1.0:VArPerHour">0</text>
    </data>
  </obs>

  <obs from="RG_LAS_LLAMAS_01">
    <stm>2013-05-13T09:33:34Z</stm>
    <what href="phenomenon:electricPotential"/>
  </obs>
</io>
14.7 Tutorials

In order to further detail to SmartSantander external users how to query and retrieve data from the different deployments within SmartSantander facilities and through the FIWARE architecture, UC has edited each tutorials, based on the documentation available through the FIWARE Wiki but focused and customized in the set of data available on SmartSantander. These are:

• **SmartSantander Open Data access using FIWARE Big Data Analysis G.E. [COSMOS]** (document currently in review process)

### 14.8 Dictionaries

Next table shows the dictionaries that have been used when both registering devices and generating observations to the FIWARE platform, including the type of device, measurement, phenomenon and units that are used for such purpose. It is important to note that we are using both the IDAS dictionary and also one that contains specific units and phenomenon defined within SmartSantander (urn:x-ogc:def:phenomenon:SmartS:)
<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>Measurement</th>
<th>Urn_phenomenon</th>
<th>Urn_unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery</td>
<td>batteryCharge</td>
<td>urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:percent</td>
</tr>
<tr>
<td>Battery</td>
<td>batteryCharge</td>
<td>urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:dimensionless</td>
</tr>
<tr>
<td>CO</td>
<td>CO_index</td>
<td>urn:x-ogc:def:phenomenon:SmartS:1.0:coIndex</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:dimensionless</td>
</tr>
<tr>
<td>Battery</td>
<td>batteryCharge</td>
<td>urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:percent</td>
</tr>
<tr>
<td>Battery</td>
<td>batteryCharge</td>
<td>urn:x-ogc:def:phenomenon:IDAS:1.0:batteryCharge</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:percent</td>
</tr>
<tr>
<td>EnvironmentStation</td>
<td>Battery</td>
<td><code>urn:x-ogc:def:phenomenon:IDAS:1.0:BatteryCharge</code></td>
<td><code>urn:x-ogc:def:uom:IDAS:1.0:percent</code></td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
<td>--------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Irrigation</td>
<td>Battery</td>
<td><code>urn:x-ogc:def:phenomenon:IDAS:1.0:BatteryCharge</code></td>
<td><code>urn:x-ogc:def:uom:IDAS:1.0:percent</code></td>
</tr>
</tbody>
</table>

D.10.7 Smart Cities connection to FIWARE Lab

180
<table>
<thead>
<tr>
<th>Phenomenon</th>
<th>Type</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mob_env_can timestamp</td>
<td>urn:x-ogc:def:phenomenon:SmartS:1.0:timeStamp</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:second</td>
</tr>
<tr>
<td>total_fuel_used</td>
<td>urn:x-ogc:def:phenomenon:SmartS:totalFuelUsed</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:liter</td>
</tr>
<tr>
<td>Tank_level_1</td>
<td>urn:x-ogc:def:phenomenon:SmartS:tankLevel1</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:liter</td>
</tr>
<tr>
<td>tank_level_2</td>
<td>urn:x-ogc:def:phenomenon:SmartS:tankLevel2</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:liter</td>
</tr>
<tr>
<td>rpm</td>
<td>urn:x-ogc:def:phenomenon:SmartS:engineRPM</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:revolutionsPerMinute</td>
</tr>
<tr>
<td>Phenomenon</td>
<td>Definition</td>
<td>Unit</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>total_distance_traveled</td>
<td>urn:x-ogc:def:phenomenon:SmartS:totalDistanceTravelled</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:kilometer</td>
</tr>
<tr>
<td>tacograph_speed</td>
<td>urn:x-ogc:def:phenomenon:SmartS:sp speed</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:kilometersPerHour</td>
</tr>
<tr>
<td>axles_number</td>
<td>urn:x-ogc:def:phenomenon:SmartS:axlesNumber</td>
<td>urn:x-ogc:def:uom:SmarS:1.0:axles</td>
</tr>
<tr>
<td>driver_work_status_1</td>
<td>urn:x-ogc:def:phenomenon:SmartS:driverWorkStatus1</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:dimensionless</td>
</tr>
<tr>
<td>driver_work_status_2</td>
<td>urn:x-ogc:def:phenomenon:SmartS:driverWorkStatus2</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:dimensionless</td>
</tr>
<tr>
<td>driver_card_presence_1</td>
<td>urn:x-ogc:def:phenomenon:SmartS:driverCardPresence1</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:dimensionless</td>
</tr>
<tr>
<td>Phenomenon</td>
<td>Identifier</td>
<td>Unit</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>verCardPresence2</td>
<td>verCardPresence2</td>
<td>nless</td>
</tr>
<tr>
<td>information_handling</td>
<td>urn:x-ogc:def:phenomenon:SmartS:informationHandling</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:dimensionless</td>
</tr>
<tr>
<td>driver_temporal_status_1</td>
<td>urn:x-ogc:def:phenomenon:SmartS:driverTempStatus1</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:dimensionless</td>
</tr>
<tr>
<td>driver_temporal_status_2</td>
<td>urn:x-ogc:def:phenomenon:SmartS:driverTempStatus2</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:dimensionless</td>
</tr>
<tr>
<td>Mob_env_no_CAN</td>
<td>temperature</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:celsius</td>
</tr>
<tr>
<td></td>
<td>Humidity</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:percent</td>
</tr>
<tr>
<td></td>
<td>CO</td>
<td>urn:x-ogc:def:uom:SmartS:1.0:milligramPerCubicMeter</td>
</tr>
<tr>
<td></td>
<td>O3</td>
<td>urn:x-ogc:def:uom:SmartS:1.0:microgramPerCubicMeter</td>
</tr>
<tr>
<td></td>
<td>air_particles</td>
<td>urn:x-ogc:def:uom:SmartS:1.0:microgram</td>
</tr>
<tr>
<td>Phenomenon</td>
<td>Identifier</td>
<td>Unit</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>0:airParticles</td>
<td>urn:x-ogc:def:phenomenon:IDAS:1.0:NO2Concentration</td>
<td>urn:x-ogc:def:uom:SmartS:1.0:microgramPerCubicMeter</td>
</tr>
<tr>
<td>NO2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>airQAgr</td>
<td>urn:x-ogc:def:phenomenon:SmartS:airAggregated</td>
<td>urn:x-ogc:def:uom:SmartS:1.0:microgramPerCubicMeter</td>
</tr>
<tr>
<td>course</td>
<td>urn:x-ogc:def:phenomenon:SmartS:1.0:course</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:dimensionless</td>
</tr>
<tr>
<td>Odometer</td>
<td>urn:x-ogc:def:phenomenon:SmartS:1.0:odometer</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:kilometre</td>
</tr>
<tr>
<td>Vehicle_counter</td>
<td>occupancy</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:percent</td>
</tr>
<tr>
<td>Vehicle_speed</td>
<td>occupancy</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:percent</td>
</tr>
<tr>
<td></td>
<td>median_speed</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:median_speed</td>
</tr>
<tr>
<td></td>
<td>average_speed</td>
<td>urn:x-ogc:def:uom:IDAS:1.0:kilometre</td>
</tr>
<tr>
<td></td>
<td>TimeStamp</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>Magnetic_loop</td>
<td>TimeStamp</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Arc</td>
<td></td>
</tr>
</tbody>
</table>
15 Sevilla (Spain): Smart Fountains

15.1 Scope & Requirements

This scenario will monitor water sensors from ornamental fountains from the city of Sevilla (Spain). It uses the FIWARE platform for data acquisition, storage, logic and user interface. A total of 3 fountains are going to be equipped with hardware from Adevice and Gerencia de Urbanismo to gather critical information as pH or chlorine levels. This information is then sent to the FIWARE platform, using the Sevilla node, and a webpage based also on FIWARE shows the current status of the fountain.

Requirements are:

- Use existing equipment (already deployed on some fountains) as sensors
- Deploy at least 3 fountains, being one of them the “Plaza de España”
- Integrate as much sensors as possible
15.2 GEs used and Architecture

The GEs used in this use case are:

Orion Context Broker, Proton Complex Event Processor, Wirecloud, CKAN, COSMOS, IDAS

15.3 Infrastructure and equipment

In order to collect the measurements gathered by installed equipment into the ornamental fountains selected for the FIWARE Pilot, a specific hardware platform has been designed and developed.

This hardware platform is capable of doing the tasks of reading measurements from fountain equipment and performs as a communication gateway to send gathered measurements to FIWARE platform.

Each fountain has basically three different measuring systems:

- Power Meter
- Drive or pump motor controller
- Water quality Meter
Depending on the type of fountain, we can find one type or another of the measurement systems indicated above, but the measured magnitudes, in most cases, will be similar.

The following table summarizes existing sensors systems in fountains:

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
</table>
| • **Manufacturer:** Schneider  
  • **Model:** Altivar 61 | The Altivar 61 drive is a frequency inverter for three-phase asynchronous motors from 0.75 kW to 800 kW.  
Gives information about the behavior of the engine pumping source, among all measurements offered by the drive, it have been selected the following:  
• Output Speed (rpm)  
• Output Torque (unit)  
• Motor Current (A)  
• Motor Voltage (V)  
• Motor Power (%)  
• Energy Consumed by the drive (Wh)  
Reading Interface: **MODBUS RTU** |
| • **Manufacturer:** Schneider  
  • **Model:** Altivar 312 | This system is similar to Altivar 61, but with lower performance.  
Measurements gathered by ATV312 are the following:  
• Output Speed (rpm)  
• Output Torque (unit)  
• Motor Current (A)  
• Motor Voltage (V)  
• Motor Power (%)  
• Energy Consumed by the drive (Wh)  
Reading Interface: **MODBUS RTU** |
The power monitor is a compact, cost-effective, electric power and energy metering device intended for use in industrial control applications, such as distribution centers, industrial control panels, and motor control centers.

It gives measurements about total energy consumption of the fountain, specifically the following magnitudes have been selected:

- Power factor: PF1, PF2, PF3 (%)
- Total Power (kW)
- Current: I1, I2, I3 (I)
- Voltage: V1N, V2N, V3N (V)
- Voltage: V12, V23, V31 (V)

Reading Interface: **MODBUS RTU**

This device measures total power consumption of the fountain.

The parameters read by the datalogger will be the same as in the case of AllenBradley PM1000:

- Power factor: PF1, PF2, PF3 (%)
- Total Power (kW)
- Intensidad de cada línea: I1, I2, I3 (I)
- Voltage: V1N, V2N, V3N (V)
- Voltage: V12, V23, V31 (V)

Reading Interface: **MODBUS RTU**

PCA320 is a Chlorine and PH Analyzer to sanitize irrigation water.

The parameters read from this device are:

- Water Temperature (°C)
- Water acidity (pH)
- Free chlorine in water (%)
All equipment described above have an output interface that allows querying data from an external device that performs data logging tasks.

