"Linked Open Apps Ecosystem to open up innovation in smart cities"

Project Number: 297363

Deliverable:	**iCity Platform Prototype final**
Version:	**2.4**
Delivery date:	**10/08/2015**
Dissemination level:	**PU**
Author:	**RETEVISION**

Summary: The aim of this document is to provide a global vision of the iCity platform. iCity is the result of the integration of many management solutions in one solution, defining the architecture, it's policies and how to manage information sources offered by cities. This deliverable also reflects the status of iCity platform through fourth year of project and the status of the different modules that compose the iCity platform. Those modules delivered by WP4 at M42 (end of June 2015) are included in this document.

DOCUMENT HISTORY

Version	Date of issue	Status	Content and changes	Modified by
0.1	29/05/2015	Draft	Creation	RETE
0.2	10/07/2015	Draft	Updated sections: 4.2.1, 4.2.4 and 7	FRAUNHOFER
0.3	13/07/2015	Draft	Updated section: 4.2.2	Fina Sala
0.4	13/07/2015	Deliverable	Integration.	RETE

DOCUMENT CONTRIBUTORS

Partner	Contributor
Retevision	Alex Sala, Celia Barca, Ariadna Vázquez
Fraunhofer	Yury Glikman
Barcelona	Fina Sala, Marc Garriga

TABLE OF CONTENTS

TABLE OF FIGURES

ABBREVIATION AND ACRONYMS

Acronym	Description
SDK	Software Development Kit
API	Application Programming Interface
REST	Representational State Transfer
ISP	Internet Service Provider
VM	Virtual Machine
MS	Management System
SOAP	Simple Object Access Protocol
SLA	Service Level Agreement
DWH, DW	Data Warehouse
BI	Business Intelligence
WP	Work Package
SME	Small Medium Enterprises
ETL	Extract, Transform, Load
EDW	Enterprise Data Warehouse
ODS	Operational Data Store
DSS	Decision Support System
RBDMS	Relational Database Management System
DDD	Domain Driven Design
O&M	Observation and Measurements
SAS	Sensor Alert Service
SOS	Sensor Observation Service
TDD	Test Driven Development

1. Preface

1.1 Introduction, Why an only document?

At the end of the second year, WP4 created a document with the aim of provide a global vision of the iCity platform and its whole status at the end of the second year of the project (which was the evolution of the first platform prototype provided at M24) and the mapping of the five prototypes D4.6, D4.7, D4.8, D4.9 and D4.10 with the different modules that compose the iCity platform.

These five prototypes have been delivered by WP4 at M24 and are also reported inside this document. They can be found in the following sections of the document:

- **D4.6 "System Management Adaptation-rev**" is enclosed in **chapter 5**.
- **D4.7 "System Operation Adaptation-rev"** is enclosed in **chapter 6**.
- **D4.8 "Data warehouse & business intelligence-rev"** in enclosed in **chapter 7**.
- **D4.9 "Open Data-rev"** is enclosed in **chapter 8**.
- **D4.10 "SDKs-rev"** is enclosed in **chapter 9**.

During the third year of the project, WP4 has been focused on the evolution of the platform also integrating the new Information Systems while enhancing the iCity API and developing new features for city managers and Information Systems' owners.

WP4 keeps working to improve the platform. Increasing number of functionalities and engaging new developers, Information Systems managers and cities to participate in the iCity project.

The tasks that WP4 has been working during the third year are described below:

- **T4.1 "Profile & Requirement Definition for each iCity case and third parts"**, enclosed in **chapters 3 and 6**.
- **T4.2 "System Management Adaptation"**, enclosed in **chapter 5**.

- **T4.3 “System Operation Adaptation”**, enclosed in **chapter 6**.
- **T4.5 “Open Data”**, enclosed in **chapter 8**.
- **T4.6 “Network Interface”**, enclosed in **chapter 3**.
- **T4.9 “Data collection/aggregation from southbound layers”**, enclosed in **chapter 4**.

WP4 has also developed the Partner’s Proposal Portal, in order to collect, classify and manage all the suggestions related to improve iCity platform.

In addition, the following documents have been delivered during this period:

- D1.7 Project Management Report.
- D5.5 Pilots Development Report-v2.
- D3.8 Open Urban Services Delivery Platform Architecture Blueprint-Final.
- D8.5 Dissemination Tools: e-Newsletter-3.
- D8.6 Dissemination Tools: e-Newsletter-4.

This deliverable seeks to report the iCity platform and systems. Aiming to improve platform understanding and usability including work packages and activities of the iCity project.

2. Introduction

2.1 Purpose

This document offers a global perspective of the platform, with aim to offer associated details to stakeholders. Reflecting the platform status, developments and improved functions introduced during the forth year of the project.

2.2 What's the iCity Platform?

Conceptually, iCity platform is an information systems manager that, by using the appropriate interfaces, allows the group of users and developers make use of the information systems in an easy and unified way.

The platform provides information and application services framework into which content through the local information systems sources are added and made available through the use of standard API's.

The 'Open' platform is regulated through a separate legal agreement among the user/developers and the data/Information System provider. The host city accepts no technical or legal liability or responsibility between the partners.

Main iCity platform features:

- Provides an additional security layer.
- It doesn't require changes at City side.
- Minimizes changes in city management and provides a Single Sign On.
- Provides a consistent API for developers.
- Provides a collaboration platform driving innovation.

3. Overview of iCity Platform

3.1 iCity platform's General Description

A vision of the iCity platform's architecture can be overviewed and represented as a set of components which interact with a central element, being the core of the iCity platform. Conceptually this is shown in the figure below:

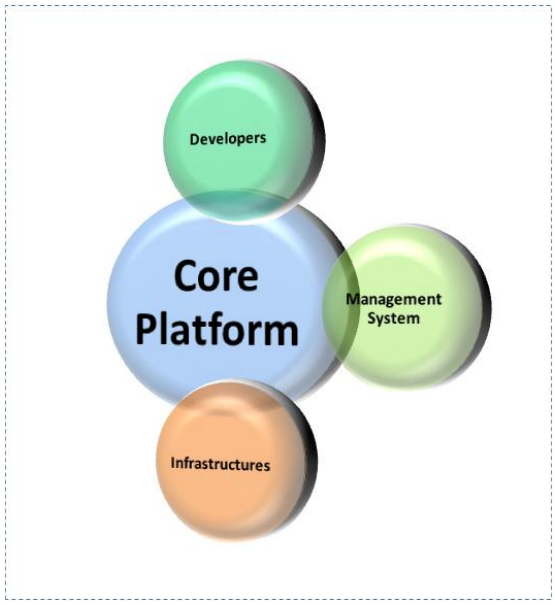

Figure 1 iCity Platform components

Using this concept as the starting reference, the following points will detail each one of these components. This structure or top level model makes easy to the reader to focus in the component in which he's most interested in.

The structure or model will enable the reader to understand for each defined part:

- Global definition and associated architecture.
- The interaction points with the other areas.
- Definition of the Access interfaces.
- Environment of a user's profile.

- Current implementation stage.

3.1.1 Core's Platform Environment

3.1.1.1 *Definition*

The iCity platform core acts as a hub, connecting the city Information System with the software developers who require access to the data and Information Systems. The iCity platform also reports its status to a third area where the monitoring functions are performed, enabling access for developers

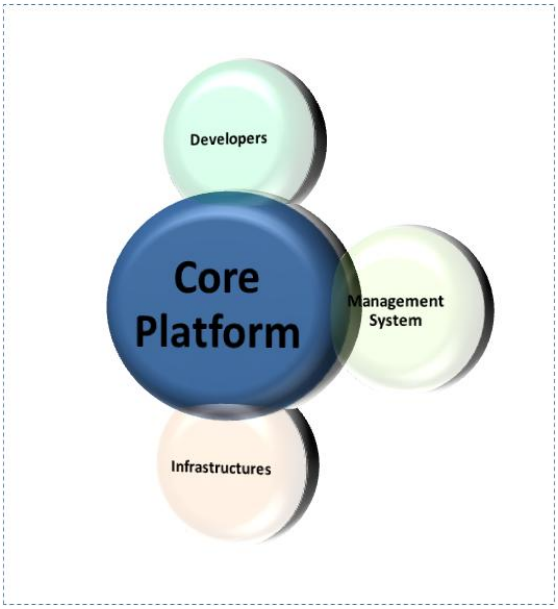

Figure 2 Core Platform Environment

3.1.1.2 *Dependencies – Relationships*

The Central element of the architecture interacts with the other remaining environments such as Developers, Management System and Information Systems of the platform.

3.1.1.3 *Access interfaces*

As mentioned, the Central Core interacts with the rest of the platform areas. The existing interfaces to communicate with those environments are:

- With Developers Environment:
  - The central system provides to developers a set of services and libraries that permits access to the information and also to the Information Systems management;
  - Initially developers and users have to access and register in the iCity platform via an interface called “Portal”. This will enable access and allow interacting with the platform.
  - There‘s an interface to access to the Open Data libraries and Open Data portal. In this interface user can find all the information related to Open Data offered by involved cities in the project.
- With Information System Environment:
  - Information Systems: Once an Information System is opened to iCity Platform by a city, the communication between both objects can be integrated with the central system through:
    - A set of libraries, in case of integration starts from the Information System to the Core.
    - An interface which gives users access to data provided by the information systems belonging to iCity.
  - These two types of defined interface provide global coverage to any kind of Information System that should be integrated in the platform. Each Information System acts like a data source that might require its own specific version (using one of the two models of integration described above), whilst some of them can be reused in the future.
- Interfaces with Management System Environment:
  - The Management System interacts with different components of the central element directly through the Operation System, in order to know the status of the processes that conform the platform.

- In order to connect, manage and administrate each component of the platform and ensure the proper operation mode. The Management System collects information from the components by using an own proprietary process called SAS. Acting like an element administrator, informing about the events, and alarms occurred in the platform.

- This area is only accessible for administrator users such as Information Systems managers from each city and iCity Platform administrators.

3.1.2 Developers' Environment

3.1.2.1 *Definition*

The developers' environment will allow developers to access the platform.

The aim of this environment is to open the platform by using multiple methods and languages in order to engage the largest number of developers interested in iCity project.

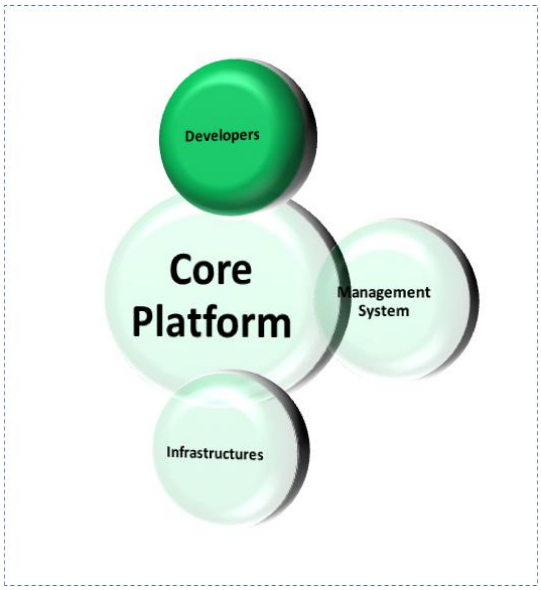

Figure 3 Developer's Environment

3.1.2.2 *Dependencies – Relationships*

The developers' environment interacts and operates only with the Core Platform.

3.1.2.3 *Access Interfaces*

Developers can access to the iCity managing data and also to the platform by simply registering into the iCity Developer's Portal. Once registered in the Developer Portal the data of the Information Systems can be accessed and used to develop apps. Public portal can be reached by accessing www.icityproject.eu.

Figure 4 iCity Public Portal

- Offers public information related to iCity Project.
- Links with the iCity Developers Portal and iCity Open Data Portal.

- Private portal (available at http://icity-devp.icityproject.com):

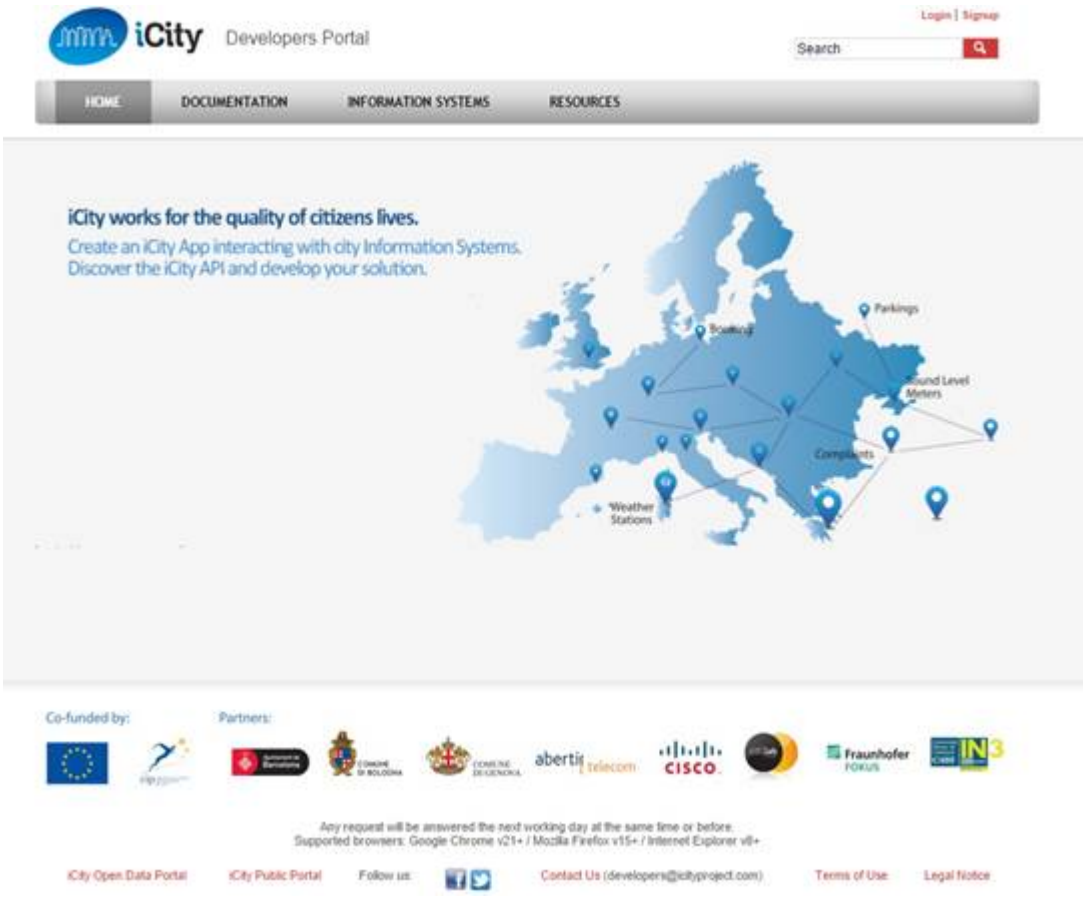

Figure 5 iCity Developer's Portal

- Allows the registration of developers.
- Guarantees secure access
- Manages workflows

- **API REST:** accessible through the iCity Developers Portal where it's possible to obtain all the information about using the API and main functionalities as well as keeping the information systems' status updated. The main characteristics are:

  - Based on standards
  - It can be chosen the output format (XML, JSON, TXT, etc.) of the response
  - Includes next functions:
    - Obtain information from all components (Information Systems, devices, collections, etc.)

- Obtain information about the latest data recorded and real-time data by including some parameters

In order to guarantee a secure access environment, the following APIs and functionalities are developed to safeguard the platform and are available only by registering into the iCity portal (see Appendix I: iCity API Functionalities)

- **Partner's Proposal Portal**, available at (http://www.api.icityproject.com)

Figure 6 iCity Partner's Proposal Portal

This portal has been developed in order to collect, classify and manage all the suggestions related with the improvement of the iCity platform.