For this reason, it has developed a communications gateway that provides, on one hand, an output interface adapted for the reading of all the selected magnitudes from devices installed in the fountains, and, on the other hand, it provides a GPRS interface for sending measurements to the FiWARE platform. Internal architecture for communication gateway is:
As a reference, we can then a picture of the installation of this system in the fountain placed in Plaza de España in Seville. In this picture, we can see all the systems described in previous paragraph and its interconnection.

The antenna is located at the site where best GPRS signal is received by the gateway during the installation tests.

In the image below, we can see in more detail inside the communication gateway:
15.4 Data

The data sources are the different devices described in the “infrastructure and equipment” section of this document. All the info that they generate is in real time. After, that info is sent to the FIWARE platform, specifically to the General Enabler (GE) known as context broker (CB).

This CB can be consulted by the different components developed, this components were made using Wire Cloud, a FIWARE tool that enables to create UI and services, with the data available in the Context Broker.

The UI can show alarms; these alarms are consequence of situations detected with the CEP. This tool is another GE available in FIWARE, in which we can define conditions, ranges of time and other context information to create those alarms.

All the developed WC components (widgets and operators) and the CEP's defined situations, are able to work directly with the data stored in the Context broker, via HTTP REST request (subscriptions, queries and notifications), following NGSI10 standard.

Next, some examples of these operations:

**Query to CB via HTTP request:**

URL:

Subscription from CEP to CB using REST request:

URL:

http://<CB url>:1026/NGSI10/subscribeContext

Payload:

```
{
  "entities": [
    {
      "isPattern": "true",
      "id": "Sevilla:FUENTES:x:y:z"
    }
  ],
  "attributes": [
  ],
  "reference": "http://130.206.82.120:1026/ngsi10/notifyContext",
  "duration": "P2M",
  "notifyConditions": [
```
In the image below we are testing also the XML mode from Orion, but finally we have selected JSON for every transaction:

```
Subscription from a WC operator to CB, using WC APIs (JS)

function subscription(){
    var entityIdList = [
        {
            id: IDfuente, isPattern: true
        }
    ];

    var attributeList = atributos_suscripcion;
    var duration = 'P12M';
    var throttling = null;
    var notifyConditions = [{
        type: 'ONCHANGE',
        condValues: atributos_suscripcion
    }];

    var options = {
        flat: true,
        limit: 100,
        onNotify: function (data) {
            trata_datos_fuente(data);
        },
        onSuccess: function (data) {
            ngsi_subscriptionId.push(data.subscriptionId);
        }
    };

    // Code to send subscription request using Orion API
```

15.5 Trials and Validation

In the previous image is the user interface created in Wirecloud for this project. The components shown in it, can represent the data retrieved (locations and measures) from the CB and the alarms (in red) detected by the CEP.

This map is other part of the mashup, in it some components query and subscribe to the CB to retrieve info about mass population in zones of interest. After, this info is shown in the map using the CB data and the WireCloud tools and APIs available in FIWARE.

If we consult the mashup wiring, the result is the next image:
This image gives us an idea of how the components developed for this mashup, work and relate to each other. In the next page there is a small description of each component and their functions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type of WC component</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Posición Fuentes desde CB</td>
<td>Operator</td>
<td>Find all the entities we need from the context broker and it subscribes to their position. Sends that info to other components.</td>
</tr>
<tr>
<td>Map Viewer</td>
<td>Widget</td>
<td>This map shows all the entities positions given and permit to click and select on one of showed entities. Forwards to other components, the info of the selected entity.</td>
</tr>
<tr>
<td>Medida Fuente desde CB</td>
<td>Operator</td>
<td>This operator receives an entity ID and retrieves and subscribes to the CB's data that we specified to consult. After that, it adapts that info to be forward to other WC components.</td>
</tr>
</tbody>
</table>
### Datos Fuentes

**Widget**

This widget receives all the measures and alarms sent to it, and it shows it to the final user.

### Consulta Alarmanas CEP

**Operator**

This component consults the alarm entities that CEP updates, to find changes in the alarm's status and send this status to other parts of the mashup.

### Datos Cámara

**Operator**

It query and subscribes, to the entities related with the amount of people in the zones of interest that we can see in the map.

### Mapa Control de Masas

**Widget**

This map shows the zones of interest and the status in these zones about the amount of people in them.

---

The data from the fountains is being stored via Big Data and can be accessed using a webhdfs client:

- **http://XXXXXXXX/webhdfs/v1/user/opendata/smartcities/sevilla/adevice/Sevilla_FUENTES_1_13_140-sevilla_fountain/Sevilla_FUENTES_1_13_140-sevilla_fountain.txt**

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Set-Cookie: hadoop.auth="xxxxx"; Version=1; Path=/

Content-Type: application/octet-stream

Content-Length: 1126

Date: Mon, 15 Sep 2014 15:36:26 GMT

[...]
Certain alarms are set by CEP to change the appearance of WireCloud gadgets, the configuration is done via web frontend from PROTON

15.5.1 Development issues

- FIWARE’s token expiration time, force us to update all the REST request commands, each time it expires.
- Some REST request options are not available via Wire Cloud (example, pagination) so the functionality via REST and WC is not the same. The second one is a step back respect to the first one.
16 Sevilla (Spain): A smart-city application for detection of crows through video cameras.

16.1 Scope & Requirements

This document describes a crowd detector system that has been created for demonstrating the capabilities of the FIWARE Stream Oriented GEri (Kurento) for smart city applications. This system captures video stream from city video surveillance cameras and uses complex computational vision algorithms for detecting when a crowd of people is being formed. We understand a crowd as a group of people moving very slowly or not moving at all. The system detects different levels of crowdness and published them into the FIWARE Context Broker (Orion), so that applications can subscribe to them and execute specific actions based on thresholds or other mechanism. The system makes also possible to visualize the video streams from any web browser compatible with WebRTC standards. This system opens interesting applications for video surveillance in smart cities where unsupervised cameras can be scanning relevant areas of the cities detecting problems as they happen (may city problems in crowded areas create fast a crowd of observers) and generate alarms to the appropriate security responsible, who can visualized the video stream from an WebRTC compatible browser (including smartphones).

16.2 GE used and architecture

The system architecture is depicted in the following figure
The architecture comprises the following components:

- The camera, which captures image on the city streets and feeds these images using the RTSP protocol to the Kurento Media Server, hosted at the FIWARE LAB.
- Kurento Media Server, which provides the media processing capabilities for decoding the stream, and applying the computer vision algorithms to it. As a result, Kurento Media Server publishes Media Events with the relevant media analysis results (i.e. levels of crowdness).
- Application Server: Provides intelligence to the application for managing Kurento Media Server capabilities and integrating the Media Events published by it with the Orion Context Broker.
- Orion Context Broker: is an implementation of the Publish/Subscribe Context Broker GE, providing the NGSI9 and NGSI10 interfaces.
- Alarms: represents subscriptions from external applications to the events published by the Context Broker.
- WWW browser: enables visualizing the video stream from any WWW browser compatible with WebRTC standards.

### 16.3 Kurento Media Server and the Crowd Detector Filter

The Crow Detector Filter is a Kurento Media Server module providing the capability of detecting crowds into the video stream. This filter is based in the comparison of temporal textures. It uses an algorithm called Local Binary Pattern [1], which compares in regions of 3x3 pixels the texture of the central pixel with the adjacent ones (8 neighbors) and obtains a average degree of similarity, as shown in the following figure.
The filter evaluates temporal changes in texture. Using a threshold mechanism, the LBP is capable of detecting whether there are relevant movements on each image area. From this, the filter generates a texture map and learns to distinguish static from moving areas.

For determining direction of movement, we use the Optical Flow algorithm. More specifically, the Lucas-Kanade method basing on a image pyramid of 3 levels. This method is implemented in OpenCV in the function calOpticalFlowPyrLK[2].

These algorithms are wrapped as a Kurento Media Server module called CrowdDetectorFilter (accessible as part of Kurento source code [3]), which exposes an interface enabling application developers to customize the internal workings of the algorithms through the following parameterizations:

- **ROIs**: a ROI (Region Of Interest) is a special region on the viewport (on the image provided by the camera) where the algorithm shall be applied. ROIs can be specified list of points, so that each ROI is a polygon joining the points in the order they are provided and which is closed by joining the last with the first point.
- **occupancyLevelMin**: minimum occupancy percentage of the ROI to send occupancy events.
- **occupancyLevelMed**: generates a occupancy level event with value “1” if the occupancy percentage is between the minimum and this level.
- **occupancyLevelMax**: generates occupancy level 2 if the occupancy is between the last level and this level. If occupancy is over this maximum, an event with level equal 3 is generated.
- **fluidityLevelMin**, **fluidityLevelMed** and **fluidityLevelMax** play the same role but with the fluidity parameter.
- **fluidityNumFramesToEvent**: number of consecutive frames that a new fluidity level has to be detected to recognize it as a fluidity change.
- **sendOpticalFlowEvent**: enable/disable movement direction detection into the ROI
- **opticalFlowNumFramesToEvent**: number of consecutive frames that a new direction of movement has to be detected to recognize a new movement direction.
- **opticalFlowNumFramesToReset**: number of consecutive frames in order to reset the counter of repeated directions.
- **opticalFlowAngleOffset**: sets the angle relative to which direction events should be interpreted.
• In addition, the filter as a **processingWidth** parameter which makes possible to define the resolution to which the processing shall be applied. By default, it is set to 320 pixels. Increasing this resolution increases CPU consumption of the algorithm.

### 16.4 The Application Server: integrating with the context broker

The Application Server is based on the Java Spring framework and contains the crowd detector application, which uses the Kurento API for instantiating and controlling the CrowdDetectorFilter instance. It is capable of publishing events, raised by the crowd detector filter, into the Orion Context Broker. This is achieved by means of a module that provides all functions and data structures needed. The module is uncoupled from the application, and can be used in any Java application that needs to push information into the context broker. It can also be used as a Spring component, declared as `@Bean` in the application `@Configuration` class along with other beans, as is the case in the Crowd detector application.

The main class in the module is the `OrionConnector` class. This class is the one that has be instantiated or declared as bean by users. The class exposes the following methods that map to analogous methods available in the context broker:

- `registerContextElements`: Registers a context element, also called entity. This should be the first step of an application that wants to publish events in Orion, since no subscriptions can be done by Orion clients, until the entity is registered.
- `updateContextElements`: Updates information relative to a certain element. The update can be in one or more of the attributes of the entity.
- `deleteContextElements`: Deletes all information relative to an entity.
- `deleteContextElementAttribute`: Deletes an attribute from a certain entity.
- `queryContext`: Queries the context broker for information about an element by it’s ID.
- `queryContextWithPattern`: Performs a query in Orion, to retrieve information about a set of entities.

The Context Broker offers a wider range of operations, but only the aforementioned are available through the module, as it is considered to be a publisher module. Thus it is not possible to register event listeners, or query elements on certain attributes. However, it could easily be expanded to provide all the methods exposed by Orion.

As mentioned before, the module is used in the application as a Spring bean, that is injected in the `CrowdDetectorOrionPublisher` class, which is also declared as a bean. The publisher is a wrapper of the `OrionConnector`, decoupling the rest of the application from the publishing module. The methods exposed by the publisher, are convenience methods that take events from the crowd detector filter, and transform them into Orion entities, to be later sent to Orion. All this process is hidden from the developer of the module for simplicity, and the API exposes the following methods:
publishEvent: This method is overloaded to publish all possible events raised by the crowd detector filter, which are:

- CrowdDetectorDirectionEvent
- CrowdDetectorFluidityEvent
- CrowdDetectorOccupancyEvent
- registerRoisInOrion: This method receives all the ROIs in the filter. For each region, it registers three entities (one for each event).

In order to register entities in Orion, the application uses the following information provided by the filter:

- Type: The type of event raised by the filter.
- ID: The ID of the Region of Interest (ROI) where the event has been detected.
- Attributes: Information relevant to each event (direction, fluidity...)
  - CrowdDetectorDirectionEvent
    - Angle of displacement.
  - CrowdDetectorFluidityEvent
    - Level of fluidity
    - Percentage of fluidity
  - CrowdDetectorOccupancyEvent
    - Level of occupancy
    - Percentage of occupancy

Thus, an Orion client can register a listener for events of a certain type in a certain ROI or in all ROIs, if the registration uses a pattern. Working with listeners in Orion is out of the scope of this document, since it is already covered in the official documentation, and the module does not offer any special method for this.

Registering the entities in Orion takes place when the application starts, and the Pipeline bean is instantiated. All ROIs are passed to the registration method, which registers all three possible events for each of the configured ROIs. When the filter raises an event, the overloaded publishEvent method from the CrowdDetectorOrionPublisher is invoked. The method transforms the event in an OrionContextElement, and the event is sent to the context broker by the OrionConnector. Then, Orion will invoke all listener subscriptions for which the event fulfills the subscription condition.

16.5 Demonstrating and validating the application

For demonstrating the crowd detector system we performed the following steps:
- We created a simple WWW application, and deployed it at the Application Server described above, capable of visualizing the video stream coming from the camera, the ROIs as well as the occupancy levels. This application implemented the Kurento Media Server pipeline depicted below.
- We installed an IP security camera in the city of Sevilla to feed the video stream to the application.

![Kurento Media Server pipeline for the demonstrator application.](image)

The camera we deployed is an AXIS Q1765-LE [4, 5, 6]. It proves RTSP connectivity using the following URL:

- `rtsp://<camera_ip_address>/axis-media/media.amp`

The camera was configured with the appropriate security for disabling unauthorized access. The camera was placed in an Areal position at the “San Francisco” Square in Sevilla.

![View from the IP camera at the “San Francisco” square in Sevilla.](image)

We successfully connected the IP camera to the FIWARE LAB deployment containing architectural description depicted in Figure 45.
As a result of this, we are able to validate the system and the demonstrator application. However, the link quality connecting the IP camera to the Internet seems to provide quite degraded quality, which depends significantly on the time of day. Probably this is due to the fact that the camera is connecting to a network shared with officer workers and other services and does not provide any types of QoS guarantees.

As a work around, we decided to find an alternative video source. There are several services offering webcam views of cities all around the world. These webcams can be accessed using the RTMP or the HLS protocols, which are not supported by Kurento Media Server. For this reason, we created a RTSP protocol adaptor basing on the Videolan player (VLC).

![Kurento CrowdDetector](image)

**Figure 49. Example of the demonstrator application working using a video feed from a public webcam showing Madrid “Puerta del Sol” square.**

In order to adapt the HLS streams from street webcams into an RTSP stream VLC is configured as indicated below:

- A stream file is created with the URL of the webcams that will be converted. The URLs have m2u8 extension, indicating a HLS playlist
  - new sol vod enabled
  - setup sol input [http://178.32.143.245/2903JN85/2903JN85.m3u8](http://178.32.143.245/2903JN85/2903JN85.m3u8)
  - w pantheon vod enabled
  - setup pantheon input [http://178.32.143.245/0801KR86/0801KR86.m3u8](http://178.32.143.245/0801KR86/0801KR86.m3u8)
  - new navona vod enabled
  - setup navona input [http://178.32.143.245/1102SKXD/1102SKXD.m3u8](http://178.32.143.245/1102SKXD/1102SKXD.m3u8)
• VLC is started as RTSP server with command bellow:
  o vlc -I telnet --telnet-password kurento --vlm-conf streams.vlm --rtsp-host 0.0.0.0 --rtsp-port 5554
    where streams.vlm is the file created in step 1

Kurento can access street webcams through RTSP stream in the following URL

• rtsp://vlchost:5554/sol

As a result, we were able to validate our system and application from different city cameras all around the world.

16.6 References


17 Torino (Italy): Monitoring Non-Emergency Calls in Torino Smart City

17.1 Scope & Requirements

The activity carried out by the Torino unit addressed the Urban Security issue in the Smart City context. The main objective of the Torino unit was to design and develop a software prototype including (i) an informative dashboard and (ii) a notification system on issues concerning urban security both exploiting open data by Torino Municipality that have been made available on FIWARE Lab. The prototype aims at supporting the interested city actors in planning strategic actions in a Smart City environment. The SpagoBI Business Intelligence suite, soon becoming FIWARE GE, and the CAP Context Broker FIWARE GE have been exploited to develop the software prototype.

The activity of the Torino unit addressed the analysis of non-emergency calls made by citizens to the Local Police’s Contact Centre in Torino. This data has been selected because it is strongly related to citizens’ participation and engagement on urban quality of life, which is a prominent issue in the Smart City context. Through the offline analysis of the received service requests using the developed prototype, the Torino Municipality experimented a way to monitor the citizens’ perception on urban security. The results of the analyses can be exploited to (i) plan targeted actions for improving security policy making, (ii) increase the perception of citizens’ security, and (iii) enhance the co-operation between citizens and the municipality. Since the selected non-emergency calls dataset was not available yet as open data, it has been firstly published as open data on the Torino Municipality Open Data portal. Then, it has been made available on the FIWARE Lab as a result of the project activities.

The Torino Municipality declared its interest in the project and supported the connection to the FIWARE Lab by performing the following actions: (i) active contribution to the activities related to the publication of Open Data from Torino and (ii) full support to the experimentation of FIWARE technologies for the development of applications/services in the Smart City domain.

17.2 GEs used and Architecture

Figure 50 reports the architecture of the developed software prototype. It contains an informative dashboard, realized using the SpagoBI Business Intelligence suite (www.spagobi.org), and a notification unit, realized using the CAP (Context Awareness Platform) implementation of the Context Broker FIWARE GE. SpagoBI implements functionalities which will soon be provided as FIWARE Generic Enabler. In the following, the main functionalities of the presented architecture are described. The source code and additional material is available on Github at

https://github.com/fiwareTorinoUnit/fiware-Torino
Figure 50 High-level architecture of the software prototype

The architecture shown in Figure 50 is used for the offline analysis of open data on non-emergency calls made by citizens to the Local Police’s Contact Centre in Torino. To analyze data from different viewpoints, the non-emergency calls open data collection has been enriched with extra contextual information as topologies and demographic statistics about the city areas and districts acquired from the official GeoPortal of the Torino Municipality (www.comune.torino.it/geoportale). Integrated information allows analyzing non-emergency calls from two main viewpoints: the spatial dimension and the temporal dimension of service requests. More details are reported in Section “Open data”.

According to the integrated data, an informative dashboard is generated based on a selection of Key Performance Indicators (KPIs), aimed at providing information on the citizens’ perception on urban security. For example, to reveal potentially critical situations in a specific urban area, the incidence per city area/district/green area of the received requests for a given category/subcategory of emergency calls has been analyzed. Furthermore, to keep track of the temporal evolution of the citizen perception on urban security, the variation between two time periods in the number of calls received from a given city area/district/green area is analysed. To avoid bias due to the unbalanced distribution of citizens per city area/district, the aforesaid KPIs are also normalized by the number of citizens per city area/district.

The informative dashboard visualizing KPI values has been developed using SpagoBI 4.2, an open source Business Intelligence suite. KPIs are showed using charts and maps. Specifically, the GeoReport Engine
4.2 of SpagoBI has been used to effectively display KPIs on top of geographical maps. This representation provides an insightful view of the received service requests in different urban areas.

The notification unit allows the Municipality staff to monitor the variation of the citizens' perception on urban security over time in specific urban areas. Notifications related to specific service requests on a given urban area are selectively forwarded by email to the municipality actors in charge of the specific issue, who may decide to plan targeted actions. To generate notification messages, the system considers KPIs on the variation of received calls between consecutive time periods per city area/district/ green area. Based on KPI values, the severity level of the warning is classified as stable, substantial or critical increase, and decrease, respectively.

The notification system is based on the Context Awareness Platform (CAP) Context Broker, a Publisher/Subscriber implementation. Three additional Java-based units support the CAP activity. The architecture of the notification system is reported in Figure 51. Specifically, the subscription unit specifies to CAP the KPIs of interest for the city roles. The context source periodically provides updated KPI values to CAP. When a new KPI value matches one subscription condition, CAP notifies to the corresponding subscriber (i.e., city role) the KPI value. The third unit allows users to register into the notification system, by specifying their role and email address. This unit receives notifications from CAP for a given KPI value and city role, and then it forwards them to all the users registered into the system for that role.

![Figure 51 The notification system](image)
17.3 Open data

The information related to the non-emergency calls made by citizens to the Local Police’s Contact Centre in Torino has been selected for the activities carried out in the project and to be published as Open Data, as it implies a number of elements rather relevant in the context of Smart Cities:

- it is strongly related to citizens’ participation and engagement on urban quality of life;
- it gives relevant opportunities to develop solutions targeted to improve security policy making, to enhance the co-operation between citizens and police, and to increase feeling of security;
- due to its inherent nature, this information is highly sensitive thus it requires to cope with issues related to its publication and use.

Concerning the latter element, the information about citizens’ engagement on urban security was not available as open data; as a result of the activity performed, it has been finally published as open data. The open data has made available in FIWARE Lab within the corresponding open data section (managed through CKAN GE).