- **Open Data Portal** includes the following functions (http://opendata.icityproject.com/) :

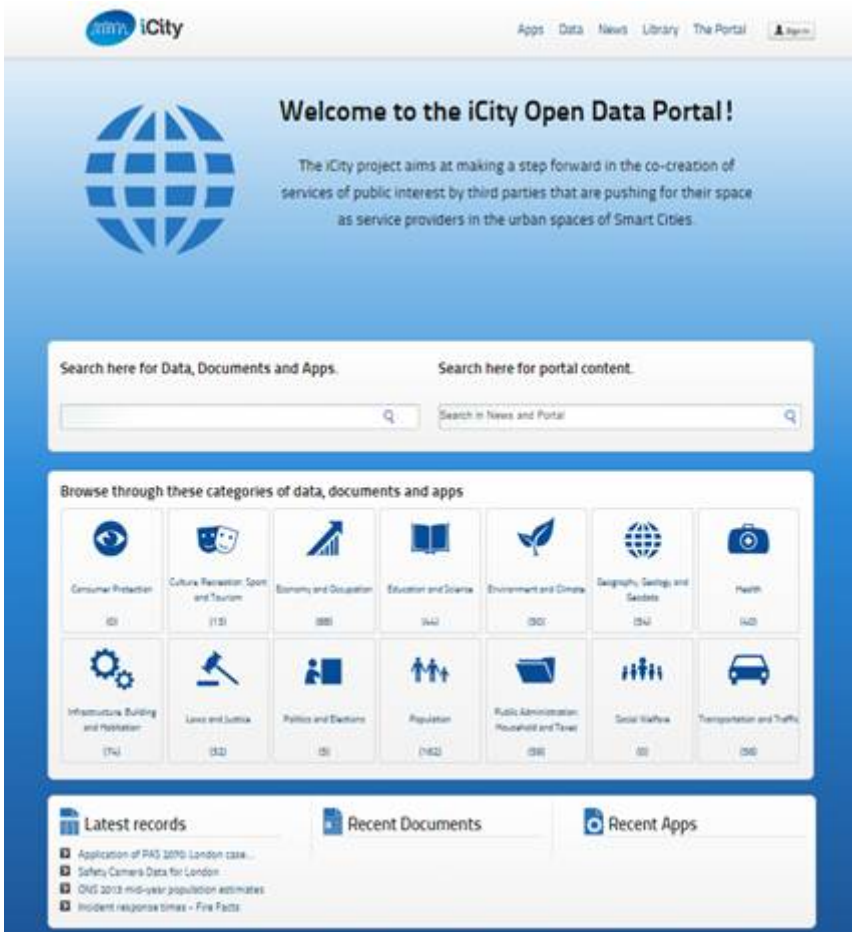

Figure 7 iCity Open Data Portal

- Search for contents
- Catalog of contents
- Publish new apps

3.1.3 Information Systems' Environment

3.1.3.1 *Definition*

The Information Systems environment is where the iCity platform will be hosted on behalf of the cities, where the local 'Open' Information System will be stored, with services provided back to the cities for local geographic dissemination to developers and users.

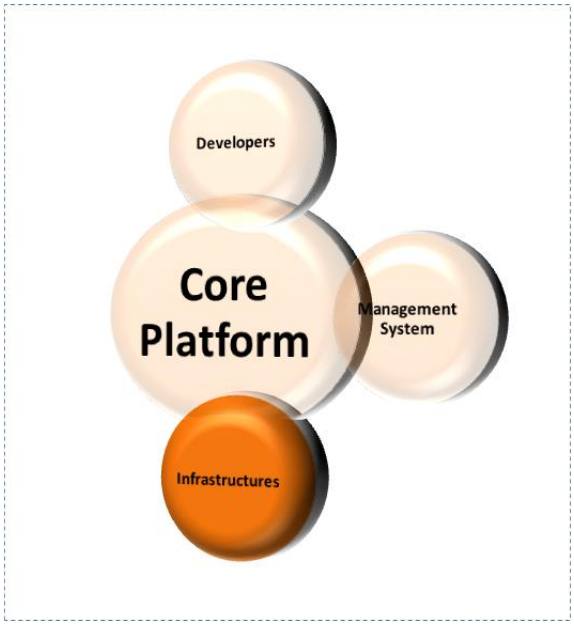

Figure 8 Information System's Environment

3.1.3.2 ***Dependencies – Relationships***

The Information System environment interacts only with the core platform.

3.1.3.3 ***Access Interfaces***

Each of the Information Systems may have different access interfaces or even not have any. In the latter case, the core will provide an input interface to perform the integration called "Universal connector". This new connector will facilitate the integration of new open Information Systems in a standard way.

To enable these Information Systems inside iCity Platform, there is available an administration Portal to manage all the information related to the new connected Information Systems as well as the definition of policies and rules of use. This portal will interact with iCity developers Portal, managing the access to the cities Information Systems'.

3.1.3.4 ***Implementation's State. Roadmap***

Currently, there are 'Open' Information Systems fully integrated and several in progress of integration. Each city provides a number of new 'Open' Information Systems that will be progressively incorporated into the platform.

WP4 aims to add new Information Systems, not only coming from consortium cities but also foreign ones.

From the beginning of iCity project to the end of the third year, the following Information Systems have been integrated with iCity platform:

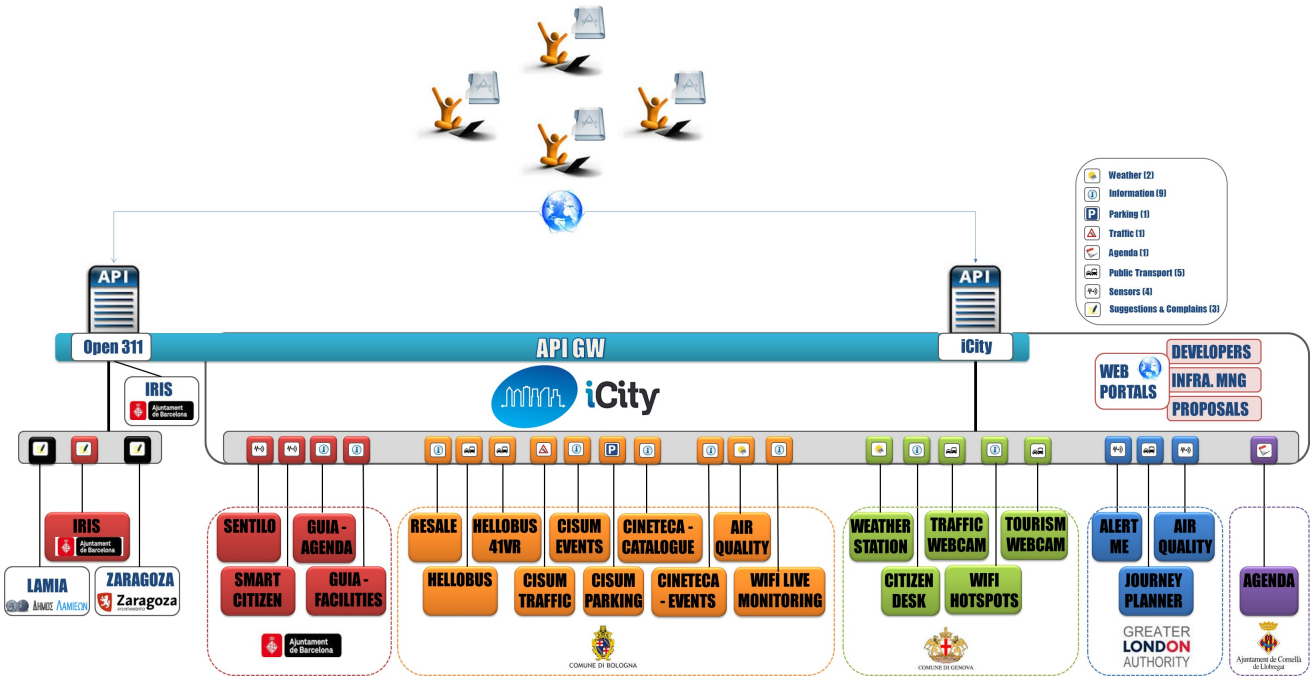

Figure 9 iCity Platform Information Systems

CITY	INFORMATION SYSTEM
Barcelona	Sentilo
	Smart Citizen
	IRIS
	Guia - Agenda
	Guia – Facilities
Bolonia	TPER Query Hello Bus
	TPER Query Hello Bus 4ivr
	TPER Query Resale

	Cisium Events
	Cisium Traffic
	Cisium Parking
	CINETECA - Catalogue of DVDs and VHS
	CINETECA - Events
	Air Quality
	Wifi Location & Live Monitoring
Genova	Weather Station
	Citizen's Desk
	Traffic Webcam System
	Wifi Hotspots
	Tourism Webcams
	Air Sensors
London	Air Quality Sensor
	Transport for London
	Alert Me
Lamia	Suggestions & Complains
Zaragoza	Suggestions & Complains
Cornellà	Agenda
Abertis telecom	Urbiótica Sensors
	Parkare Sensors

In order to continue promoting the co-creation of public interest services, WP4 will integrate the following Information Systems at the end of the project:

CITY	INFORMATION SYSTEM

Barcelona	Wifi Network – MSE Service
Bologna	Geocoding
	Agenda Cultural
Genova	Toponyms
	Hydrometers
	Geographical Information

3.1.4 Management System Environment (Administration)

3.1.4.1 *Definition*

The target of the management system is to supervise, monitor and manage the iCity network as well as the platform, guaranteeing the SLA service.

At this stage of the iCity platform the management system provides the following functions:

- An Administration portal allows the configuration and edition of parameters related to events and alarms for the monitoring module. Also managing the different users and groups of the Management System.
- A Monitoring module checks the different components of the iCity platform like element status, hardware status, network status, services monitoring, threshold alarms and spatial alarms.

3.1.4.2 *Dependencies – Relationship*

The Management System environment only interacts with the Core Platform.

3.1.4.3 *Access Interfaces*

This system has a graphical user interface as follow:

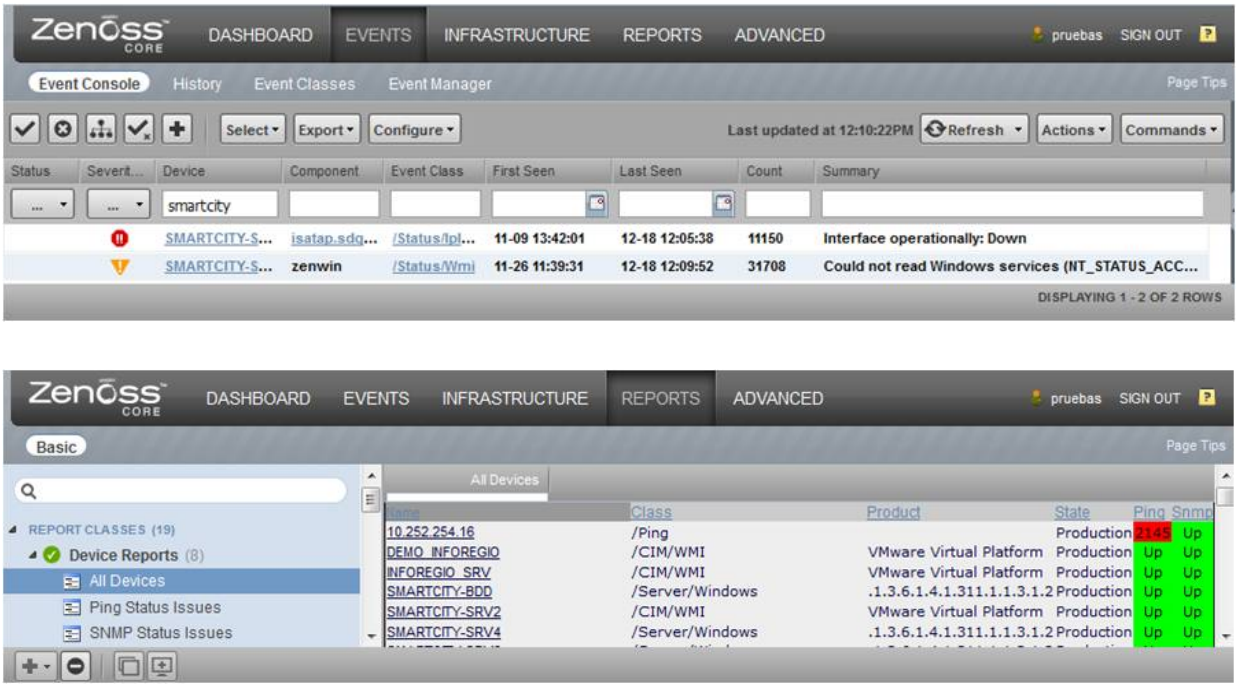

Figure 10 Management System Environment

3.1.4.4 *User's Profile*

The user of this environment can be an administrator user or an operator. The administrator user is responsible for managing and monitoring the whole iCity platform.

Additionally, the system enables access to external users in order to provide customized monitoring service.

4. iCity Platform's Architecture

4.1 Introduction

This section introduces the iCity architecture in an easy way conceived to be understood by non-technical users. The following image depicts how the platform is integrated and its functionalities.

iCity architecture and user interface modules have been redesigned. First synthesizing the old version into a newer one, with an easier, complete and friendlier user interface. Old modules shown on the left are segregated into User, iCity Platform and Net getting a cleaner environment interface as described in past sections.

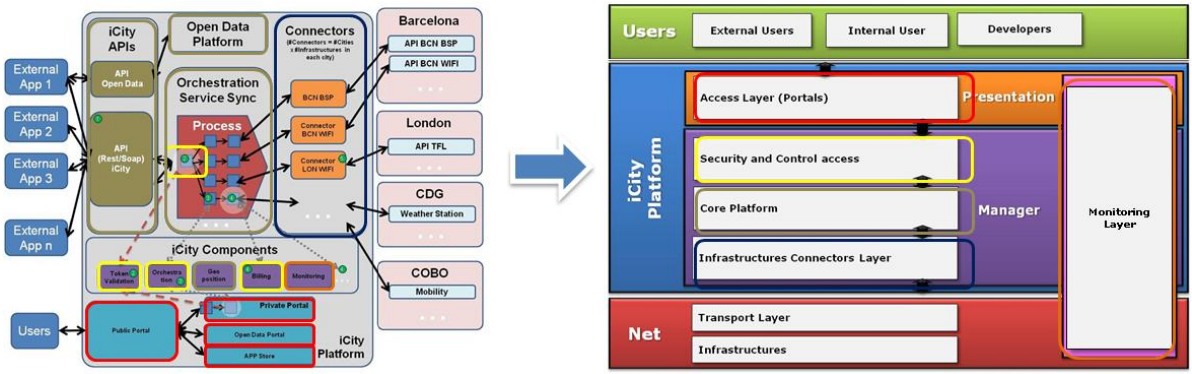

Figure 11 New architecture version

These blocks show how iCity Platform is conformed.

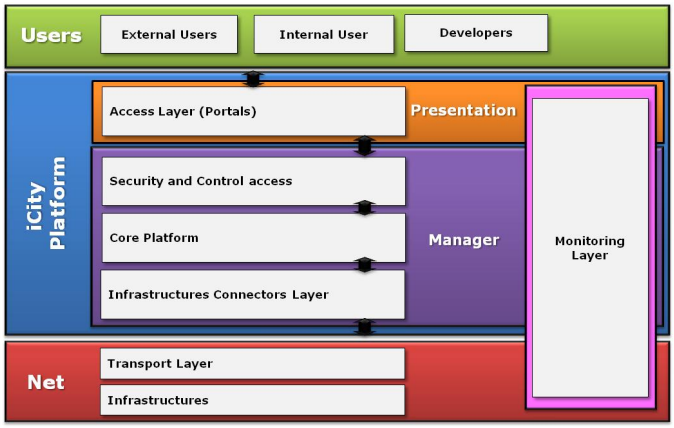

Figure 12 iCity's Architecture Blocks

4.2 Presentation - Access Layer (Portals)

Access layer permits different users to monitor, control and manage any object located into the Information System's layer.

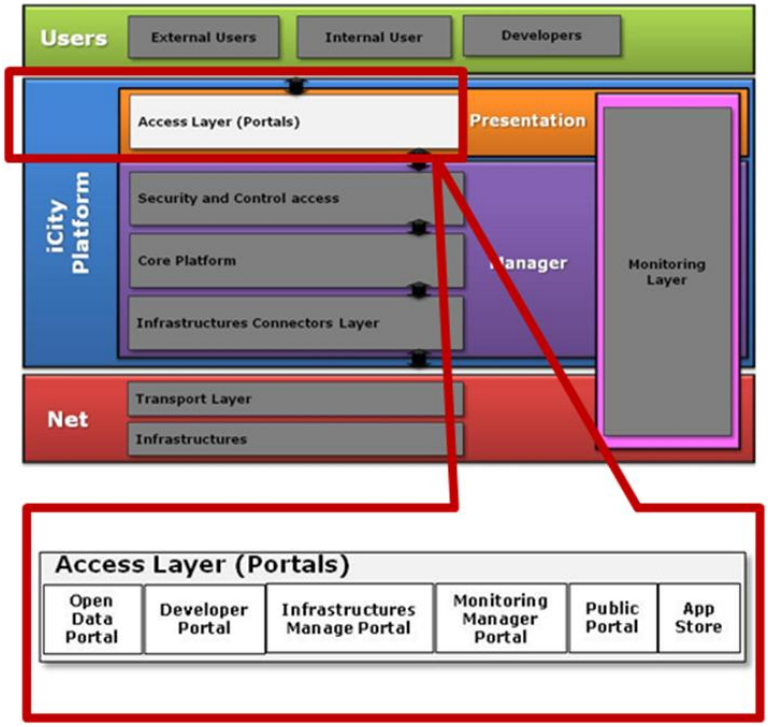

Figure 13 Presentation - Access Layer

- **Open Data Portal:** This portal gives access to the Open data for any developer and explains how to use it.
- **Developers Portal:** iCity Developers Portal provides access to the developers from the registration to the set of services and libraries that allow them accessing to the information and its management.
- **Information Systems Manage Portal:** This environment enables the Information System's manager to add, remove or edit any Information System included in iCity platform.
- **Monitoring Manager Portal:** The aim of monitoring is to provide a portal where users can get specific information about the current usage of the platform.
- **Public Portal:** Public portal offers all the information related to the iCity Project and also includes the links to redirect to the different portals mentioned above.
- **App Store:** The App Store provides a portal to access to all applications developed for the iCity Project. Those apps can also be used by end users.

4.2.1 Open Data Portal

The iCity Open Data Portal[1] is an instance of the Open Data platform developed by Fraunhofer FOKUS in its previous projects, which was customised by the iCity project and integrated with the open data platforms for the cities involved in iCity. The details on that are presented in the section 7 of this document.

The first version of the Fraunhofer Open Data platform was developed by Fraunhofer FOKUS during the course of the EU Open Cities project (http://opencities.net, Grant agreement:

[1] http://opendata.icityproject.com/

270896) and it was published under the AGPL version 3 license. Later, it has been further developed and deployed as the German national governmental data portal GOVDATA.de[2].

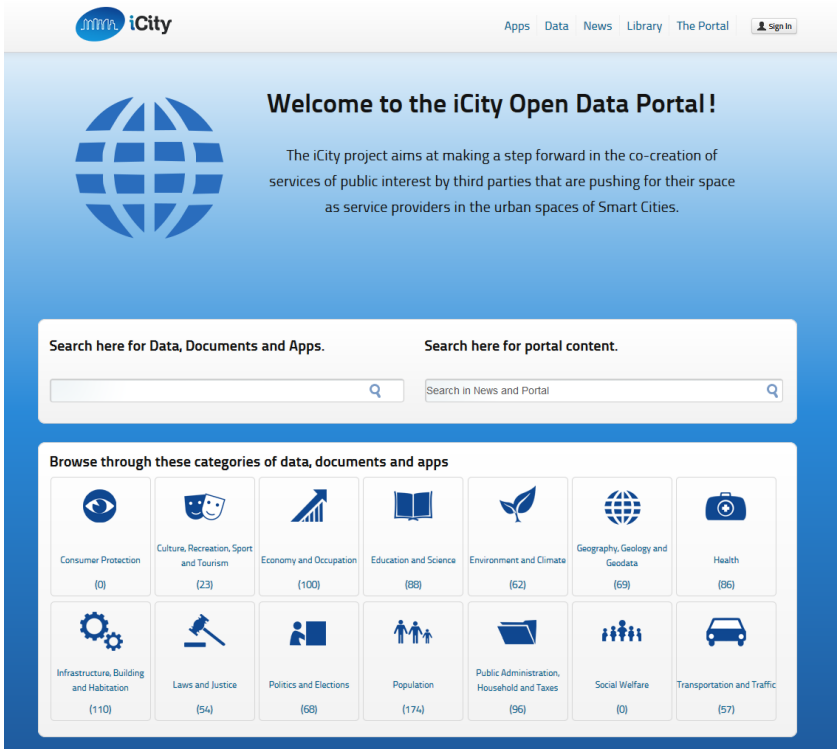

Figure 14 Homepage of the iCity Open Data Portal

The platform offers an integrated solution for publishing open data. It provides a data portal (i.e., the user front-end) and a data registry. Furthermore it offers a Content Management System to maintain static content and a blog-like news section.

The latest version of the platform was developed only with a German user interface, which was extended with the support of English in the iCity project. The primary benefit that the platform offers is a “one-stop-shop” experience enabling the development of novel third-party applications.

[2] https://www.govdata.de/

Application developers have a consolidated view on all of the open data available that has been catalogued, and allow navigating, identifying, accessing and using data of interest. The platform's main functions like viewing and downloading data, documents and apps are accessible without registration, thus for guests. The user can access the data via several channels.

Starting from the homepage he/she can directly access one of the categories (e.g. Education and Science or Health) and browse the associated datasets. On the homepage as well, users can find a search input field to search for keywords within the data catalogue. Another possibility is to navigate to the data catalogue view (or the app view) and browse the datasets from there.

The datasets can be filtered by several filter categories (Categories, Authors, Keywords, Formats and Licences) and can be sorted by relevance, name or update date. In this view as well it is possible to enter a search term.

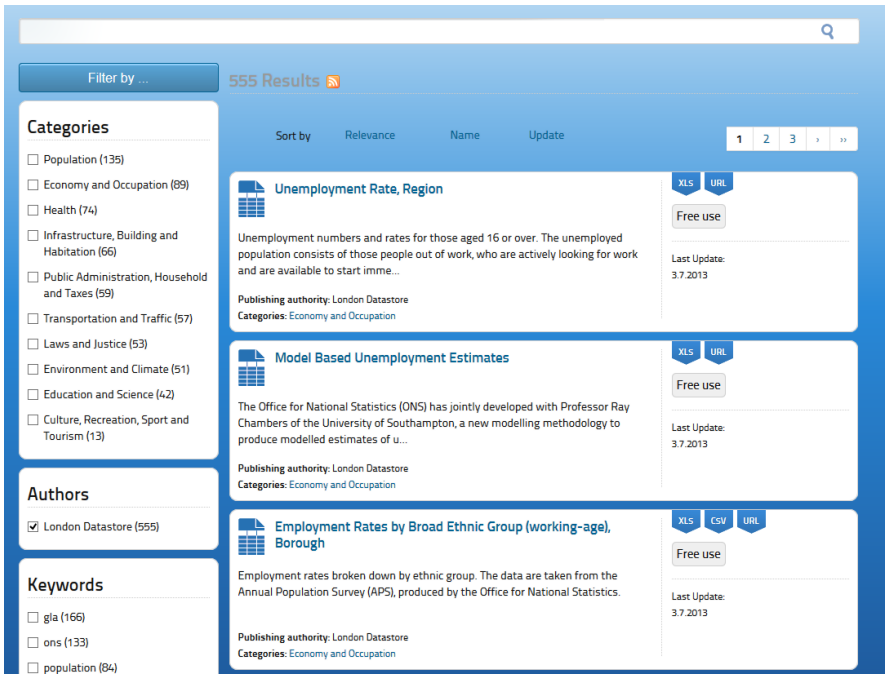

Figure 15 The Data Catalogue View

Each dataset is provided with a detail view to browse the full data of it. All metadata and resources are displayed in this view. In addition registered users have the option to comment and rate on the dataset. The metadata includes a description, license, date, categories and keywords. For the iCity project the integration with social networks has been implemented. It

enables users to communicate in Facebook, Twitter or in Google+ about datasets directly from the dataset detailed view (see the figure below).

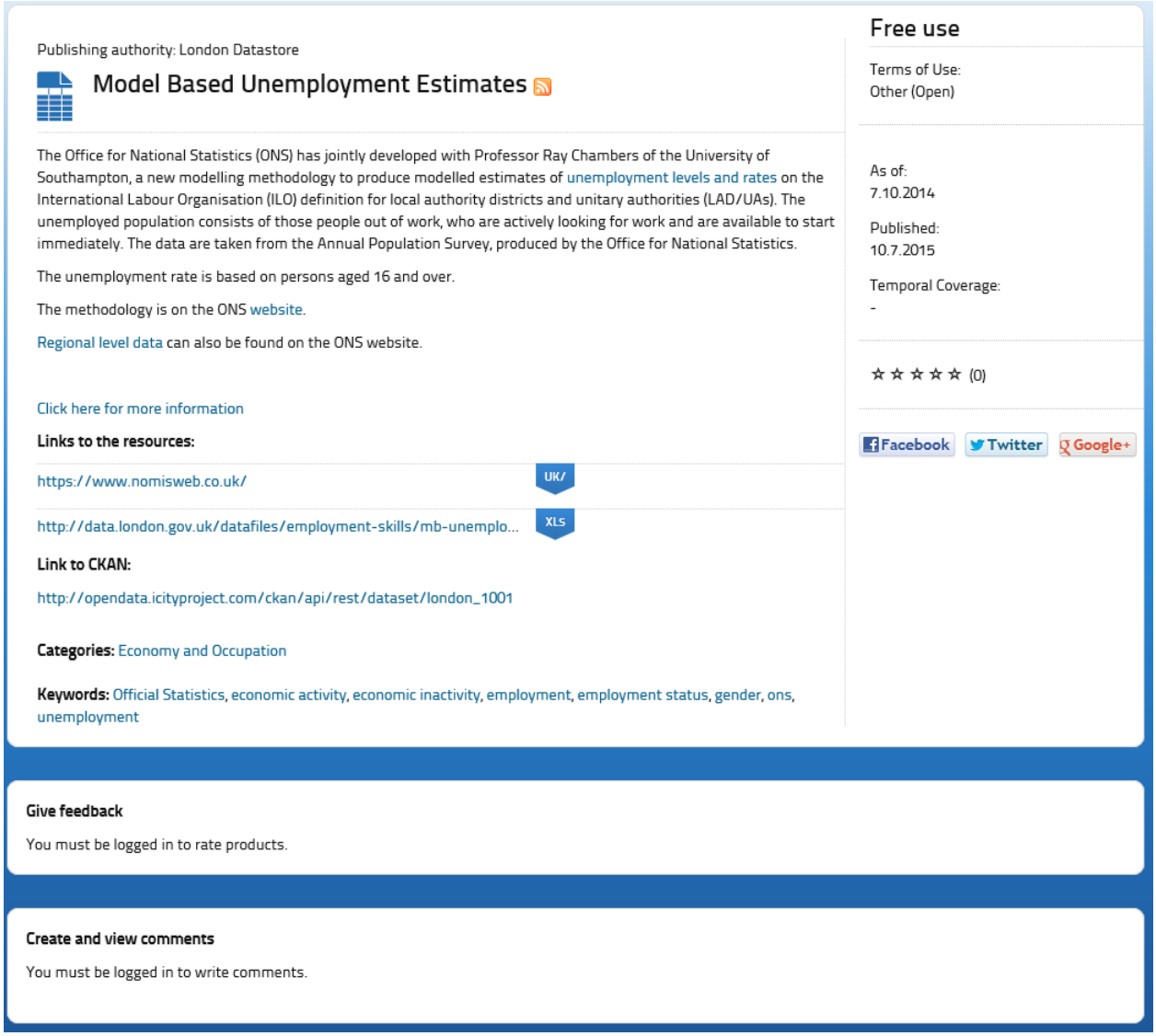

Figure 16 Detail View of a Dataset

Besides those basic functionalities the portal offers several roles for users, providing them with additional rights. Those roles are defined by entities:

- **User**: Users have basic permissions to act within the portal. To be assigned this role a person needs to be authenticated, thus registered and logged in.
- **Data Provider**: Users with advanced permissions responsible for publication of open datasets or apps in the platform.

- **Editor**: Editors are users with an extended scope of permissions. They have access to several configuration functions of the control panel and can edit and delete web content, like the static content or the news section.
- **Administrator**: Administrators are super users who have all kinds of permissions. They have full access to the control panel and e.g. can assign roles and define permissions.

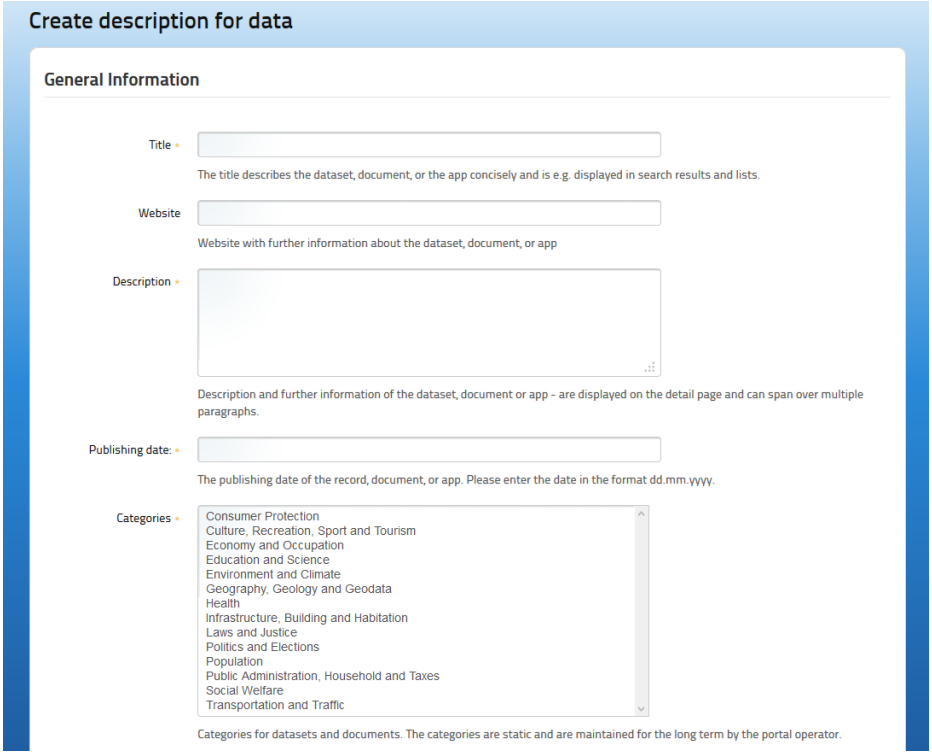

Figure 17 Creating a Dataset

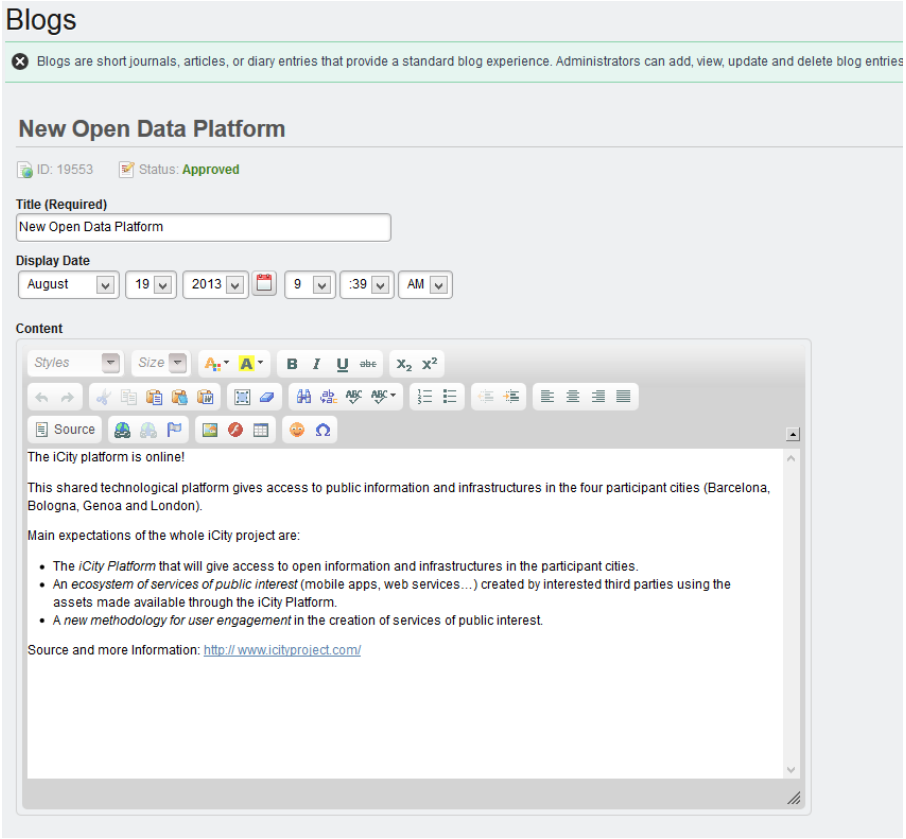

Figure 18 Editing a Blog/News Entry

4.2.2 Public Portal

The public Portal is the portal that can be accessed by everybody with internet access. No passwords are required.

The public portal provides information about the iCity Project and a link to the other portals that require registration. This portal is aimed at social networks and highlights the most important news and events.