The following Tables 1 and 2 describe the non-emergency calls dataset published as open data. The service requests, which this data set refers to, are currently collected by the Local Police’s Contact Centre through emails and phone calls. Table 1 lists the fields in the open dataset. To provide alignment of this data model with the Open 311 data model also applying to urban security issues, for each field in the open dataset the corresponding field in the Open311 specification (as in GeoReporting version 2, [http://wiki.open311.org/GeoReport_v2](http://wiki.open311.org/GeoReport_v2)) is reported in brackets. Table 2 lists the types of service requests included in the open dataset.
Future Internet Core Platform

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subcategory (service_name)</td>
<td>Human readable service request type</td>
</tr>
<tr>
<td>Category (group)</td>
<td>It provides a way to group several service request types under one category.</td>
</tr>
<tr>
<td>Address, District (address_string)</td>
<td>Street address and the corresponding district</td>
</tr>
<tr>
<td>Service request date and time (requested_datetime)</td>
<td>Date and time the request was created</td>
</tr>
<tr>
<td>Green area (additional field)</td>
<td>To denote if the service request refers to a green area</td>
</tr>
</tbody>
</table>

Table 1: Fields in the non-emergency calls open dataset

<table>
<thead>
<tr>
<th>Category</th>
<th>Subcategory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social tension</td>
<td>Vandalism; Other</td>
</tr>
<tr>
<td>Civil tension</td>
<td>Youth gathering; Disturbing behaviours; Disturbance from dogs; Disturbance from other animals; Disturbance from public venues; Noise nuisance; Improper use of common areas; Other</td>
</tr>
<tr>
<td>Urban quality</td>
<td>Urban blight and renewal; Abandoned vehicles; Other</td>
</tr>
</tbody>
</table>

Table 2: Type of services in the non-emergency calls open dataset

Additional datasets related to urban security issues and urban areas have been made available as open data in FIWARE Lab, even though they were not directly used in the project. These datasets are available as open data to the Torino Municipality.

To allow the offline analysis of non-emergency calls from different viewpoints, in the project the original open data schema reported in Table 1 has been enriched with extra contextual information.

To analyze the spatial distribution of non-emergency calls, beyond the location addresses and districts of the service requests, each location address is mapped to the corresponding city
area, as additional contextual feature to analyze data at an intermediate granularity level. For service requests referring to a green area, the green area name and type (e.g., park) are added. Moreover, to analyze the *temporal distribution* of non-emergency calls, beyond the date of the service request, the month, 2-month, 6-month, and yearly time periods were considered, and the day is classified as working or holiday.

*Topological* and *demographic information* about city areas, districts, and green areas are integrated as well. Topologies are used to graphically analyze the most significant spatial trends in non-emergency calls, while demographic information is exploited to characterize non-emergency calls according to the distribution of citizens per area and district. Demographic statistics and topologies about the city areas, districts and green areas, were acquired from the official GeoPortal of the Torino Municipality (www.comune.torino.it/geoportale).

### 17.4 Trials and Validation

The software prototype was implemented and tested on a local platform. SpagoBI was locally installed in the version 4.2. For the CAP Context Broker, an instance was locally installed and accessed.

The usage of CAP Context Broker for the activity carried out in the project has been made easier by the documentation available on FIWARE mediawiki for this GE implementation. This documentation significantly supports in the comprehension of the CAP functionalities and how to use them. In addition, it guides in the CAP installation through a detailed explanation of all steps.

SpagoBi is a Business Intelligence suite providing various functionalities. GeoReport Engine 4.2 represents a valuable unit in the SpagoBI suite, allowing the geolocalized representation of KPIs. It is a GIS engine to produce georeferenced reports, connectable to WFS sources or loading geographic boundaries from text files. In our application scenario, it is particularly useful for mapping non-emergency calls to the corresponding city area/district/green area. The usage of the SpagoBI suite has been supported by the available documentation and a Forum to answer to specific developer questions.
The following specific issues and difficulties were found in using the software during the project:

<table>
<thead>
<tr>
<th>Issue</th>
<th>FIWARE Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Graphical User Interface (GUI) is not available. Although the</td>
<td>CAP Context Broker</td>
</tr>
<tr>
<td>documentation on CAP available on FIWARE mediawiki properly supports</td>
<td></td>
</tr>
<tr>
<td>the developers both in setting up and using CAP, a GUI might improve</td>
<td></td>
</tr>
<tr>
<td>its usability.</td>
<td></td>
</tr>
<tr>
<td>CAP Context Broker is a Publishing – Subscribing component where</td>
<td>CAP Context Broker</td>
</tr>
<tr>
<td>requests for content writing and reading are performed by exchanging</td>
<td></td>
</tr>
<tr>
<td>messages with CAP. The provided documentation, including various</td>
<td></td>
</tr>
<tr>
<td>example cases, significantly supports in understanding how exchanging</td>
<td></td>
</tr>
<tr>
<td>messages with CAP.</td>
<td></td>
</tr>
<tr>
<td>In those cases where the documentation wasn’t of help to understand</td>
<td>CAP Context Broker</td>
</tr>
<tr>
<td>an issue encountered, a direct contact with the GE owner was</td>
<td></td>
</tr>
<tr>
<td>established, and this helped a lot to rapidly overcome the issues.</td>
<td></td>
</tr>
<tr>
<td>In case of syntax errors in exchanged messages, the CAP Context</td>
<td>CAP Context Broker</td>
</tr>
<tr>
<td>Broker returns a NACK message with details on the error. This</td>
<td></td>
</tr>
<tr>
<td>aspect is a good support in the development of applications.</td>
<td></td>
</tr>
</tbody>
</table>
17.5 Screenshots

Figure 52, Figure 53 and Figure 54 report some example screenshots for the informative dashboard. They show the distribution of service requests on different urban areas (city districts, green areas, city areas) using maps (Figure 52, Figure 53) and bar diagrams (Figure 54).

Figure 55 reports a screenshot related to the notification system. It shows the webpage used to set up the KPIs of interest and reference threshold values for the city roles.

![Example of informative dashboard](image-url)
Figure 53 Example of informative dashboard

Figure 54 Example of informative dashboard
Figure 55 Setting up KPIs of interests and reference values for city roles
18 Trento (Italy): Smart Campus

18.1 Scope & Requirements

The activities of the Trento Smart City pilot addressed three main objectives. First of all, we integrated the global instances of the FIWARE Lab GEs with the Trento installation of an existing Smart City platform, namely the so called “SmartCampus” Open Service platform, which aims at delivering, through Open APIs, real-time data and interactive services provided by the administration. The use case scenario for testing such an integration consisted of collecting data coming from different sources and, after opportune aggregation, of showing them on a cockpit, to monitor the evolution of some indicators related to traffic and mobility - such as the occupation of parking places and the usage of shared bikes. As a second objective, we wanted to verify the possibility of reproducing the same scenario based on an identical architecture but with a different deployment in which all the components, including the FIWARE GEs, were installed locally. Finally, our last goal was of making available to the FIWARE Lab developers the full set of open data produced and managed by the Municipality and the Province of Trento, by mirroring them onto the FIWARE Lab CKAN-based data portal.

18.2 GEs used and Architecture

The pilot activities related to the exploitation of the FIWARE components were split in two: on the one hand, we integrated the remote global instances of FIWARE GEs with the “SmartCampus” Open Services platform and, on the other hand, we integrated a set of locally installed instances of the FIWARE GEs with the “SmartCampus” Open Services platform itself.

18.2.1 Integration with FIWARE Lab GEs

In this configuration (see Figure 56) we used the FIWARE Lab installation of the Orion Context Broker and Cosmos (with Cygnus connector) GE, and we deployed the demo application onto the FIWARE cloud infrastructure.

We created a virtual machine in FIWARE Lab cloud using a Orion Context Broker image, on which we installed the Cygnus connector and where we deployed a mobility cockpit (a web application implemented with Node.js).
We extended the “SmartCampus” Open Services Platform so that its backend services could forward information about parking places occupancy and shared bikes availability to the Orion Context Broker installation (see Figure 58 and Figure 59).
Figure 57 An extract of data about parks availability inserted into Orion
Figure 58 an extract of data about bikes availability inserted into Orion

With the help of the Cygnus connector the data are stored onto the Cosmos GE global instance (see Figure 60).
Figure 59 Orion data saved by Cygnus in HDFS
We used Hive SQL queries to perform the map/reduce routines which aggregated data that are on the one hand shown in the mobility cockpit (see Figure 61) and on the other hand are pushed back to Orion Context Broker as overall average of shared bikes to become a piece of the global context (see Figure 62). We also used the Wirecloud GE to show the data on a graph but, owing to the issues reported later in this document, we were not able to use it as the mobility cockpit.
18.2.2 Integration with locally installed GEs

In this configuration (see Figure 63) we were able to successfully install locally the Orion Context Broker and the Cygnus connector FIWARE components. Though, the installation of Cosmos failed for the reason explained in the last section. Hence, we replaced the Hadoop functionalities featured by Cosmos with a Hortonworks Hadoop sandbox.

The execution flow is almost the same as in the previous case but for the elaboration of the historical data through map/reduced routines. In this case we took advantage of the tools offered by the Hadoop
ecosystem. In detail, we defined a workflow using PIG scripts and the Sqoop tool. PIG scripts load data from the HDFS data folders, process them and save the result into a target folder; the Sqoop tool moves them from the final folder to a RDMS database. The web dashboard access this database to show the charts in the mobility cockpit. We scheduled the recurrent execution of the workflow using an Oozie scheduler.

The source code developed for the implementation of these two version of the use case scenario is available on Github at the following URL:

- https://github.com/smartcampuslab/fi-ware

In particular it consist of the following modules:

- Backend of service integrated with Orion (under Apache 2 license)
  https://github.com/smartcampuslab/fi-ware/tree/fi-ware/smartcampus.vas.parcheggiausiliari.web
- Web dashboard (under MIT license)
  https://github.com/smartcampuslab/fi-ware/tree/master/demo-fronteend

18.3 Infrastructure and equipment

We used two different types of sensors to collect the data processed within the use case scenario. For the data about the occupancy of parking places we used Android devices on which the app 'ParkingCounter' had been installed. Such an app - developed for the Municipality of Trento - supports
traffic operators in notifying the availability of parking slots along roads and city squares. For the data about the usage of shared bikes we have integrated the information coming from the relevant monitoring system. The latter is a third-party system on which the Municipality of Trento relies, as part of the infrastructure of the shared bike service that it offers to its citizens.

18.4 (Open) Data

We accomplished the goal of publishing the huge amount of open data (currently 867 datasets) that the Municipality and the Province of Trento produce and manage (see Figure 64 Trentino open data sets on FIWARE Lab data portal).

![Figure 64 Trentino open data sets on FIWARE Lab data portal](image)

We achieved this result by mirroring the CKAN-based open data portal of the Trentino province ([http://dati.trentino.it/](http://dati.trentino.it/)) onto the CKAN-based instance of the data portal of the FIWARE Lab ([https://data.lab.fiware.org/](https://data.lab.fiware.org/)).

We implemented a generic procedure to perform both the initial import of the data and to perform subsequent periodical, and manual, updates between two CKAN instances. Such operation can be triggered using a simple interface (see Figure 65, Figure 66 and Figure 67) and requires the user to authenticate to the destination portal.

The import/update procedures are implemented as Python script that relies on CKAN API version 3.
Figure 65 CKAN mirror tool home page

Figure 66 CKAN mirroring tool create mask
The source code and the technical documentation can be found at the following URLs:

- https://github.com/smartcampuslab/fiware/tree/master/ckan-harvester
- https://github.com/smartcampuslab/fiware/wiki

The problems encountered during the implementation are summarized in the next section.

## 18.5 Trials and Validation

The goal of the Trento Smart City pilot activities was to test the integration of already existing (and working) pieces of IT resources with FIWARE components. In particular, we can distinguish two kinds of such integration: the one related to IT platforms and the one related to open data. The former - focused on having the Open Service platform and FIWARE GEs to co-operate - occurred in two flavors: one using the global FIWARE Lab instances of the FIWARE GEs communicating remotely with the Open Services platform and the other using installation of the FIWARE GEs co-located with the Open Service platform. The latter consisted in integrating a generic CKAN-based open data portal with the data portal maintained within FIWARE Lab.

All the integrations mentioned above were tested and validated through the adoption of opportune scenarios. In the case of the platform integration, the Municipality of Trento proposed some use cases centered on its concrete interest in improving the mobility policies that are currently adopted with respect to parking and shared bikes. We implemented such scenarios and helped the Municipality to
execute them. A tight collaboration was established with the Municipality for the exploitation of the the systems and sensors that were available. The scenarios are described in the previous section with the help of screenshots. A video is also available here:

- [http://www.smartcampuslab.it/download/fi-ware/FIWARE_Trento_Smart_City_demo.ogv](http://www.smartcampuslab.it/download/fi-ware/FIWARE_Trento_Smart_City_demo.ogv)

For the open data integration, the validation consisted in executing the mirroring procedure using, as source, the open data portal of the Province of Trento.

Overall the test and validation activities were successful. They produced the expected results and they were completed within the awaited time frame and using the foreseen number of resources. The data part was indeed very easy..

Nevertheless, we feel like mentioning the most important issues that we encountered and that, from our perspective, should be carefully taken into account when porting the FIWARE technologies from research projects to a more widespread adoption:

- **GE adoption:**
  - Installing the Cosmos GE using the source code took quite much time and could not be completed but using undocumented workarounds; this is quite limiting since, currently, the usage of the global instance of the Cosmos GE cannot be considered sufficient not even for demo purposes owing to its shared nature; more in general, the capability of granting a local integration of FIWARE pieces with other already existing platform is an important aspect;
  - The adoption of the Wirecloud component, which might have been very useful to save time in the results display activities, was not feasible; though we, apparently, used it correctly, following the instructions, no results were shown in the graphs and no error message appeared to give a rationale for the failure;

- **Support:**
  - We submitted all the issues we encountered to the different channels made available by the project (stackoverflow dedicated channel, Jira ticketing system, support mailboxes, individual mailboxes) but in most cases we did not get actual attention and satisfying replies to our questions; the quality of the support should probably be somehow enhanced.

A more specific list of issues can be found in the following table:

<table>
<thead>
<tr>
<th>ISSUE</th>
<th>COMPONENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH connection to Cosmos global head node returns error “Permission denied”</td>
<td>Cosmos</td>
</tr>
<tr>
<td>Documentation present errors and is not up-to-date:</td>
<td>Cosmos</td>
</tr>
<tr>
<td>- within Cygnus connector section it is advised to</td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td>Platform</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Install release 0.1 of Cygnus (not supporting multiple attribute), but the last one is 0.4 - Cygnus installation test is wrong</td>
<td></td>
</tr>
<tr>
<td>Possibility to allocate only one floating IP to own cluster machines is poor.</td>
<td>Cloud portal</td>
</tr>
<tr>
<td>The known problem on Spark cluster in global instance blocked the development for almost two days. It should be at least documented.</td>
<td>Cosmos</td>
</tr>
<tr>
<td>Error installing cosmos: we have an error executing puppet installation script on Cosmos master node during cosmos-admin installation. Nobody has resolved our issue.</td>
<td>Cosmos (LOCAL INSTALLATION)</td>
</tr>
<tr>
<td>Wirecloud currently can use only Orion GE as a data source for the widgets. The same should hold for Cosmos GE as well.</td>
<td>Wirecloud</td>
</tr>
<tr>
<td>The feedback provided by Wirecloud in case of errors is not existent at all.</td>
<td>Wirecloud</td>
</tr>
<tr>
<td>Creating a dataset in lab.fiware portal with CKAN API version 3 currently we receive an error “403 User is not authorized to edit these groups.”</td>
<td>FIWARE CKAN data portal</td>
</tr>
<tr>
<td>Updating an existing dataset in lab.fiware portal with CKAN API version 3 we receive an error “500 Server Error. An internal server error occurred.”</td>
<td>FIWARE CKAN data portal</td>
</tr>
<tr>
<td>Creating a group in lab.fiware portal with CKAN API version 3 we receive an error “403 User is not authorized to edit these groups.”</td>
<td>FIWARE CKAN data portal</td>
</tr>
<tr>
<td>Updating an existing group in lab.fiware portal with CKAN API version 3 we receive an error “403 User is not authorized to edit these groups.”</td>
<td>FIWARE CKAN data portal</td>
</tr>
</tbody>
</table>
19 Valencia (Spain): Smart Taxi

19.1 Scope & Requirements

The main objective of the pilot was to prepare a working prototype software to:

- **Gather** location data from all taxis in our network with Orion Context Broker.
- **Store** it automatically with Cygnus in Cosmos.
- **Analyze** the taxi location data to forecast the demand in the city. A dedicated instance was setup using FIWARE Cloud environment.
- **Display** a heat-map directly to the taxi driver Smartphone, the mobile application reads from Object Storage.

19.2 GEs used and Architecture

The FIWARE **generic enablers that will be used** in the project are the following:

- Context Broker GE for context-awareness.
- Big Data GE for data storage and processing.
- Proton Complex Event Processing GE for pattern detection and action triggering.
- Virtual machine instance for proprietary algorithm execution
- Object Storage GE for final result storage
19.2.1 Architecture overview:

We rely on the Orion Context Broker to receive and query data of thousands of taxis each second. It is an easy, robust and scalable way to solve a hard problem on concurrent sensor data inputs. The Orion entities are individual taxis, where each of them will hold information about its location and if it is taken or not. This information will be queried later to calculate the next forecast.

Once the data is received, we use Complex Event Processing to transform data and coordinate the data input.

After data is gathered, we use Cygnus to store it automatically in Cosmos Big Data GE in several instances. Forecasts models need several data aggregations and Cosmos storage and querying capabilities allow working with BIG DATA easily without managing heavy data logs or txt files using utilities as Hive.

We use R and C for processing the predictive models, so we use virtual machine instances for these functions that aren't available in GEs, so all the infrastructure is kept in the FIWARE cloud.

Forecasts results are stored on one hand in Orion so it is used for model learning process and on the other hand in the Object Storage GE so the taxis mobile application read always from a single file without multiplying the query load of the server.

19.3 Infrastructure and equipment

We receive data from two different kinds of sensors:

- Data from fleets: taxi fleets send us their data gathered through sensors placed in taxis (taximeters).
- Data from the Smartaxi App: users of the service send their location data while using the app

19.4 Open Data

There are two main datasets used by or produced from this service:

**Input data:**

The raw data received from fleets and taxi drivers (as explained previously) which consists in the position of a taxi at a certain time and if it’s booked or not. This data is stored in an Orion Context Broker entity in the following format:

```json
{
  "contextResponses": [
    {
      "contextElement": {
        "type": "Taxi",
```
"isPattern":"false",
"id":"1907",
"attributes": [
  {
    "name":"time",
    "type":"timestamp",
    "value":"1369004400"
  },
  {
    "name":"lat",
    "type":"coord",
    "value":"4122.9238N"
  },
  {
    "name":"lon",
    "type":"coord",
    "value":"00211.1181E"
  },
  {
    "name":"status",
    "type":"int",
    "value":1
  }
],
"statusCode": {
  "code": "200",
  "reasonPhrase": "OK"
}
}

**Output data**

The output data generated by our algorithms is stored in the Object Storage GE in a ZIP-compressed JSON file containing the different forecast results (5 and 20 minutes forecast, forecast by money and hourly forecast) in this format:

```json
{
  "map":"41.399765,2.020725;41.399765,2.020725;41.422680,1.997000;",
  "24h":[
    "11.0488",
    "3.6832",
    "6.8476"
  ],
  "time":1418493058,
  "kmean":"41.3871116638,2.1709804535,13",
  "money":"41.399765,2.020725;41.399765,2.020725;41.422680,1.997000;",
  "map20":"41.470917,2.054387;41.470917,2.054387;41.399765,2.020725;"
}
```

**Openness of the data**
This data may or may not be open depending of the business plan developed in a particular city. For example, in a city where there is no collaboration from city managers, data has to be private in order to avoid competition.

But in a business plan where a city makes our service available by assuming the costs instead of individual taxi drivers having to pay for it, the city managers may decide to make our data open so other services can benefit from it.

19.5 Trials and Validation

The platform was tested by comparing the results of the forecast produced in the FIWARE-powered platform and the ones produced in our current infrastructure. As the algorithms and processing are the same, the important part was ensuring data reception from the different sources (as explained above) and synchronization of the processes.

In terms of functionalities, in our case there are GEAs available for almost all of our needs (except our proprietary algorithm of course, but we use R which is available as a package in the creation of the VM instance)

The problems that we’ve had are those of a platform in development: lack or inconsistency of documentation in some components or lack of cloud resources like storage space or IP addresses for example. Most of them we were able to overcome, for a production environment, FIWARE infrastructure should be more reliable than actual lab.

Difficulties with the software:

The following specific issues and difficulties were found with the software/platform during the project:

<table>
<thead>
<tr>
<th>Issue</th>
<th>FIWARE Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation of Cygnus is not consistent between GitHub readme and the wiki. We ended up using a combination of both.</td>
<td>Cygnus connector</td>
</tr>
<tr>
<td>Cygnus stored the data from Orion Context Broker in Cosmos in JSON format and Hive was not loading the JSON deserializer. For this reason, the SQL Query to the table was wrong, only worked SELECT *. To solve this we loaded the JSON library before doing the Query.</td>
<td>Hive (Cosmos Big Data GE)</td>
</tr>
</tbody>
</table>
We couldn’t create a 10GB volume and the instance disk space wasn’t enough to store our platform, but we were provided with a special flavor of VM with more capacity.

| Cloud |

Initially we only had one public IP available so we had to access to the Proton web interface through a tunnel from the instance which has the IP assigned. We contacted the FIWARE team and were given another.

| Cloud |

The main VM instance suddenly stopped working, stopping development and data reception for a few days until the FIWARE team restored it.

| Cloud |

**Source code**

https://github.com/talleria/SmartaxiFiware

**Screenshot**

Link to an online video demo:

https://www.youtube.com/watch?v=h7cCS8E7d5I&feature=youtu.be
D.10.7 Smart Cities connection to FIWARE Lab
20 Valencia (Spain): Open data migration

20.1 Scope and Requirements

Inndea has developed and delivered the needed tools to proceed with the migration of data from the open source catalog of Valencia City Council (PDA) to live instance FIWARE Lab of FIWARE. These data are both static and dynamic. The first one have been migrated to the tool CKAN (CKAN) and the dynamic data to Orion (ORION)

20.2 GEs used and Architecture

The project mainly used two General Enables or tools: CKAN and ORION (Context Broker)

In a general sense, CKAN stocks dynamic data files enabling their upload so that other users might download them. Besides, depending on the kind of file (CSV, JSON, etc.) also offers additional functionalities, such as: graphic representation of a geopositioned data maps.