The Public Portal can be reached at following URL and is open for the public: http://www.icityproject.eu/

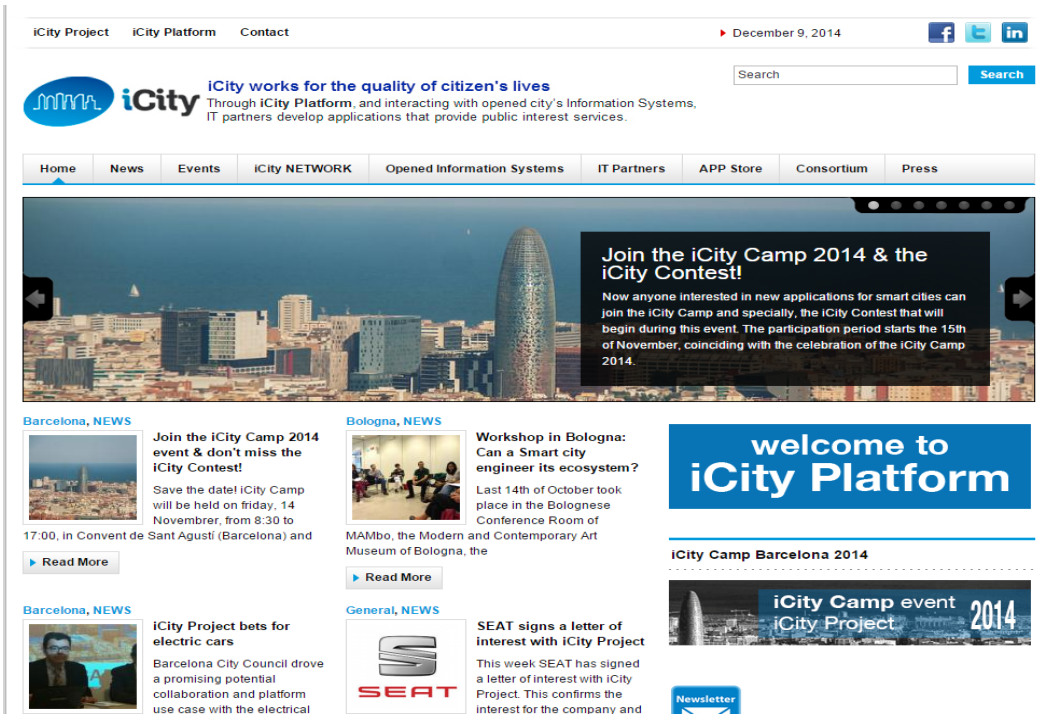

Figure 19 iCity Public Portal

From May-June 2014, when it was activated the new web; to October 2014 it's been monitoring the following results:

General Public Overview	may-14	june-14	aug-14	sep-14	oct-14
Sessions	198	557	538	753	1278
Users	157	335	456	598	956
Pages	244	4028	1678	2035	2878
Pages/session	1,23	7,23	3,12	2,7	2,25
Bounce	87,37%	46,68%	64,13%	61,22%	60,33%
% new sessions	72,73%	56,73%	79,00%	74,90%	71,06%

General Public Overview	Nov-14	Dec-14	Jan-15	Feb-15	Mar-15
Sessions	1.735	1.106	983	1.540	1.399
Users	1.231	860	836	1.125	1.026
Pages	4.304	2.644	2.121	3.599	3.502
Pages/session	2,48%	2,39%	2,16%	2,34%	2,5%
Bounce	53,95%	66,55%	61,65%	55,65%	59,61%
% new sessions	66,40%	71,61%	79,45%	67,66%	67,19%
General Public Overview	**Ap-15**	**Jun-15**			
Sessions	1.085	1.184			
Users	936	1.065			
Pages	2.345	2.078			
Pages/session	2,16%	1,76%			
Bounce	70,14%	64,36%			
% new sessions	80,83%	85,98%			

4.2.3 Developer Portal

The developer portal provides all the information required for developers to develop applications using the iCity API. It also provides reporting and management capabilities for the developers.

To access to iCity Developers Portal you can go to http://icity-devp.icityproject.com or click on the developer's portal in the public portal as seen in above figure.

Figure 20 iCity Developers Portal

4.2.3.1 *Logging in*

To log in to the API Portal:

Open your web browser and navigate to the following address: http://icity-devp.icityproject.com.

Click Login at the top of the browser window and then enter your Username and Password (if you do not have a login, please signup). To find more info on the Registration Process Scenario, click the link on our documentation page). The Dashboard is displayed:

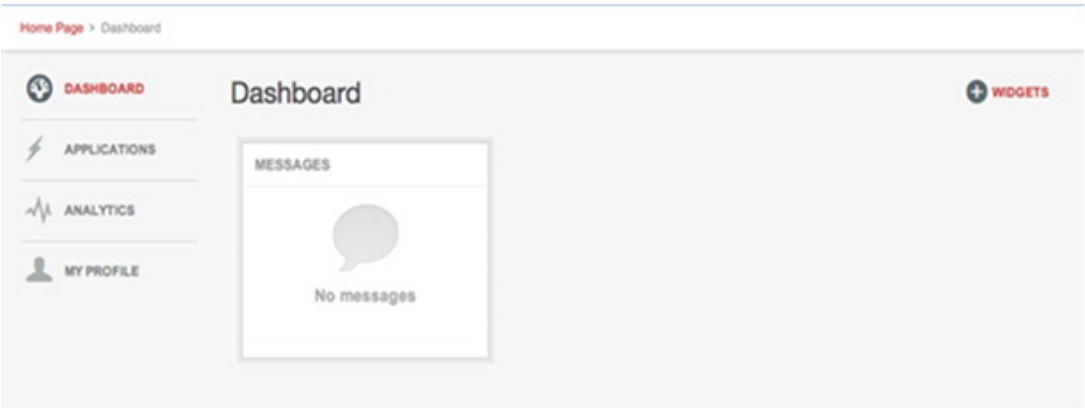

Figure 21 Developer Dashboard

4.2.3.2 ***Functionality by User Role***

The iCity Developers Portal has several user roles pre-configured on the system. These roles can be defined as being either internal or external. Internal roles are created on the CMS and internal to business of implementing the portal, whereas external roles are accounts that must be invited to the system.

When you login with administrator rights, you have access to the user management, the API management; you can approve users and applications and manage the overall status of the platform. (See more details in section 5 System Management)

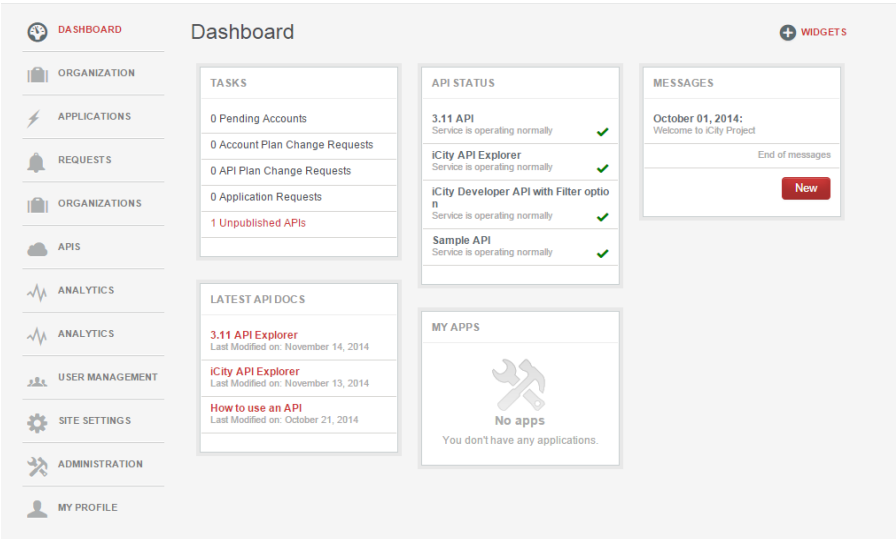

Figure 22 Login with admin rights

4.2.3.3 ***Navigation Structure***

Each user role sees different levels of the navigation to match their functionality. The following images display the different menu items available to each user role on the Dashboard page. This UI is based on the default configuration of the API Portal.

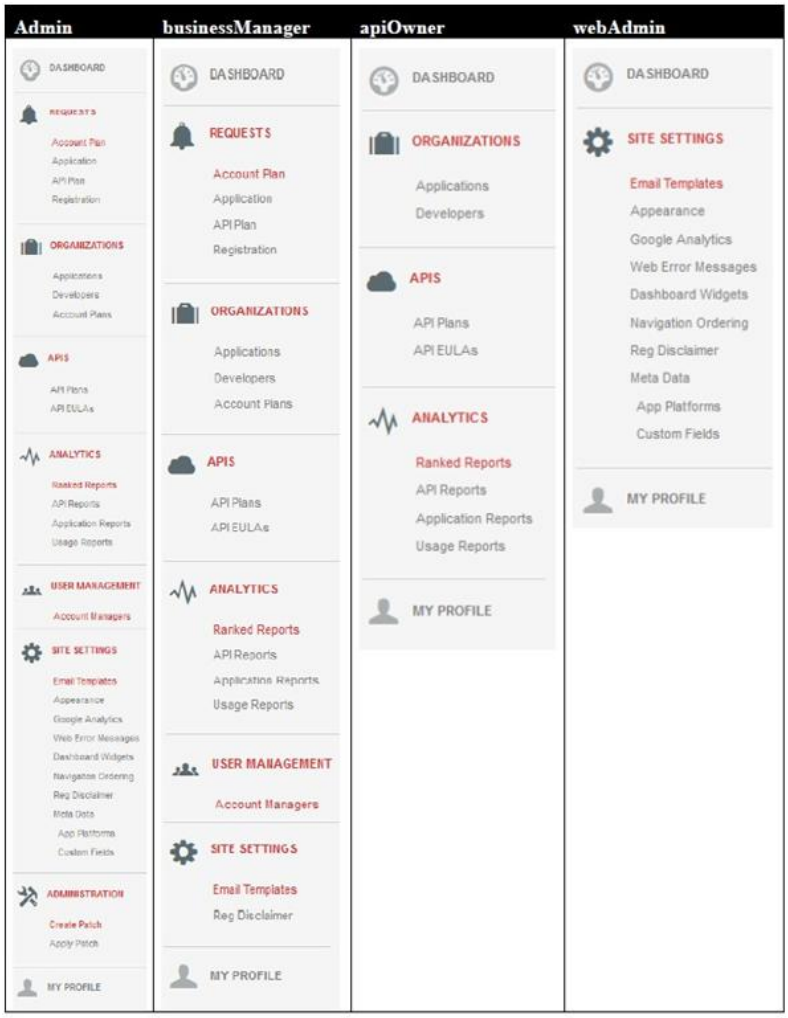

Figure 23 Navigation available to Admin, businessManager, apiOwner, and webAdmin

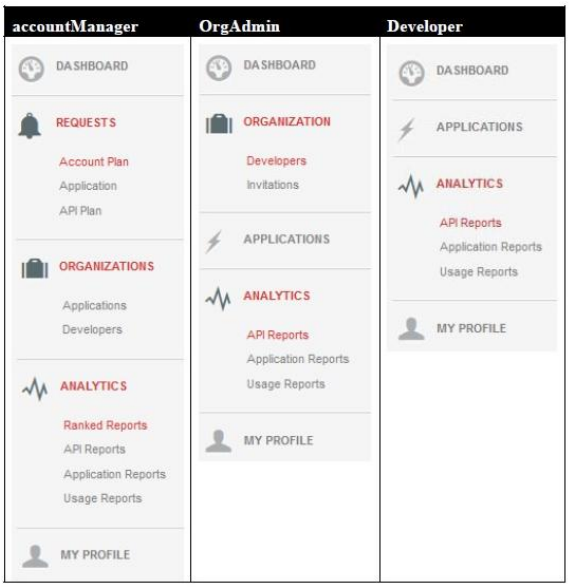

Figure 24 Navigation available to accountManager, OrgAdmin and Developer

4.2.3.4 *Register an account*

The first thing developers need to do is register for an account. They do this by signing up on the iCity API Portal and completing a registration form. [3]

If registrations are subject to approval, the developer will receive an email stating that the account is under review. Otherwise, the developer will be emailed a link to click on in order to activate the account.

Each new developer has assigned a default **Account Plan** when they finish the registration process. An Account Plan determines the **number of queries (hits)** that each user can make to

[3] The iCity API Portal does not permit registration of duplicate organization names. As a result, once the first developer from an organization has registered an account with the portal, subsequent developers from that same organization require an invitation to be registered.

any existing applications. For example, the default one (Bronze Account Plan) allows doing 10000 hits per day.

Each user can request an upgrade of Account Plan depending on their needs, increasing the value of the threshold's quota.

4.2.3.5 ***Add New Applications***

Developers can add applications of their own through the API Portal. Once they have completed the application information, the system will send them an email confirming the API application, and it will add the application to their Business Manager's queue to be approved. Applications created by Business Managers, Account Managers, or Administrators will automatically be given the status of Active.

Once the application has been approved or rejected, the Organization Admin will receive an email notification of its status. An approved application will have a status of Active and will be assigned an API Key. If an application is rejected, it will be returned with a status of Rejected with details of the rejection sent via email. You can then edit the application. Once you save your edits, the application will be added to the Business Manager's queue with a status of Revised. Rejected applications can only be revised once.

4.2.3.6 ***Manage Applications***

Developers can add, edit, enable, disable, or delete their applications via the Manage Applications page. The Manage Applications page also allows you to view your organization's Account Plan quotas.

Each new Application has assigned a default API Plan when the application has been enabled. An API Plan determines the number of queries (hits) that each application can make to their assigned APIs.

IMPORTANT: Deleting an application makes all report history for that application irretrievable. If Account Plan Quotas are in effect, API hits made via an application prior to deletion will continue to count toward the Account Plan Quota, even though these hits won't be shown in the

---

report. iCity recommends disabling applications instead of deleting them to maintain report accuracy.

4.2.3.7 *APIs Explorer*

APIs Explorer lets developers interactively discover APIs.

By making choices from among your API's valid resources and methods, and then submitting queries and viewing responses, developers can gain a better understanding of not only how your APIs work, but also the authentication methods required to access them.

Using an API Explorer

Use the API Explorer to test or change an API resource by sending a request. You can also view the queries sent that generated the response as well as code samples. Any published API with a WADL attached to it is automatically pre-populated into the API Explorer.

Because authentication methods are used to control access to each API resource on the server side, valid credentials are required in order to test the API.

To test an application with an API key, choose the application from the **API Key** drop-down list. This pre-populates the API Key value and API Key secret of the chosen application in the **Service Authentication** dialogue box.

Example response

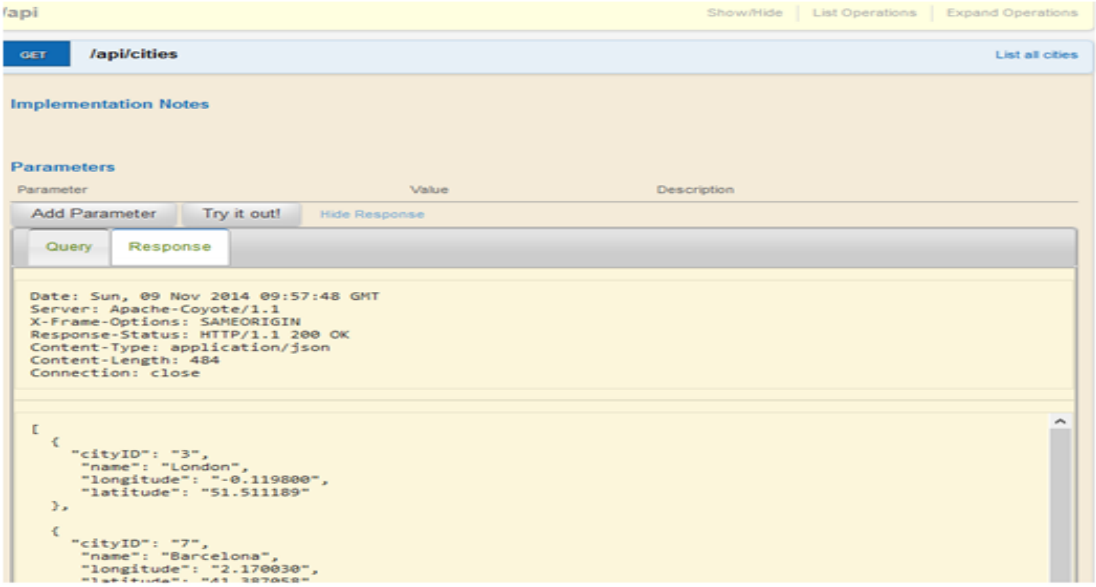

Figure 25 Example Response

Example Query

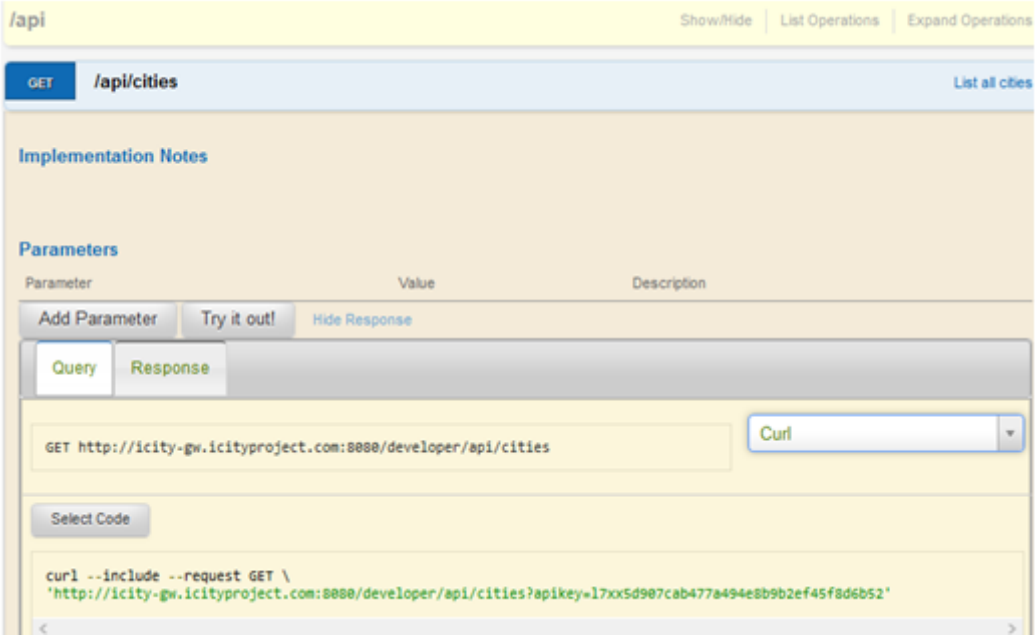

Figure 26 Example Query

4.2.4 iCityApps Catalogue

The iCityApps catalogue is realised as an online portal implemented on the base of the Liferay content management system. Liferay Portal[4] is a web platform for developing web sites and portals. iCityApps enables metadata management of apps or other applications and it is dedicated to apps relying on the iCity platform or other relevant apps.

Figure 27 iCityApps home page

[4] http://www.liferay.com/

The portal provides a text search and a faceted search according to the following categories as depicted in Figure 28 :

- Supported by the App platforms (WebApp, Android, Apple iOS, Blackberry, Ubuntu, Windows)
- Regions supported by the App (Europe, Barcelona, Bologna, Genoa, Berlin).
- Topics (Education, Family, Health, News, Reference, Navigation, Policy, Organisation, Travel, Social networks, Sport, Weather)
- Permissions requested by the app (Network communication, Media access, Location Services, System Settings, Paid services, Your personal data)
- Costs (Free, Paid)
- Uses iCity API – special field indicating if the app relies on iCity API or not.

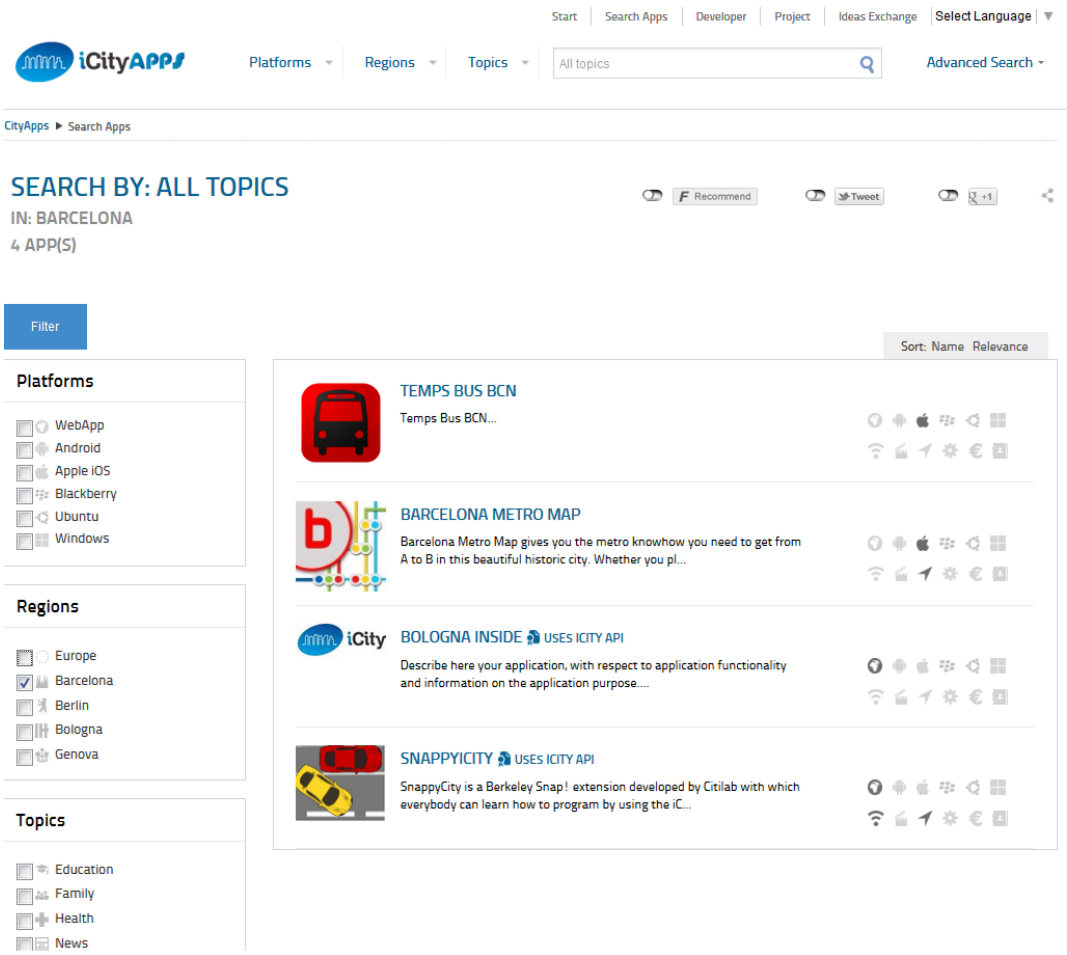

Figure 28 Search dialogue

By selecting one of the apps from the list user can get more detailed information about the app including (Figure 29):

- App description
- App screenshots
- Requested by the App access permissions
- Category, Size and Language of the App
- Regions address by the App
- Publisher of the App
- Links to app stores or website, where the App can be downloaded

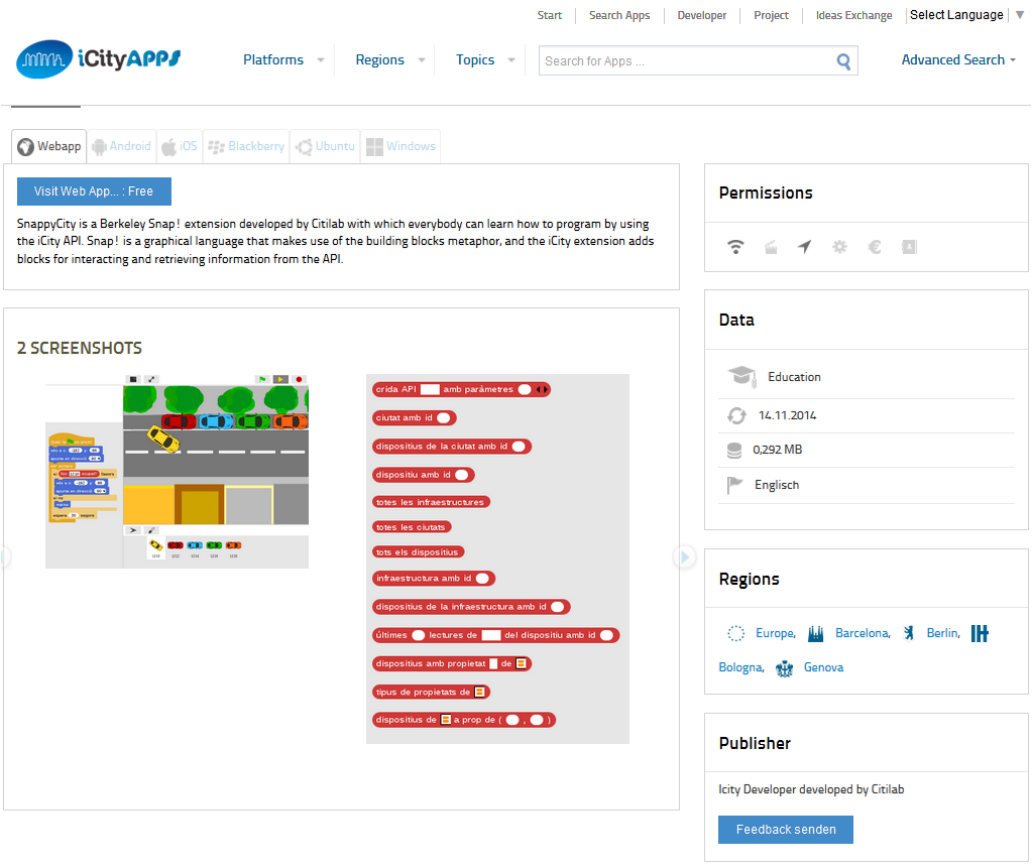

Figure 29 App details view

Any registered user can add apps in iCity Apps, but they have to be approved by a privileged user representing the city/region before they are published in the portal.

4.2.5 Information System Manager Portal

The Information System Manage Portal allows administrators to manage the Information Systems that are connected to the iCity Platform.

A security and control access layer, to provide security connections to Information Systems opened to iCity, has been included in the management system. This security and control access layer it's composed mainly by a token mechanism and API use control.

Also a management portal for Information System is being developed. This management portal will allow cities to control and manage access to their own Information Systems for the different applications which have required this data feed. The next functionalities are available to the Information System manager:

- Check applications that have access to a specific Information System.
- Check applications that are pending for approval to have access to a specific Information System.
- Allow or restrict the access of an application to their Information Systems.
- Connect/Disconnect an Information System.

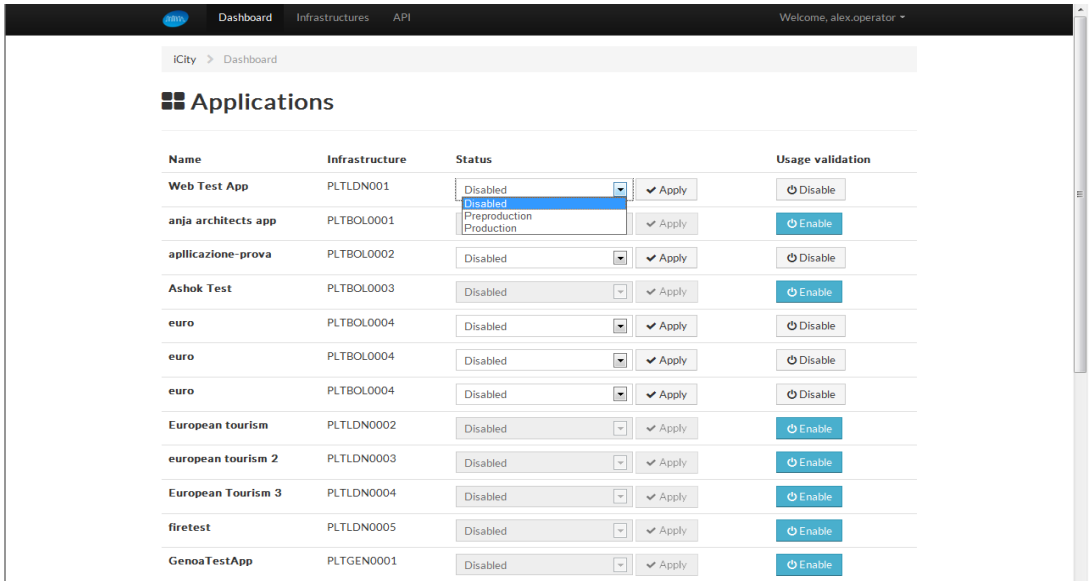

Figure 30 Information System Manager Portal

See section 5 iCity System Management (D 4.6) for more details.

4.2.6 Monitoring Manager Portal

The Monitoring Manager Portal offers to Administrators and operators roles a global view of the current status of the Platform. It offers visual information about the performance of the different components of the platform as well as monitoring the connectivity with the different Information Systems connected. It offers multiple views:

- The first one for administration portal allows the configuration of the parameters related to events and alarms of the monitoring module as well as manages the user's access the this portal.

- The monitoring window offers a view and informs about the status of the different components of iCity platform like element status, hardware status, network status, services monitoring, threshold alarms and spatial alarms.

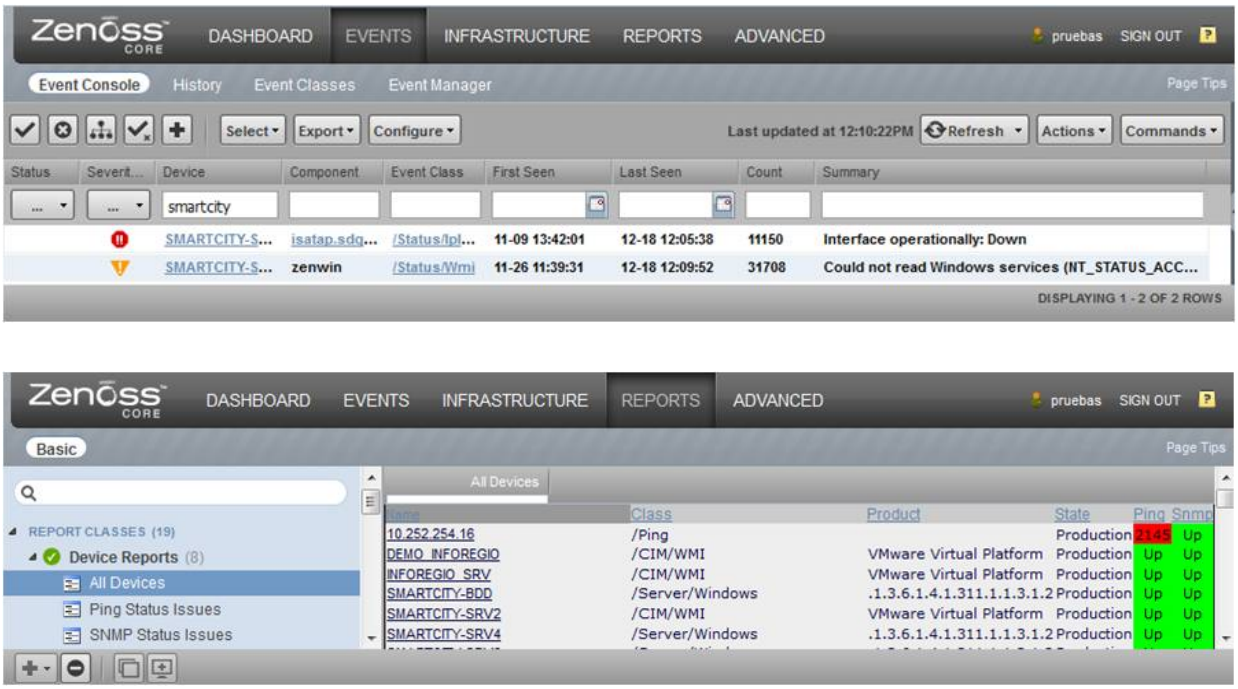

Figure 31 Monitoring Manager Portal

4.3 Manager

The Block Manager iCity Platform contains three big blocks to control and facilitate access to the data:

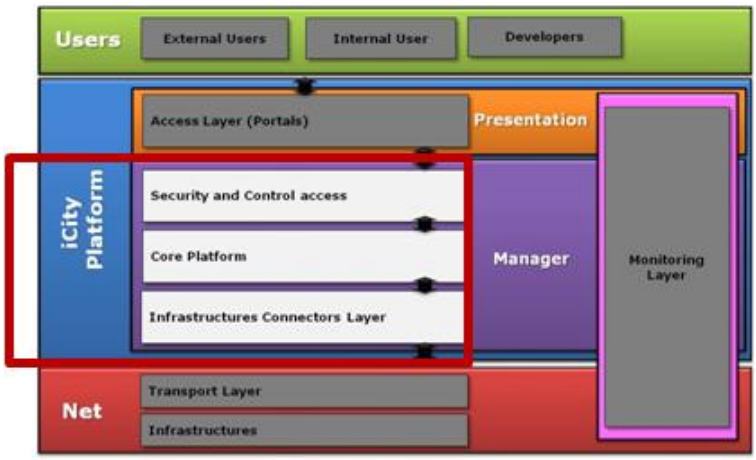

Figure 32 Manager Block

- Security and Control access
- Core Platform
- Information Systems Connectors Layer

See section 5 iCity System Management (D 4.6) for more details.

4.3.1 Security and Control Access

The security and controlling access available to users the APIs provided by the platform, adding a layer of security and access control. In this way it can be possible manage the access of developers and applications using Information Systems across the platform media offers.

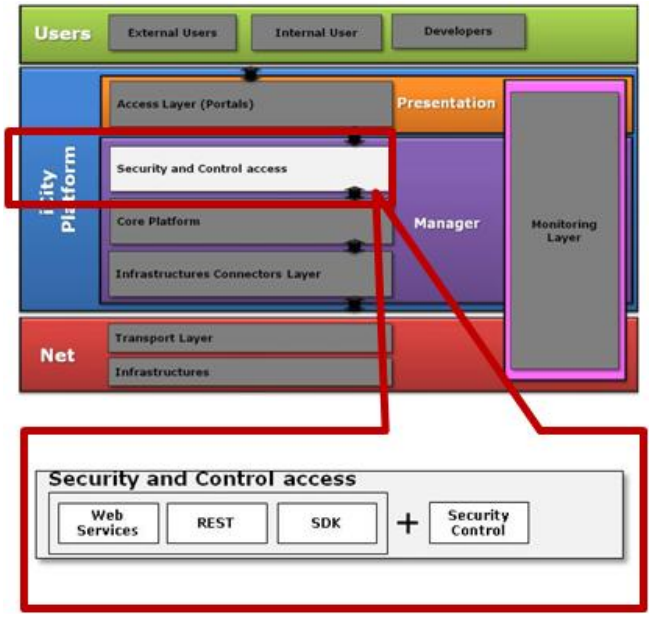

Figure 33 Security Control Access

4.3.1.1 *API management Platform models*

We have selected for the iCity Platform a software based (plug in) solution as this is the most appropriate choice to facilitate the deployment in the cities and to guarantee scalability and flexibility.

Given there is a wide variety of Information Systems in the cities, it implies the need for a wide range of integration functionalities and capabilities to define the policies for accessing those Information Systems.

See below an overview of the most known API management possibilities.

API Gateways

- A hardware appliance that an entity deploys on-premises in DMZ.
- Centre operational traffic management on appliances (Hardware), most of them, or software on virtual appliances.
- Allows managing operational API traffic in a Data centre (Private or Public).

- Best performance with predictable API traffic and enough of it to justify the purchase of multiple appliances.
- Good option for entities that are concerned about using public cloud or where cloud base provider offer is poor.