In one hand, CKAN uses the concept of organization as representation of an enterprise, institution or group of users. For the purposes of this project, the selected organization has been Valencia. In an organization are defined group of resources called datasets or packages, which are representations in CKAN of an URL or of a conventional file.

The organization valencia of CKAN presently contains 75 datasets organized in same thematic areas as the original PDA. In each dataset there used to be eight different resources. Some of them contain data files that can be downloaded by the users (CSV, JSON, SHAPE, ZIP, GML and XML. The rest are URLs of resources of the Valencia City Council (WFS, WMS).

In the other hand, ORION is a Context Broker (or context information manager). In a general sense, it stocks key values pair/pairs allowing to update and recover individual data. Besides, it enables additional services such as subscription and reception of changes notifications of selected data. In (GOR) there is the official guide of ORION and in CMF there is a description of the data model implemented b Orion

The migration of the static data is carried out in two steps: first takes place the configuration of CKAN and then the migration itself. For dynamic data there is another different procedure. All three procedures have been implemented in programmes using JAVA language. These programmes use the REST interfaces used by the data catalog of VALENCIA CITY COUNCIL to recover the requested data. Besides, these programmes use the REST interface used by the indicated GE’s.
20.3 Infrastructure and Equipment

The project does not use any specific device, for it only relies on the data catalog of VALENCIA CITY COUNCIL.

20.4 (Open) Data

20.4.1 Static data

The project has migrated 75 datasets to CKAN with data related to following category:

- Culture and Leisure (4 datasets)
- Education (1 dataset)
- Environment (28 dataset)
- Public Sector (3 datasets)
- Society and Wellbeing (6 databases)
- Traffic and transport (18 databases)
- Urbanism and infrastructures (16 data bases)
- Housing (1 dataset)

These data have been migrated to CKAN keeping the original organization of the VALENCIA CITY COUNCIL catalog.

Each dataset includes specific metadata describing their data. The metadata are indicated in the description of the dataset stocked in CKAN. All content is accessible either manually through the CKAN user’s interface, or automatically through the REST interface of CKAN.

Every dataset is served in one or several of these formats: CSV, JSON, KML, KMZ, GML, SHAPE, WFS and WMS. In the case of CSV and JSON, the underlying data model is given by the respective metadata. For the rest of formats the data represent geographical data in standards format designed by third companies, therefore the data model is given by them. It has to be taken into account that in the case of WFS and WMS resources, CKAN does not store the data, but it only keep the links leading to these data in the map server of VALENCIA CITY COUNCIL.

20.4.2 Dynamic Data

Four dataset have been created containing following information:

- Traffic and Transport: Car parks
- Traffic and Transport: Bike parks of the public bike rental service called “Valenbisi”
- Traffic and Transport: Real time information about traffic
- Traffic and Transport: intensity of traffic information
In all these cases the migration consists on recovering from the catalog of VALENCIA CITY COUNCIL the JSON version of these datasets and store them in ORION as key values. Given that the respective metadata are kept in CKAN, it has been considered unnecessary to also store them in Orion.

Related to the protocol and access mechanisms, these data are accessible through the REST interface made available by ORION (GOR). In this project has been used the JSON version, but the XML version is also suitable. Likewise, to acceee the data stored in ORION, the user can achieve it through both interfaces (JSON and ORION).

20.5 Trial and Validation

Given the complexity of the procedure and the difficulty of setting automatic test mechanisms, the validation works have been semi-automatic in two steps:

1. First, a manual validation to detect errors and any other problem that needed to be fixed. Once solved, the process is repeated until no problems arise
2. Secondly, REST has been used to carry out queries to CKAN and ORION. The results have been checked manually.

GEs CKAN and ORION make able the implementation of migration process in a proper way. The use of standards (web services REST, interoperability data format JSON, etc.) and the availability of technical support and documentation, have made possible the proper development of the validation tests. For this reason, the used GEs have been enough to carry this action out.

Finally, the procedures have been not addressed to the final user, so it is not possible to show any demo image. The procedure has enabled to migrate the data of the VALENCIA CITY COUNCIL catalog to store them in CKAN and Orion so that other programme can access them automatically

Nevertheless, as part of the communication actions of WP12, Inndea organized the event “Startup Weekend” in October 2014 where several real cases of projects using FIWARE technology were presented. The aim was to show to general public the benefices for the day to day life that through the launch of application using FIWARE technology. The event gone beyond of merely showing project examples, for included workshops where the attendant could learn how to reach and use the data available through FIWARE technology specially those provided by City Council. Furthermore, four real case studies competed for a price:

- Smartcart: FIWARE used in secure online shopping
- Drinkaway; FIWARE applied to leisure activities using traffic data
- Cross Cultural Collaboration: FIWARE implemented in the field of on-line teaching
- Fiscappacity: FIWARE overcome the accessibility barrier for disable people.
- Drinkaway was de teal case study awarded for its use of real time traffic data provided by City Council.

A video of the event can be checked [here](#)
20.6 References and Links

- [CKAN] CKAN de FIWARE LAB
- [ORION] Context Broker de FIWARE LAB
- [PDA] Valencia City Council OpenSource Data Portal
- [CMF] Context Management in FIWARE
- [GOR] Orion User Guide
- [GSRCH] Context Broker User Guide
21 Vigo (Spain): Integration of Vigo data sources into the FIWARE architecture

21.1 Scope & Requirements

The main objective of the project was to describe and analyze the data sources from the city of Vigo (described on section Open Data) that were available to feed the FIWARE architecture and integrate them in the platform.

There are not many public APIs with relevant information about Vigo, so a big part of the effort inside this project was to obtain the desired information from public websites, and process the available information in order to create a clean dataset that can then be published to the FIWARE platform.

Software has been created to gather the information available on the most relevant of those data sources. The interconnection architecture used to keep the information up to date in the FIWARE system is described in section Services architecture.

Finally, a simple demonstration WireCloud application was developed in order to validate the information retrieval process and to show a possible use case for the gathered data that is described in the Test and Validation section of this document.

21.2 GEs used and Architecture

The services developed to integrate the most relevant of the detected data sources in the FIWARE architecture.
21.2.1 Services architecture

Due to the diversity of the data sources, we have transformed them in order to get them integrated inside the FIWARE platform. A more detailed schema will be provided for each dataset in the Open Data section of this section.

21.3 Open Data

Two types of datasets were studied and are provided:

- **Static datasets**: The information provided is not updated frequently and thus can be published on the FIWARE platform as a static dataset using the CKAN platform.
  - For example: Vigo Budget Information is created on a yearly basis.
- **Dynamic datasets**: The information provided is updated regularly and thus should be published on the FIWARE platform through one of its GE that allows scheduling and dynamic content like the ContextBroker GE.
  - For example: Vigo’s cultural agenda.

21.3.1 List of the published datasets

- Vigo Budget Information (static)
- Vigo Historic Harbour ship movements (static)
- Vigo Points of interest (static)
- Vigo Beach Information (static+dynamic)
Future Internet Core Platform

- Vigo Pharmacies Information (static)
- Vigo Gas Station Information (static+dynamic)
- Vigo Climate information:
  - Vigo Weather Information (dynamic)
  - Vigo Air Quality Information (dynamic)
  - Vigo Seawater measurements (dynamic)
- Vigo RSS information:
  - Vigo Cultural events (dynamic)
  - Vigo Traffic announcements (dynamic)
  - Vigo News Information (dynamic)

The following subsections will describe in more detail the integration process for each dataset.

21.3.2 Vigo Budget Information

21.3.2.1 Description

Vigo Budget Information: Expenses and Income.

New information is available each year after the budget has been approved and published.

21.3.2.2 Relevance

From the city council perspective opening up its budget will enhance its transparency and accountability helping Vigo join the existing movement of openness that is rising among some of the most important cities around the world.

From the citizen point of view understanding a city budget is not an easy task, but with the help of developers inside the FIWARE platform the bulk information could be processed and visualized in order to give more insights on the most relevant information. As a result of this process citizens will be able to have a more active participation leading to a more mature democracy.

21.3.2.3 Data Location

The original information is published inside Vigo Council portal financial section.

Integration architecture schema
Developed components

To obtain the reports we have created two scripts:

The first one, informes.sh, is a bash script that uses sqlplus to connect to the council's Oracle database (with user, password, and instance of database) and executes a sql script. We need to configure 'ruta' variable to indicate the folder where we want to save reports. To run the script you have to indicate the year for which you want to obtain reports. For example: ./informes.sh 2013

The second file, querys.sql, is the sql script. This script executes a select sentence for each report to obtain data and dumps this data to a CSV output file. A CSV file can be opened in any program, however, for most users a CSV file is best viewed through a spreadsheet program such as Microsoft Excel or Open Office Calc.

Example output file:

<table>
<thead>
<tr>
<th>EJERCICIO;CENTRO GESTOR;ECONOMICA;DESCRIPCION;IMPORTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012;1320;2120000;REPARACION E MANTEMENTO OUTRAS CONSTRUCCIONS;40000</td>
</tr>
<tr>
<td>2012;1320;2130000;REPARACION E MANTEMENTO MATERIAL DIVERSO.;12000</td>
</tr>
</tbody>
</table>

21.3.2.4 Final Results

The information regarding Vigo Budget is being collected and will be available before the end of the year inside FIWARE CKAN public instance.

21.3.3 Vigo Harbour historic ship movements

21.3.3.1 Description

The Vigo Harbour is one of the main drivers of the city’s economy and one of the most important harbours in Europe.

The available information includes the list of ship movements from 2002 until now.
21.3.3.2  **Relevance**

Having access to the historical data of ship movements inside Vigo Harbour will provide developers inside the FIWARE platform with a very rich dataset to analyze using the tools offered by the platform, like the BigData/Cosmos GE.

There are more than 30,000 ship movements available since 2002 each of which provides very detailed information.

From the citizen’s point of view, using applications developed inside the FIWARE platform they could find out more information about Vigo Harbour, what routes are the most commonly used by ships that stop in Vigo, What are the most frequent visitors to our Harbour, etc.

21.3.3.3  **Data Location**

The original information is published inside Vigo Harbour portal [ship movements section](#).

Integration architecture schema

---

21.3.3.4  **Developed components**

**Static data:**

- A Ruby script was developed that uses web scraping techniques to extract the information for each ship movement and transforms it into a CSV file that has been uploaded to the CKAN public instance in FIWARE.
- We have decided to create a separate resource by year. Actually there’s data from 12 years from 2002 to 2013.