Cloud-based proxy

Service providers:

- Interpose their operational traffic subsystem between customers APIs and the client apps that call them.
- Check calls authorization privileges and routing it according to API plan and access capabilities.
- Handles the entire Information System for traffic authorization and developers portal.
- Minimum upfront.
- Good for entities that are testing with API, or with no internal capacity to deploy and manage API gateway appliances.
- Be careful, it can be an expensive approach if expect to handle millions of API calls on a daily basis.

Plug-in

- Software solution that the entity integrates into its own code and deploys wherever its servers are normally deployed.
- Gives API administrators direct access to the operational function of the API management platform, adding them as extensions to existing HTTP servers.
- Combines the on-premises traffic handling of the API model with the cloud-based authentication and portal capabilities of Cloud-Based Proxy model.

- On-premises operational traffic-shaping capability less expensive to deploy than an appliance model.
- Good when the entity has an API already in place and is looking to add security and provisioning functions on top of existing, on-premises Information System.

Critical in a city and public sector environment are the security aspects. Layer 7 has been found to be the best performing for our city deployment criteria.

In the below figure, you can see the API management components of the iCity Platform.

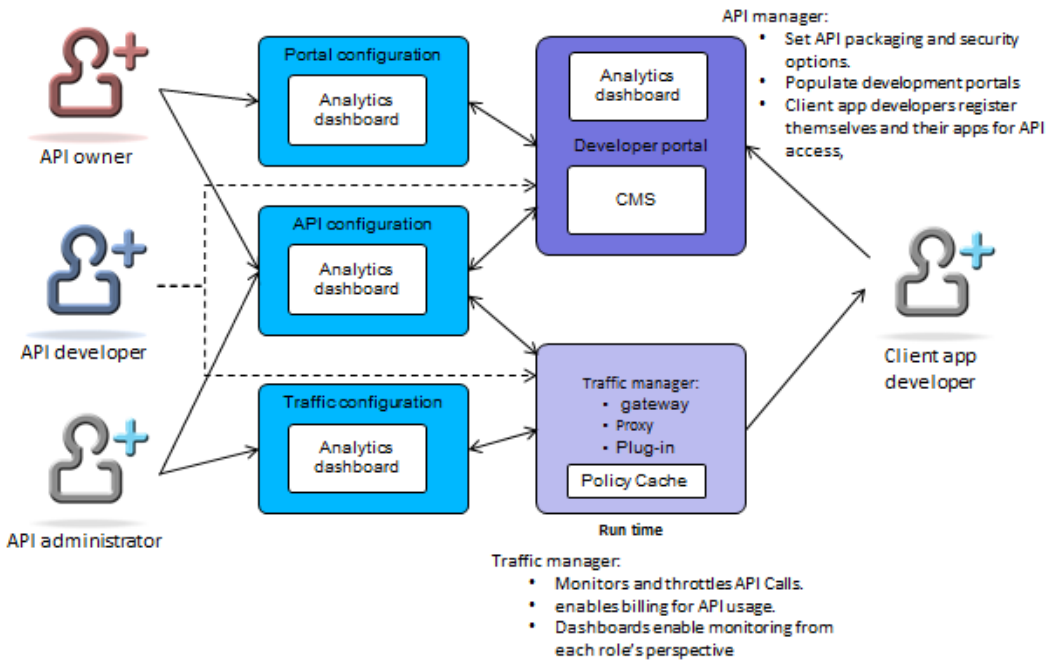

Figure 34 API Management Components

4.3.1.2 *Managing API Plans*

The API Owner can view, edit, or delete API Plans on the Portal via the Manage API Plans.

Each new Application has assigned a default API Plan when the application has been enabled. An **API Plan** determines the number of queries (hits) that each application can make to their assigned APIs.

For example, the default one (**Sandbox API Plan**) provides an appropriate API service level for developers who are starting to build an application.

Developers can request a different API Plan (for example), when they move their application into test or production.

API Owners can also perform a variety of tasks from the Manage API Plans page as view the applications, organizations and members of an API Owner Group associated with an API Plan.

4.3.1.3 *Authenticating an API*

In order for a request to execute correctly, an API must be authenticated on the iCity Developers Portal with an API Key Authentication method.

To get an API Key, Developers must be registered in the iCity Developers Portal and make a request for an Application proposal. Once approved, they can use their API KEY to access to the APIs. (see more details Appendix IV: Registration process & Appendix V How to use your token)

To authenticate an API using an API key:

- From the API Explorer page, select the API to authenticate.
- Click [**Authentication**].
- Choose **API Key** from the **Service Authentication** drop-down list.
- Enter the **Name** of the API Key to add. This field is required.
- Enter the **Value** of the API Key to add. This field is required. The API key must be generated on the iCity API Portal.
- Select whether the API Key Type is part of the **Query** parameter, or part of the request **Header**.
- Click [**OK**] to validate your input and add it to the request.

4.3.2 Core Platform

The Core of the Platform is the element placed between the connectors and the access layer. This module offers a secure and easy access to data and Information Systems connected to the Platform.

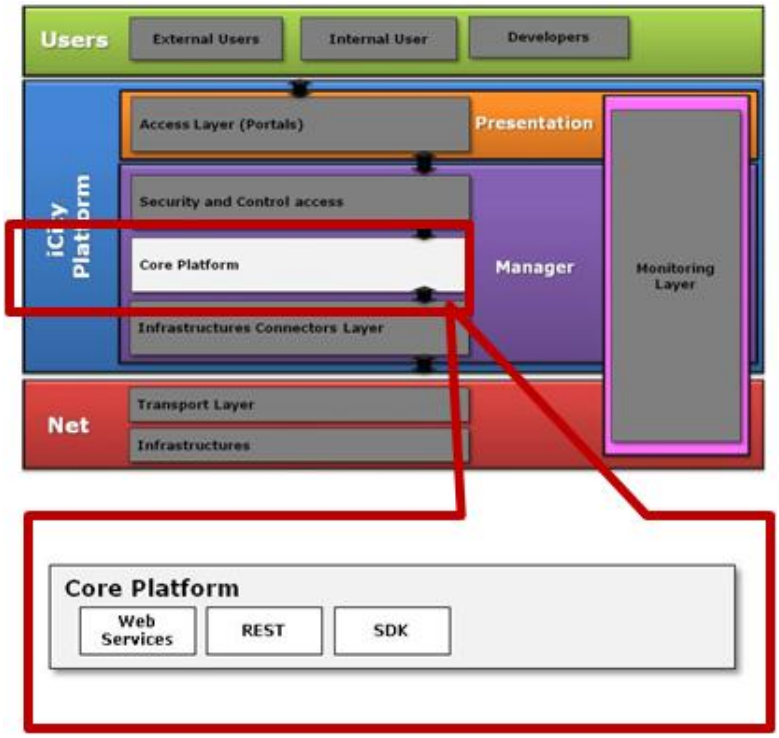

Figure 35 Core Platform

To provide access to these Information Systems it offers many options:

- **Web Services:** The web services are used as a standard protocol to provide access to data from any Information System.
- **API REST:** The API REST provides the same capabilities as the Web Services by using another standard protocol.
- **SDK:** This Software Development Kit offers to developers the information and methods for accessing to the data from iCity Platform. The SDK have 3 important blocks:
    - Example Application to test functionalities.

- Extended Documentation.
- .NET library to make developments easily.

See Section 9 iCity SDK (D 4.10) for more details.

4.3.3 Information Systems Connectors Layer

The Information System Connector Layer provides a connector for every Information System.

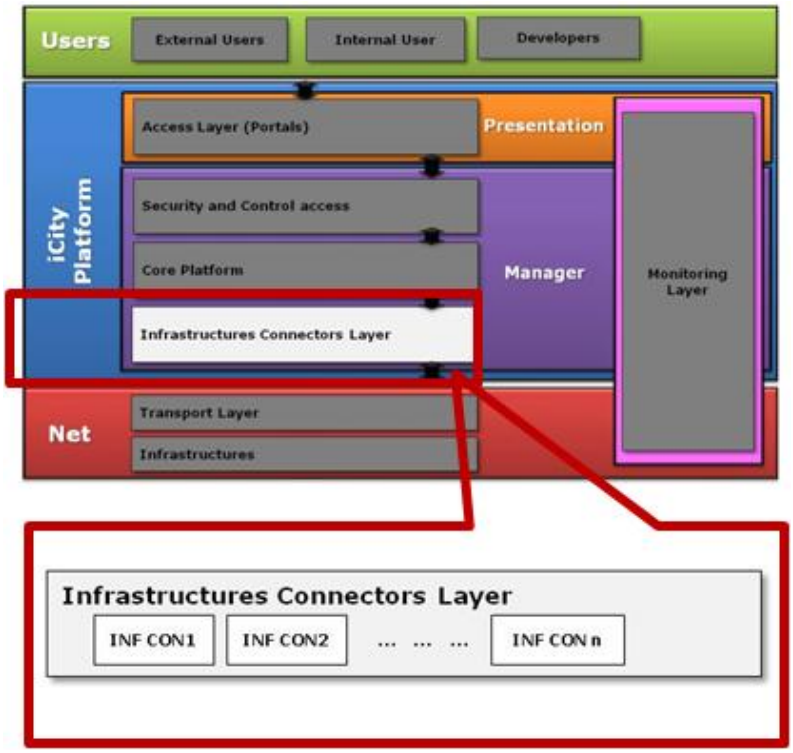

Figure 36 Information System Connectors Layer

Usually every device manufacturer works with a proprietary protocol. For this reason a connector is created to interact and obtain information from those Information Systems. All connectors are transparent for end users, which will only interact with iCity API on the access layer. At M42 the iCity Platform Prototype worked on and has integrated the following Open Information Systems from each one of the cities involved in the project (see also Chapter 6- Integration of new Information Systems for further details).

- ABERTIS Telecom: Smart Zone- Urbiotica Sensors
- ABERTIS Telecom: Smart Zone- Parkare Sensors

- BARCELONA: Sentilo
- BARCELONA: Smart Citizen Platform
- BARCELONA: Guia – Agenda
- BARCELONA: Guia – Facilities
- BARCELONA: IRIS
- CORNELLÀ: Agenda
- GENOA: Weather Station
- GENOA: Citizen's Desk
- GENOA: Traffic Webcam System
- GENOA: Wifi Hotspots
- GENOA: Toursim Webcams
- GENOA: Air Sensors
- BOLOGNA: TPER-QueryHelloBus
- BOLOGNA: QueryHellobus4ivr
- BOLOGNA:QueryResale
- BOLOGNA: CISIUM Events
- BOLOGNA: CISIUM Metropolitan Traffic
- BOLOGNA: CISIUM Parking
- BOLOGNA: CINETECA – Catalogue of DVDs and VHS
- BOLOGNA: CINETECA – Events
- BOLOGNA: Air Quality
- BOLOGNA: Wifi Location & Live Monitoring
- LAMIA: Issue Reporting
- LONDON: Air Quality
- LONDON: Journey Planner
- LONDON: AlertMe
- ZARAGOZA: Suggestion & Complaints

See Section 6 iCity System Operation for more details.

4.4 Monitoring Layer

The monitoring layer checks the status of the different components that make up the platform iCity. (see Section 5 iCity System Management and Section 6 iCity System Operation).

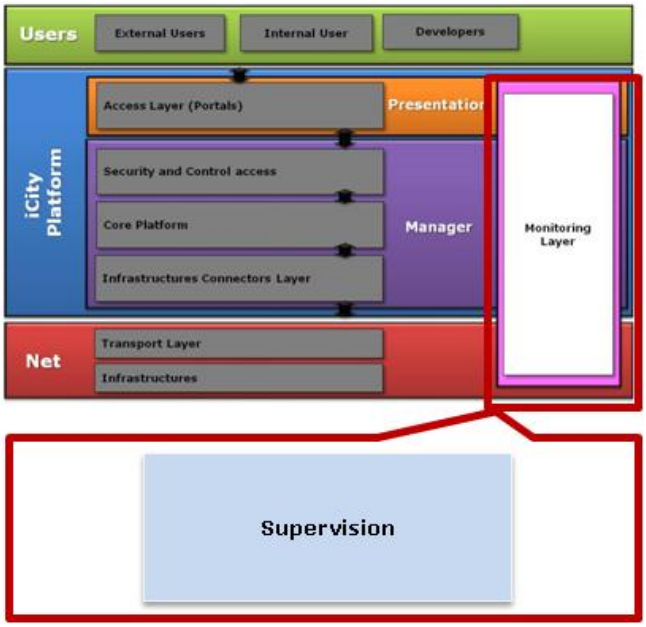

Figure 37 Monitoring Layer

The system analyzes:

- The status of the physical elements.
- The status of the network elements.
- The services running on operating systems.

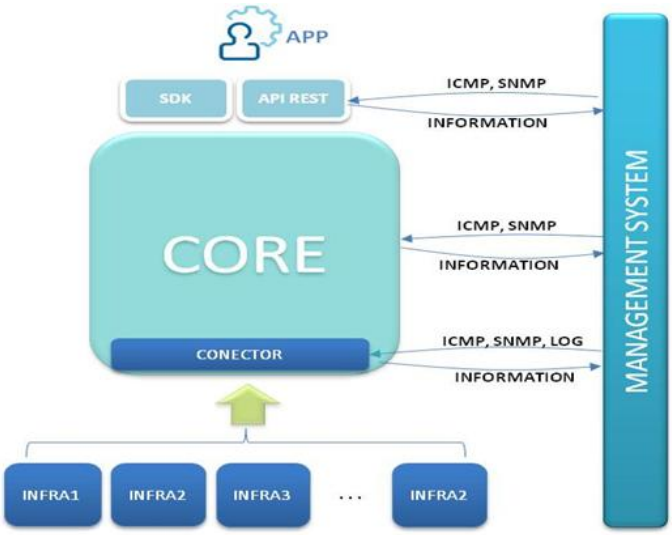

Figure 38 Monitoring Layer Modules

5. iCity System Management

The aim of this section is to set up the first version of system management prototype, which has the main goal t provide real time information concerning the cities Information System integrated in the iCity Platform and also the second year version which includes new functionalities that improve the System Management prototype with the inputs received from pilots and other WPs.

5.1 Purpose

The system management has to be able to work with different Information Systems and also has to provide access to them identifying each one of the external authentication processes. It has to establish an access control in order to decide the different type of information.

This activity is alive until the end of the project and the system management prototype will be enriched following the inputs coming from WP3, WP5 and WP7.

Finally, it is important to mention that this document will be alive and modified during the whole life of the project in order to adapt its contents to the final iCity platform architecture.

This section is mainly focused on the definition of different features of system management, in order to describe in a easy way the prototype and also show status of the prototype at M42.

5.2 System Management Definition

5.2.1 Security Access

The main goal of security access procedure is to provide iCity platform the right levels of authorization and access to iCity resources, confidentiality and privacy assurance.

Thus, iCity platform must implement a system management that provides cross functionality to register all the actions on the iCity platform components.

The management system has to provide components for treating key aspects of security, tracking and reporting not only iCity platform activities but also apps that use iCity platform modules. Therefore system management is responsible for:

- Provide both authentication and authorization services; and ensure proper levels of privacy, confidentiality and integrity of data.
- Log all the activities performed on iCity platform components, identifying relevant information from these actions.
- Use information from the iCity services catalog to manage all activities related with them.
- Log all the activities performed on iCity platform components, identifying relevant information from these actions.

The system management will cover the main requirements as:

- Authentication
- Single Sign On (SSO)
- Authorization
- Confidentially of data
- User privacy
- User management: profiles, access, etc.
- Tracking management: settings, queries, views
- Tracking log registration
- Reporting

5.2.2 Functional description

5.2.2.1 *Authentication & Authorization*

This component will be responsible for ensuring right levels of protection, security and privacy assurance in the iCity platform, as well as services or apps that interact with iCity platform.

The main goal is manage authentication and authorization of “users” (apps and iCity services).

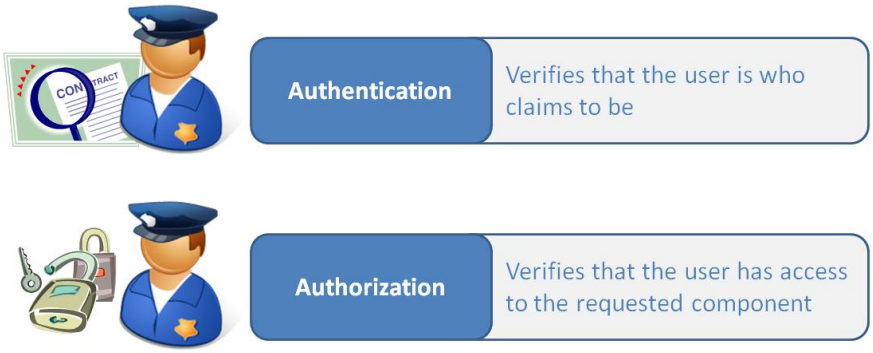

Figure 39 Authentication & Authorization

Furthermore, system management has to ensure secure interactions between users and iCity platform components, and covering following items:

- **Confidentiality:** Assure the privacy of the information, especially personal data.
- **Integrity**. Ensure that third parties do not manipulate information.
- **Authentication**. Assures end-to-end identity of users and iCity platform components involved in a process.
- **No-rejection**. Ensure is not possible to refute the validity or ownership of information exchanged through the platform.

Authorization access module will enable settings to manage policies based on different user profiles and iCity components. It will use the information provided by the catalog of iCity services applying the right polices.

System management will include a setting module to manage security aspects related to users and services authentication and authorization requirements.

Internal Roles

The following “internal” user roles are preconfigured on the iCity Developers Portal:

- **Administrator**: the super user with access to all functionality for all the roles listed below.
- **WebAdmin**: the person responsible for setting up the API Portal, including:
- **API Owner**: The person within your organization tasked with defining, publishing, or promoting your APIs. On the iCity Developers Portal, this person will be responsible for:
    - Measuring the effectiveness and usage of their APIs using the Analytics and Reporting feature
    - Defining API Plans.
    - Publishing the APIs for use by developers.
    - Manage organizations
    - Edit, enable, or disable applications
    - Email Organization Administrators
- **Business Manager:** The person within your organization tasked with managing the developers who sign up to use your APIs. On the iCity Developers Portal, the Business Manager will be responsible for these tasks:
    - Defining Account Plans (i.e., technical support levels) that can be assigned to each developer
    - Assigning Account Managers to developers
    - Measuring the rate at which developers sign up
    - Ensuring that SLAs (Service Level Agreements) are being adhered to, by using the Analytics and Reporting feature
    - Process requests (such as application requests, API Plans, and Account registrations)
    - Manage organizations (the same as API Owners)

- Edit email templates and registration disclaimers

- **Account Manager:** The person inside your organization in charge of assisting the Business Manager and the developers. On the iCity Developers Portal, this person will be responsible for these tasks:
  - Approving API and Account plan requests
  - Managing the developer's account on a daily basis
  - Managing organizations (similar to an API Owner)

External Roles

The following "external" user roles are preconfigured on the iCity Developers Portal:

- **OrgAdmin:** The owner of an organization. This person is responsible for managing his or her own organization and is usually the unique developer or the first one to be registered in the organization.
- **Developer:** A user that has been invited to join the iCity Developers Portal by an organization owner (OrgAdmin).

The following table summarizes the tasks each user role can perform.

	API Owner	Bus. Mgr	Acct. Mgr	Dev	Web Admin	Admin
Task						
View APIs	X					X
Publish APIs	X					X
Use or Designate Private APIs	X	X	X			X
Deprecate API	X	X	X			X

	API Owner	Bus. Mgr	Acct. Mgr	Dev	Web Admin	Admin
Task						
Add/Edit API EULAs	X					X
View and Message OrgAdmin	X					X
Create and Manage Account Plans		X				X
Request Account Plan Change				X		
Manage Account Managers		X				X
Manage Organizations (Access varies by user role)	X (for APIs)	X (all orgs)	X (only assigned orgs)	X (if OrgAd-min)		X (all orgs)
Manage or Work with Applications (Access varies by user role)	X	X	X	X		X

	API Owner	Bus. Mgr	Acct. Mgr	Dev	Web Admin	Admin
Approve/ Reject New Accounts		X				X
Approve/ Reject API Plan Requests		X	X			X

	API Owner	Bus. Mgr	Acct. Mgr	Dev	Web Admin	Admin
Approve/ Reject Account Plan Requests		X	X			X
Assign Private API Access (to Devs)		X				X
Register for an Account				X		
Add New Apps		X	X	X		X
Use the API Explorer	X	X	X	X	X	X
Work with Interactive Docs	X	X	X	X (can not edit)	X	X
User Management						X
Access the Site Settings		X (some)			X	X
Administration						X
Access the Content Management System (CMS)	X				X	X

5.2.2.2 ***Reporting and Tracking***

This module is in charge of activity tracking (logs) and transactions registry (reporting).

It will provide tracking registry of iCity platform components and services or apps, generating activity messages. Every track should include a time stamp, the security level of the track, the app and/or iCity service, and the description of the action.

Developer Reports

Developers have access to both application reports and API reports. In addition, each report offers two views located on two separate tabs: Usage and Latency.

The Application reports allow developers to:

- **View usage for an application**: Select an application from Application drop- down. The graph shows total API queries/requests (a.k.a. “hits”) for that application.
- **Top graph**: Shows all hits against all the APIs the application uses.
- **Middle graph**: Shows, of the total hits, how many successfully received a reply (i.e., resulted in a successful transaction).
- **Bottom graph**: Shows, of the total hits, how many did not receive a reply (i.e., resulted in an error).
- **View latency for an application**: Select an application from Application drop- down. The graph shows average latency for that application.
- **Top graph**: Shows the time it takes for an application request to enter the API Proxy, get passed to the back-end API(s), and then leave the API Proxy.
- **Bottom graph**: Shows the time it takes for a request to be processed by the API Proxy.

The API reports allow developers to:

- **View usage for an API**: Select an API from the API drop-down (note that the API drop-down will be populated only with APIs to which the developer has access). The graph shows hits against that API.
- **Top graph**: Shows total hits against the API from all the developer’s applications.
- **Middle graph**: Shows, of the total hits, how many successfully received a reply (i.e., resulted in a successful transaction).
- **Bottom graph**: Shows, of the total hits, how many did not receive a reply (i.e., resulted in an error).

- **View latency for an API**: Select an API from API drop-down. The graph shows average latency of the API for all the developer's applications.
- **Top graph**: Shows the total, round trip time for an application request to enter the API Proxy, go to the back-end API(s), and then pass back through the API Proxy.
- **Bottom graph**: Shows the time it takes for a request to be processed by the API Proxy.

Usage Reports

iCity Developers Portal provides a high level view of Account Plan usage by organization.

To view usage reports:

- From the Dashboard, select **Analytics** > **Usage Reports** from the navigation sidebar. The Usage Reports page displays.

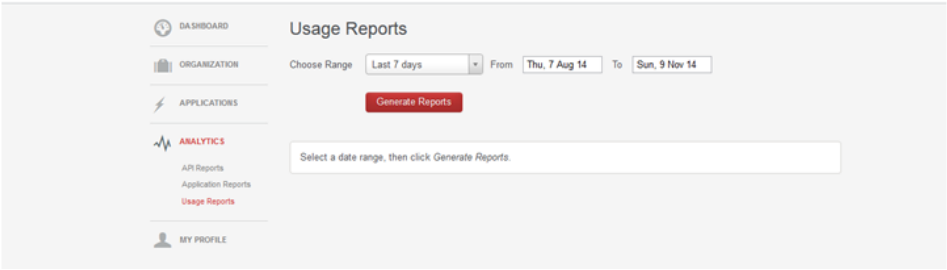

Figure 40 Usage Reports Dashboard

- In the **Choose Range** drop-down, select the date range, from Last 24 hours, Last 7 days, Last 30 days, and Last 365 days. Alternatively, you may select specific dates in the **From** and **To** fields. If you belong to more than one organization, choose the organization to report on from the **Organization** drop-down.
- Click [Generate Reports] to view the report.

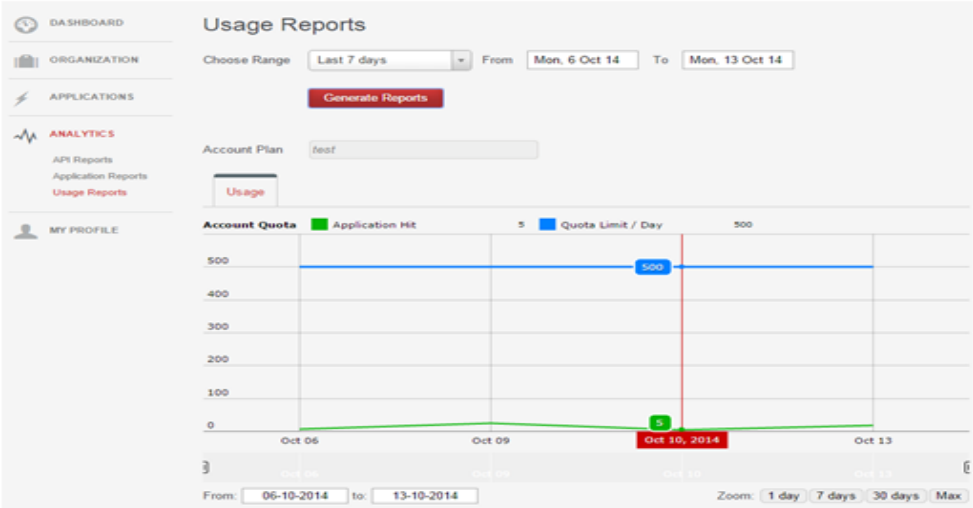

Figure 41 Usage Reports Generator

5.3 iCity System Management Prototype

5.3.1 City Admin Portal

The iCity Private Portal includes a management module for managing user's portal and registration.

When a user or developer wants to access to the private part they should first sign up for a registration in the portal.

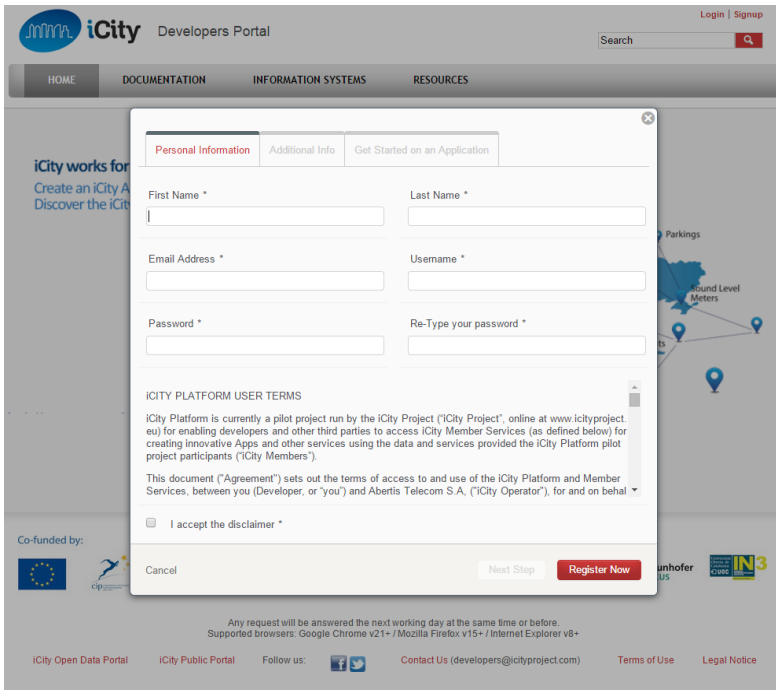

Figure 42 iCity Private Portal Registration

Then users and developers will receive an email confirming that the user's account has been approved.

Figure 43 Registration process

See APPENDIX IV: REGISTRATION PROCESS for more details.

In order to approve an account the administration users should access to the Private part of the iCity Developers portal by using its credentials. Then in the dashboard inside the REQUEST section it will appear a table with the name of the users' accounts pending to be approved. By selecting the option it is possible to view the user's profile, reject or enable the user's account.

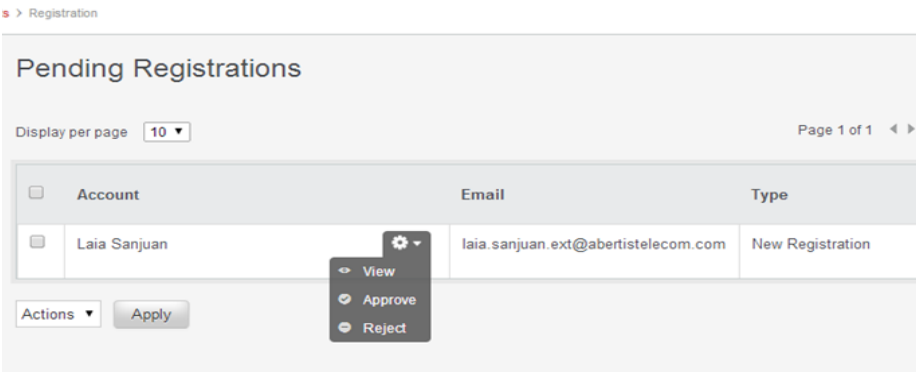

Figure 44 User's Account Management

Administrator users can control and disable a user's account by accessing to the Organization section and see the registered user's or developer's list.

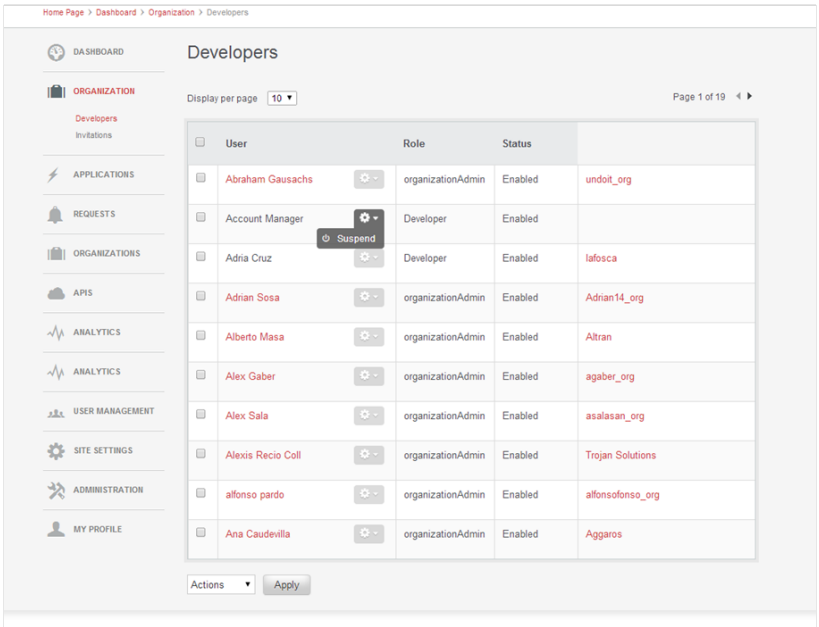

Figure 45 Registered user's list

There are some users in charge of the approval task. These will be account managers. As it happened before it is also possible to manage Account manager users by accessing in the User Manager section where it's possible to enable or disable the account. They will act like iCity Portal Administrators.

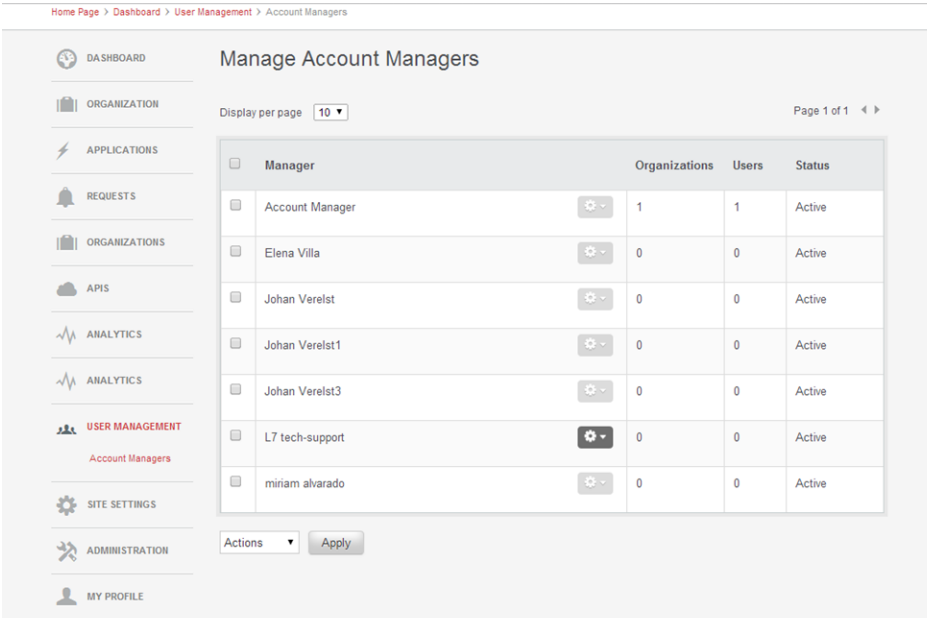

Figure 46 Manage Account Managers

5.3.2 Developer Roles

The first developer to register for your organization will also have the role of Organization Admin.

There are some preconfigured Developer roles on the iCity Developers Portal:

- **OrgAdmin**: The owner of an organization. This is typically a third-party user that signs up for an account on the iCity API Portal using the Registration Form. This person is responsible for managing his or her own organization and is usually the only developer or the first one to register for the organization.
- **Developer**: A user that has been invited to join the iCity API Portal by an organization owner (OrgAdmin). These users are enrolled under the OrgAdmin's account. Developers are responsible for creating and managing new applications.

5.3.3 iCity System Management

This system permits the management of all the iCity Platform elements and services as well as the creation, erase and administration of the System Management users and groups.

The **Advanced** menu makes possible the visualization and the management of all the users and groups of the system.