21.3.3.5  **Final Results**

The information regarding ship movements inside Vigo Harbour could be considered static since we are creating a separated resource for each past year. The available information has been published in a [dataset](#) inside FIWARE CKAN instance.
21.3.4 Vigo Points of interest Information

21.3.4.1 Description
Information about Vigo’s main points of interest including many categories like: Public administration resources, libraries, education, sports and many more. There are more than 1000 points of interest listed in Vigo’s area of influence.

21.3.4.2 Relevance
Having an easy access to all the points of interest listed by the Vigo Council will provide visitors and citizens to relevant information improving their knowledge of the city and services.

21.3.4.3 Data Location
The information about each point of interest is published inside Vigo Council portal - guía cidadá 2.0.

Integration architecture schema
21.3.4.4 Developed components

Static data:

- A Python script was developed that uses web scraping techniques to extract the information for each Point of Interest and transforms it into a CSV file that has been uploaded to the CKAN public instance in FIWARE.

21.3.4.5 Final Results

The information regarding Vigo points of interest has been published in a dataset inside FIWARE CKAN instance.

21.3.5 Vigo Beaches Information

21.3.5.1 Description

Information about Vigo’s beaches including location, access information, and latest water quality analysis if available. There are 46 beaches in Vigo’s area of influence.

21.3.5.2 Relevance

Vigo sits on the Atlantic coast offering its citizens and visitors many beaches for recreational purposes during the summer. Vigo will have 9 blue flag beaches in 2014.

From the citizen or visitor point of view having detailed information about the available beaches will help having a more satisfying experience during its visit and thus improving Vigo tourism outreach in the long term.

21.3.5.3 Data Location

The information about each beach is published inside Vigo Council portal - city information section. The information regarding water quality analysis can be found in the Náyade initiative from the Spanish Ministry of Health. Also information about the weather conditions on the beaches in Vigo area can be found here.
21.3.5.4 Developed components

Static data:

- Two Ruby scripts were developed that use web scraping techniques to extract the information for each beach in Vigo and Water quality information respectively. Both scripts clean and transform the data into CSV files that has been uploaded to the CKAN public instance in FIWARE.

Dynamic data:

- A Python script was built that interacts with the geoRSS service provided by Meteogalicia and retrieves the weather forecast for all the beaches inside Vigo area.
- The script cleans and transforms that information into a suitable NGSI payload including geolocation metadata and sends the result to the FIWARE Orion ContextBroker public instance.
- In order to facilitate the information mashup, the ids for the real-time data are linked to the ids of each beach inside the CKAN public instance.
The retrieved information sent to the Orion Context Broker is structured as follows:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context element ID</td>
<td>Context element type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vigo:Playa:nn</td>
<td>Vigo:Playas</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(where nn is the idMeteoGalicia attribute at CKAN [dataset])</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name (nombre)</td>
<td>Beach name</td>
<td>location (posición)</td>
<td>WGS84 coordinates</td>
</tr>
<tr>
<td>water temperature (temp_agua)</td>
<td>water temperature in ºC</td>
<td>min temperature (temp_min)</td>
<td>minimum temperature in ºC</td>
</tr>
<tr>
<td>Other attributes</td>
<td>Custom attributes as defined by the meteogalicia beach geoRSS service [documentation]</td>
<td>max temperature (temp_max)</td>
<td>maximum temperature in ºC</td>
</tr>
<tr>
<td>timestamp (fecha_creación)</td>
<td>timestamp when the prediction was performed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The script has been scheduled to run daily at 8:00 UTC+1h.

21.3.5.5 **Final Results**

The information regarding each beach could be considered static since it is not updated regularly and has been made available in a [dataset] inside FIWARE CKAN instance.
On the other hand the information about the weather forecast can change on a daily basis and has been treated as a real-time dataset suitable for integration inside the public instance of Orion Context Broker. To check the results follow the instructions provided by the Orion ContextBroker documentation.

21.3.6 Vigo Pharmacies Information

21.3.6.1 Description
Information about Vigo’s pharmacies including location and opening hours.

21.3.6.2 Relevance
From the citizen or visitor point of view having an easy access to the location and opening hours of pharmacies inside Vigo area will be very useful since drugstores have an important role in emergencies handling protocols and citizens comfort in general.

There are currently more than 140 pharmacies in Vigo.

21.3.6.3 Data Location
The original information is published in the “Colegio de farmacéuticos de Pontevedra” website.

Integration architecture schema
21.3.6.4 Developed components

Static data:

- A Ruby script was developed that uses web scraping techniques to extract the information for each pharmacy and transforms it into a CSV file that has been uploaded to the CKAN public instance in FIWARE.
- The information about duty pharmacies is available for the current year so we have also scraped that information and linked it with the pharmacy inventory as another resource.

21.3.6.5 Final Results

The information regarding pharmacies inside Vigo area has been published in a dataset inside FIWARE CKAN instance.

There are two resources linked to the dataset: Pharmacies inventory and Duty pharmacies for 2014.
21.3.7  Vigo Gas Stations Information

21.3.7.1  Description
Information about Vigo’s gas stations including location and latest gas price information. There are 24 gas stations in Vigo’s area of influence.

21.3.7.2  Relevance
Vigo has 24 gas stations where citizens can refuel their vehicles. The Spanish Ministry of Industry provides information about Vigo gas stations including the last available prices for each type of gasoline.

From the citizen or visitor point of view having detailed information about the available gas stations will help them easily locate the gas stations and even save some money trying to go to the cheapest ones when they need to refuel.

21.3.7.3  Data Location
The information about each gas station location along with descriptive data for each gas station is provided by the Spanish Ministry of Industry dedicated portal - geoportalgasolineras.es. Inside that same portal the latest gas prices are published once each gas station communicates them.

21.3.7.4  Integration architecture schema

21.3.7.5  Developed components
Static data:
- A Ruby script was developed that uses web scraping techniques to extract the information for each gas station in Vigo and transforms it into a CSV file that has been uploaded to the CKAN public instance in FIWARE.

**Dynamic data:**

- A Python script was built that retrieves a list of Vigo’s gas stations from FIWARE CKAN instance and then connects to the geoportalgasolineras.es website to retrieve the updated gas price information for each gas station.
- After that the script cleans and transforms that information into a suitable NGSI payload including geolocation metadata and sends the result to the FIWARE Orion ContextBroker public instance.

- The retrieved information sent to the Orion Context Broker is structured as follows:

<table>
<thead>
<tr>
<th>Context element ID</th>
<th>Context element type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vigo:Gasolinera:nn</td>
<td>Vigo:Gasolineras</td>
</tr>
<tr>
<td>(where nn is the id attribute inside the CKAN dataset)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gas prices (precios)</td>
<td>Compound attribute with the price (EUR)  for each available gas type</td>
<td>reference_date (fecha_referencia)</td>
<td>gas price information reference date</td>
</tr>
<tr>
<td>location (posición)</td>
<td>WGS84 coordinates</td>
<td>address (dirección)</td>
<td>gas station address</td>
</tr>
</tbody>
</table>

- The script has been scheduled to run daily at 8:00 UTC+1h.

21.3.7.6 **Final Results**

The information regarding each gas station could be considered static since it is not updated regularly and has been made available in a [dataset](#) inside FIWARE CKAN instance.
On the other hand the information about the gas prices can change on a daily basis and has been treated as a real-time dataset suitable for integration inside the public instance of Orion Context Broker. To check the results follow the instructions provided by the Orion ContextBroker documentation.

21.3.8 Vigo Weather Information

21.3.8.1 Description

Weather information in the Vigo area, including seawater measurements and air quality information. There is one air quality sensing station located in the city and four meteorological stations in Vigo:

- Vigo harbour
- Vigo University Campus
- Galician Atlantic Islands Maritime-Terrestrial National Park
- Vigo city center

21.3.8.2 Relevance

Weather information is crucial when planning a visit to Vigo, it can be useful to decide if we can plan an outdoor activity or an indoor one. Historical weather conditions can also be useful when planning a visit on the long term or to analyze outliers in Vigo’s climate.

Air quality impacts heavily on the quality of life of citizens and visitors of Vigo. Having that information available inside the FIWARE platform will allow developers to either create dashboards that quickly inform a user about the air quality in a glance or make analysis to see how the air quality is evolving in time.
21.3.8.3 **Data Location**

The information for Galicia Region is provided by [Meteogalicia](http://www.meteogalicia.es). The four weather stations available for the Vigo area are identified by the following Meteogalicia IDs:

- Vigo harbour: 14001
- Vigo University Campus: 10161
- Galician Atlantic Islands Maritime-Terrestrial National Park: 10125
- Vigo city center: 10142

Air quality information is published [here](http://www.meteogalicia.es), and the data for seawater measurements is retrieved from [here](http://www.meteogalicia.es).

Integration architecture schema

---

21.3.8.4 **Developed components**

The scripts were developed using [Pentaho Kettle](http://www.pentaho.com) tools, and are run on an Ubuntu instance every 15 minutes via a cron job.

**Dynamic data:**

- A kettle script was developed to retrieve information from each of the weather stations available. Two additional kettle scripts were developed for air quality information and seawater data.
- Each script retrieves information in XML format from the Meteogalicia service and selects the relevant values, discarding irrelevant information.
- Then the scripts transform the information into a suitable NGSI format and store it in a temporary file.
- Finally, this temporary file is read and its contents are published via the Orion Context Broker located in [http://orion.lab.fi-ware.org:1026](http://orion.lab.fi-ware.org:1026)

The retrieved information sent to the Orion Context Broker is structured as follows:
### Vigo Harbor weather

<table>
<thead>
<tr>
<th>Context element ID</th>
<th>Context element type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vigo:Porto</td>
<td>WeatherStatus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Update time of the current element</strong></td>
<td>lastupdated</td>
<td>timestamp</td>
<td>Wind speed</td>
<td>Velocidade do Vento</td>
<td>m/s</td>
</tr>
<tr>
<td><strong>Night sky luminosity</strong></td>
<td>Escuridade do Céu Nocturno</td>
<td>mag/arcsec2</td>
<td>Wind direction</td>
<td>Velocidade do Vento</td>
<td>°</td>
</tr>
<tr>
<td><strong>Average temperature</strong></td>
<td>Temperatura media</td>
<td>°C</td>
<td>Average relative humidity</td>
<td>Humidade relativa media* type</td>
<td>%</td>
</tr>
<tr>
<td><strong>Global Sun radiation</strong></td>
<td>Radiación Solar Global</td>
<td>W/m2</td>
<td>Rain</td>
<td>Chuvia</td>
<td>L/m2</td>
</tr>
<tr>
<td><strong>Barometric pressure</strong></td>
<td>Presión Barométrica</td>
<td>hPa</td>
<td>Wind gust</td>
<td>Refacho</td>
<td>m/s</td>
</tr>
<tr>
<td><strong>Sun hours</strong></td>
<td>Horas de Sol</td>
<td>h</td>
<td>Wind gust direction</td>
<td>Dirección do Refacho</td>
<td>°</td>
</tr>
<tr>
<td><strong>Dew temperature</strong></td>
<td>Temperatura de Orballo</td>
<td>°C</td>
<td>Sea level reduced pressure</td>
<td>Presión reducida ao nível do mar</td>
<td>hPa</td>
</tr>
</tbody>
</table>

### Vigo University weather

<table>
<thead>
<tr>
<th>Context element ID</th>
<th>Context element type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vigo:Campus</td>
<td>WeatherStatus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
</table>

---

D.10.7 Smart Cities connection to FIWARE Lab
### Update time of the current element
- **lastupdated**
- **timestamp**
- **Wind speed**: Velocidade do Vento (m/s)

### Rain
- **Chuvia**
- **mag/arsec**
- **Wind direction**: Velocidade do Vento (°)

### Average temperature
- **Temperatura media**: °C
- **Average relative humidity**: Humidade relativa media type (%)

### Global Sun radiation
- **Radiación Solar Global**
- **Solar W/m2**: Diffuse Sun radiation
- **W/m2**: Radiación Solar Difus

### Direct Sun radiation
- **Radiación Directa**
- **Solar W/m2**: Reflected Sun radiation
- **W/m2**: Radiación Solar Reflectida

### Sun elevation
- **Elevación Solar**
- **°**: Sun orientation
- **°**: Orientación Sola

### Barometric pressure
- **Presión Barométrica**
- **hPa**: Wind gust
- **m/s**: Refacho

### Sun hours
- **Horas de Sol**
- **h**: Wind gust direction
- **°**: Dirección do Refacho

### Dew temperature
- **Temperatura de Orbalo**
- **°C**: Sea level reduced pressure
- **hPa**: Presión reducida ao nivel do mar

---

**Galician Atlantic Islands Maritime-Terrestrial National Park weather**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update time of the current element</td>
<td>lastupdated</td>
<td>timestamp</td>
<td>Wind speed</td>
<td>Velocidade do Vento</td>
<td>m/s</td>
</tr>
<tr>
<td>Night sky luminosity</td>
<td>Escuridnade do Céo Nocturno</td>
<td>mag/arsec2</td>
<td>Wind direction</td>
<td>Velocidade do Vento</td>
<td>°</td>
</tr>
<tr>
<td>Average</td>
<td>Temperatura de Orbullo</td>
<td>°C</td>
<td>Average relative humidity</td>
<td>Humidade relativa</td>
<td>%</td>
</tr>
</tbody>
</table>
### Temperature

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Media</th>
<th>Humidity</th>
<th>Media Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Sun radiation</td>
<td>Radiación Solar Global</td>
<td>W/m²</td>
<td>Rain</td>
</tr>
<tr>
<td>Barometric pressure</td>
<td>Presión Barométrica</td>
<td>hPa</td>
<td>Wind Gust</td>
</tr>
<tr>
<td>Sun hours</td>
<td>Horas de Sol</td>
<td>h</td>
<td>Wind Gust Direction</td>
</tr>
<tr>
<td>Dew temperature</td>
<td>Temperatura de Orballo</td>
<td>°C</td>
<td>Sea Level Reduced Pressure</td>
</tr>
</tbody>
</table>

### Vigo City Center Weather

<table>
<thead>
<tr>
<th>Context Element ID</th>
<th>Context Element Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vigo</td>
<td>WeatherStatus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
<th>Update Time of the Current Element</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Updated</td>
<td>timestamp</td>
<td></td>
<td>Rain</td>
<td>Chuvia</td>
<td>L/m²</td>
</tr>
<tr>
<td>Average Temperature</td>
<td>Temperatura media</td>
<td>°C</td>
<td>Average Relative Humidity</td>
<td>Humidade media Type</td>
<td>%</td>
</tr>
<tr>
<td>Dew Temperature</td>
<td>Temperatura de Orballo</td>
<td>°C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Air Quality Information

<table>
<thead>
<tr>
<th>Context Element ID</th>
<th>Context Element Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vigo</td>
<td>AirQuality</td>
</tr>
<tr>
<td>Attribute</td>
<td>Name</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Update time of the current element</td>
<td>lastupdated</td>
</tr>
<tr>
<td>Chemiluminiscence</td>
<td>Quimioluminisce ncia</td>
</tr>
<tr>
<td>Chemiluminiscence</td>
<td>Quimioluminisce ncia</td>
</tr>
<tr>
<td>Beta absorption</td>
<td>Absorción beta</td>
</tr>
<tr>
<td>Position of the station</td>
<td>position</td>
</tr>
</tbody>
</table>

Seawater measurements

<table>
<thead>
<tr>
<th>Context element ID</th>
<th>Context element type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vigo:Cies</td>
<td>SeaStatus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update time of the current element</td>
<td>lastupdated</td>
<td>timestamp</td>
<td>Dew temperature</td>
<td>Temperatura de Orbollo</td>
<td>ºC</td>
</tr>
<tr>
<td>Average temperature</td>
<td>Temperatura media</td>
<td>ºC</td>
<td>Average relative humidity</td>
<td>Humidade relativa media” type</td>
<td>%</td>
</tr>
<tr>
<td>Level 1 salinity</td>
<td>Salinidade Nivel 1</td>
<td></td>
<td>Level 1 water temperature</td>
<td>Temperatura Auga Nivel 1</td>
<td>ºC</td>
</tr>
<tr>
<td>Level 1 density anomaly</td>
<td>Anomalia da Densidade Nivel 1</td>
<td>kg/m3</td>
<td>Level 1 conductivity</td>
<td>Conductividad Nivel 1</td>
<td>mS/cm</td>
</tr>
</tbody>
</table>
21.3.8.5 **Final Results**

The information about the weather measurements changes frequently depending on the station and has been treated as a real-time dataset suitable for integration inside the public instance of Orion Context Broker. To check the results follow the instructions provided by the Orion ContextBroker documentation.

An example of the resulting data for one of the kettle scripts is shown on the right:

**Vigo RSS information**

21.3.8.6 **Description**

Three RSS sources of news and events for the city of Vigo are used as data sources to be included in the FIWARE system, namely:

- **Vigo traffic information**: List of the traffic situation warnings and advices to inform the citizen about possible traffic jams due to some maintenance on the streets.

- **Vigo cultural events**: List of the main cultural events that take place in Vigo around the current date: festivals, expositions, performances, shows, theatre plays...

- **Vigo news**: Information about Vigo’s recent events and other alerts.

21.3.8.7 **Relevance**

Traffic information is provided by the Vigo Council and allows citizens to be informed of the traffic related events such as closed roads, public work executions...

Cultural events information may be useful for citizens and tourists that look for entertainment. The knowledge of schedule of the current events can help the users when they try to plan a visit or a spare day.
Vigo news is provided by the Vigo Council and it allows citizens to be informed of the most up-to-date information about what is going on in Vigo City.

### 21.3.8.8 Data Location

The information for cultural events is accessible online as a website at [http://hoxe.vigo.org/actualidade/axenda.php](http://hoxe.vigo.org/actualidade/axenda.php) and as a RSS format at [http://hoxe.vigo.org/rss/rss_axenda.php](http://hoxe.vigo.org/rss/rss_axenda.php). Information is offered by Concello de Vigo. The whole month is accessible at the web but information is only complete for a couple of days in advance. There is a RSS feed for each single day.


### Integration architecture schema

![Integration architecture schema](image)

#### 21.3.8.9 Developed components
The scripts were developed using [Pentaho Kettle](http://pentaho.org) tools, and are run on an Ubuntu instance every 15 minutes via a cron job.

**Dynamic data:**
- Three kettle scripts were developed to retrieve information from traffic, cultural events and city events.

- Each script retrieves information in XML format from the RSS service and selects the relevant values, discarding irrelevant information.

- Then the scripts transform the information into a suitable NGSI format and store it in a temporary file.

- Finally, this temporary file is read and its contents are published via the Orion Context Broker located in http://orion.lab.fiware.org:1026

The retrieved information sent to the Orion Context Broker is structured as follows:

<table>
<thead>
<tr>
<th>Context element IDs</th>
<th>Context element types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vigo:RSS</td>
<td>News</td>
</tr>
<tr>
<td>Vigo:Trafic</td>
<td>Event</td>
</tr>
<tr>
<td>Vigo:Cultural</td>
<td>Event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Publication title</td>
<td>DescriptionText</td>
<td>Publication description</td>
</tr>
<tr>
<td>Link</td>
<td>Publication reference link</td>
<td>PubDate</td>
<td>Publication date</td>
</tr>
</tbody>
</table>

21.3.8.10 **Final Results**

The information described in this section has been treated as a real-time dataset suitable for integration inside the public instance of Orion Context Broker. To check the results follow the instructions provided by the Orion ContextBroker documentation.
21.4 Test and Validation

A simple demonstration WireCloud application was developed in order to validate the information retrieval process and to show a possible use case for the gathered data.

21.4.1.1 Application description

As Vigo is settled right at the coast, we have developed a simple WireCloud application to show the potential of the data about beaches we have integrated on the FIWARE platform.

The use case: A tourist is planning to go swimming and wants to find a nice beach not too far from the places he is going to visit tomorrow.

In the application, the MAP VIEW shows the user the location of each beach on a map. When the user selects any of the beaches, the main information about it is shown in the WEB INFORMATION box (buses lines, size, handicapped access, etc).

At the same time the user can check real-time predictions about tomorrow’s weather conditions on each beach. The two temperature prediction graphs show the maximum and the minimum water temperature expected. Moreover, the CKAN DATA, offers the users data about the water quality.

In our use case, the tourist can select a beach near his hotel or the restaurant he has reserved for lunch, check the water quality or if it is going to be too cold and also get the bus lines information about how to get there.
21.4.1.2 **Application architecture overview**

Each beach location is retrieved from a CKAN source ([https://data.lab.fiware.org/dataset/vigo-playas/resource/b3061e34-7a83-4c67-8864-f3918acd7714](https://data.lab.fiware.org/dataset/vigo-playas/resource/b3061e34-7a83-4c67-8864-f3918acd7714)) and transformed into a POI to show it on the map. When the user selects a concrete POI, the information URL is extracted from it and passed as a parameter to the web information widget.

The water quality information is retrieved from another CKAN source ([https://data.lab.fiware.org/dataset/vigo-playas/resource/0c4ab487-18cf-4549-b11b-eb65b8c16339](https://data.lab.fiware.org/dataset/vigo-playas/resource/0c4ab487-18cf-4549-b11b-eb65b8c16339)) and directly show in a Data View Table widget.

Temperature predictions are obtained from a NGSI source integrated with the FIWARE ORION Context Borker ([http://orion.lab.fiware.org:1026/](http://orion.lab.fiware.org:1026/)). Entities are received periodically and used to feed two graphical widgets.
New widget/operator implementations

Two new elements were developed to implement the demonstration application:

1. **EntityToAttribute Operator**: This operator allows to extract a single attribute from a given entity.
2. **Multibar Viewer Widget**: A graphical tool to draw three types of graphs: line graphs, pie charts and multi-bar graphs.

Both elements are fully configurable from the Settings tab of the Configuration panel, so they can be easily reusable.