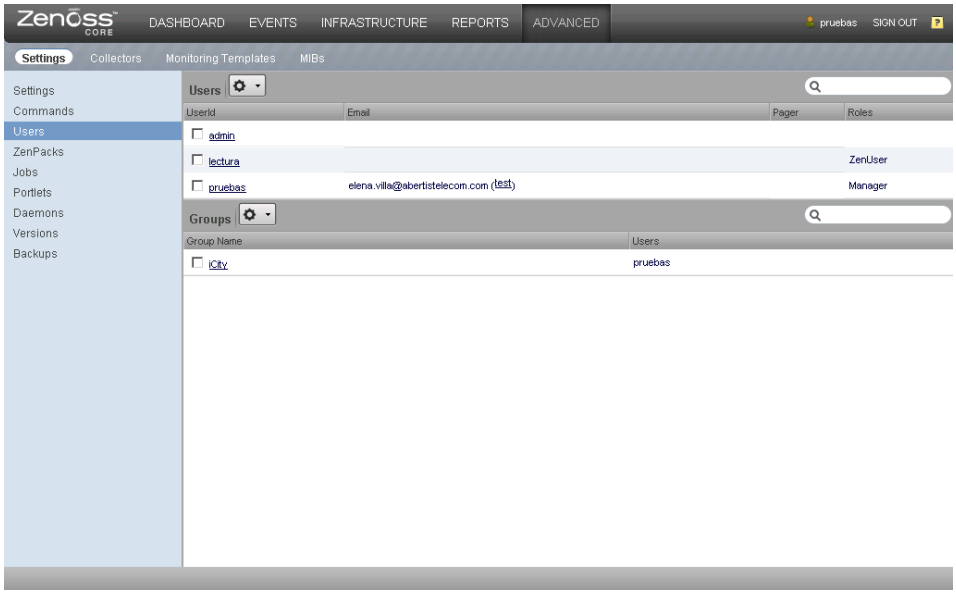

Figure 47 Example of Management System

The administrators can create new users assigning to them different roles or groups and modifying its properties.

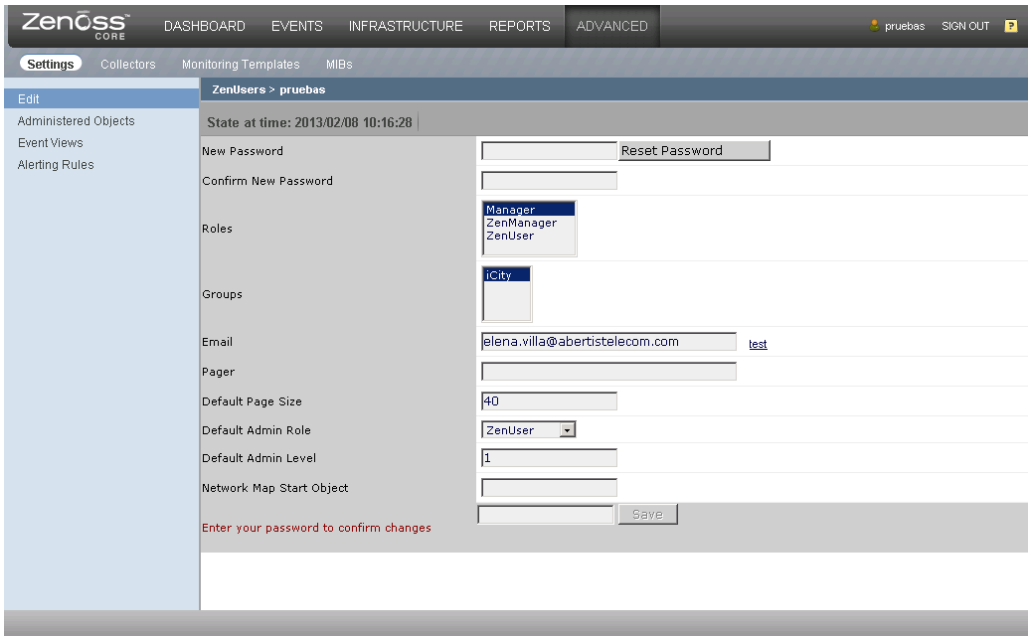

Figure 48 Example of User's Creation

In addition, the administrator users can create groups to allocate and agglutinate the different users of the System Management and facilitate their administration and permissions.

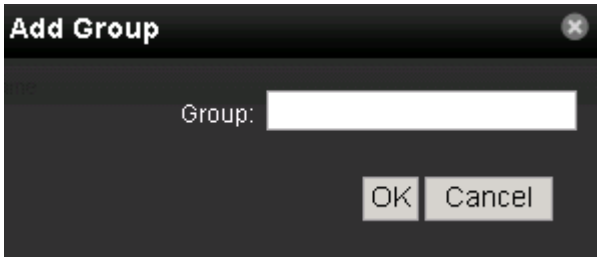

Figure 49 Example of Group Creation

The administrator users can manage and edit the existent ones and change their properties (passwords, group, e-mail, etc.).

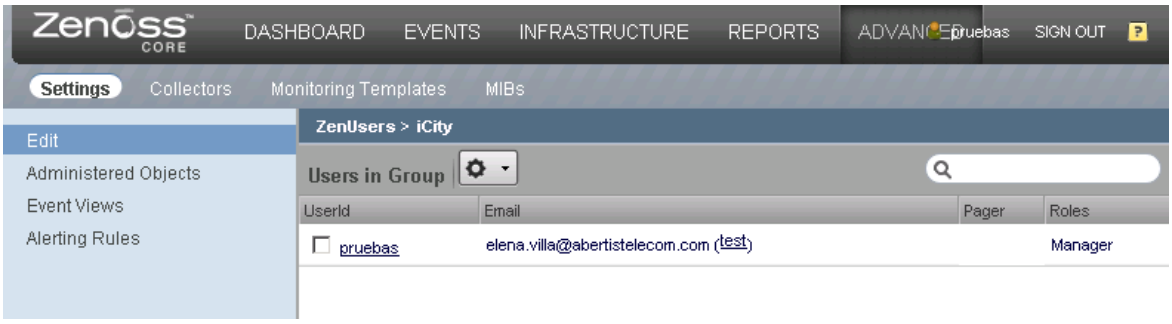

Figure 50 Example of Editing Users

The System Management permits the visualization of the number of events, separated by their severity level, of the iCity Platform elements and services associated to this user or group of users.

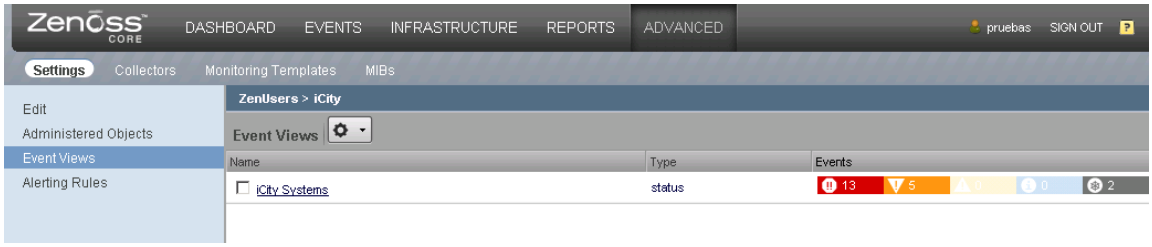

Figure 51 Example of editing Event Views

5.4 Prototype on M42

For the first year, a preliminary system has been included in the prototype in order to allow, in a short time, the deployment of pilots and also the development of apps, with basic system management adaptation that would provide access to different urban service delivery platforms for each external authentication process. This system is still accessible and useful for the third year prototype due this it offers the following benefits:

- Reduce risk of denial-of-service attacks.
- Permits graceful, linear scaling.
- Improved customer experience through easy service access, and streamlined service delivery.

- Reduced operational costs and prospective investments through flexibility and simplicity of service delivery.
- Provides extensive capabilities for integrated management of identity, rules, en-user devices, content and partners.

For the second year prototype, WP4 also realized the necessity of a system for managing the access to the Open Information Systems exposed by cities for the users and applications developed by iCity developers. Thus, during the second year WP4 worked on the definition and development of a portal for administrators and operators from cities.

For the third year prototype, the Information System's management portal is still being developed. It will allow cities to control and manage access to their own Information Systems for the different applications which have required this data feed in the iCity Developers Portal.

The main functionalities offered by the Information Systems Manager portal are the following:

- Check applications that have access to a specific Information System.
- Check applications that are pending approval to have access to a specific Information System.
- Allow or restrict the access of an application to their Information Systems.
- Connect/Disconnect an Information System.

When a Developer or a user wants to develop a new application, it has to make an application Proposal in the Developers portal. Then they have to indicate the Information System they want to use and have access for the data feed of their application.

In the custom field (Multi-selected Dropdown), called Information Systems they have to select the option they want to. Automatically this application is registered in the Information Systems Portal and will appear in the Manager console pending for approval. Once the application is approved by administrators the access should be enabled in the Information Systems Manager portal.

See APPENDIX VI: APP PROPOSAL PRE-VALIDATION for more details

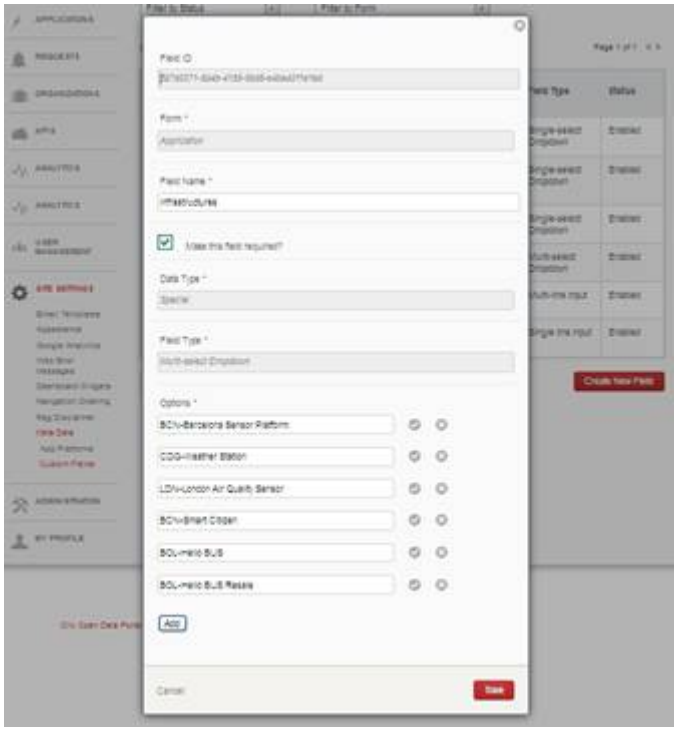

Figure 52 Application Proposal Information Systems dropdown

For accessing to that portal it will be necessary introduce some credentials in order to ensure the privacy and the restricted access for Administrators and Operators roles from involved cities.

Once the user accesses to the Portal they will see the management console, where there is a list of the applications registered.

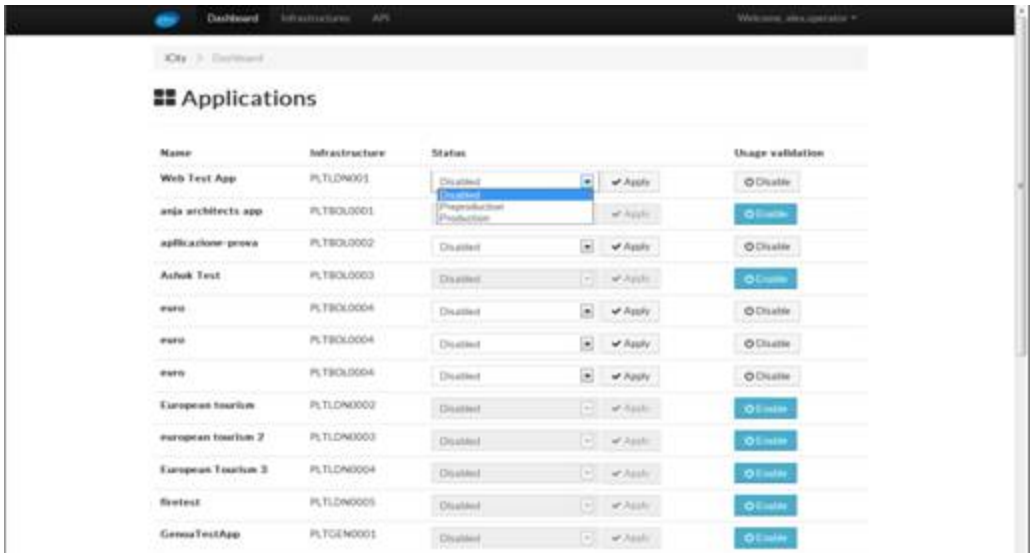

Figure 53 Information Systems Manager Portal

Each application approved will be assigned with a Preproduction Plan. When developers wants to upgrade to a Production Plan, should select the option in the iCity Developers Portal.

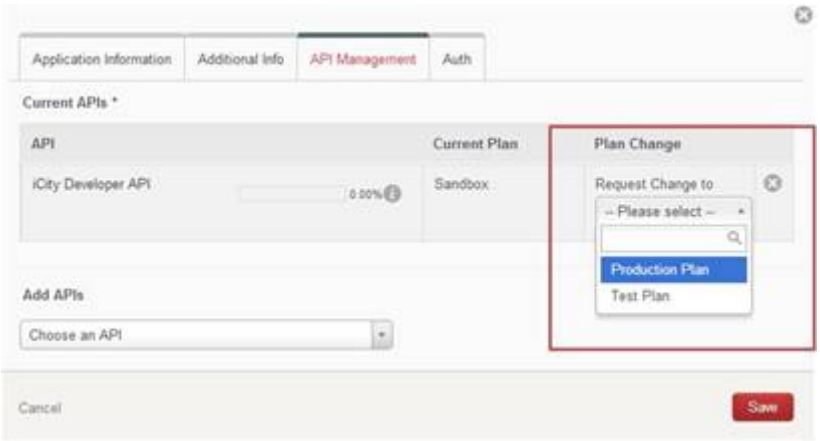

Figure 54 Application Plan Change

In the Information System Management Portal, for each application it's indicated the Information System involved and it is possible to select the status by choosing one of the options in the status tab. This status can be:

- Disabled: the applications doesn't have access to that Information System and for instance to that data feed.

- Preproduction: The application has been approved for a preproduction plan, so it has a restricted access with a concreted number of hits to the data feed.
- Production: The application has been approved for a Production Plan, so it has unlimited access to the Information System data feed.

Figure 55 Managing Application status

For more information about application plans see http://icity-devp.icityproject.com/documentation. See also APPENDIX VII: APP PROPOSAL VALIDATION.

Furthermore for each application it's possible to enable or disable the Usage validation by selecting the button in the application management console.

Figure 56 Application Usage Validation

As a result, this third year prototype offers an improved version of the iCity Platform which includes more functionalities ensuring the proper management of the iCity Platform, like for example:

- **FILTER**: this new functionality allows developers to reduce the number of calls to obtain certain data. We have done a cross-over data adding new parameters in the existing calls.

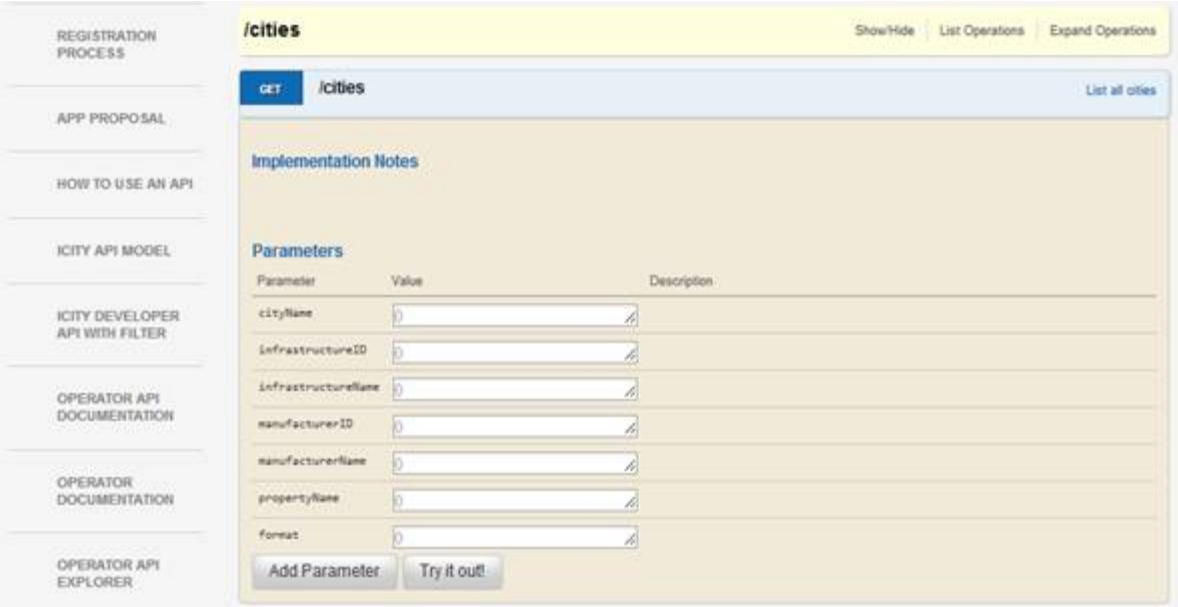

Figure 57 Filter Functionality

- **RADIUS:** this new call allows developers to get the information from all devices, services or events located in a specific place indicating latitude, longitude and distance (in radius mode).

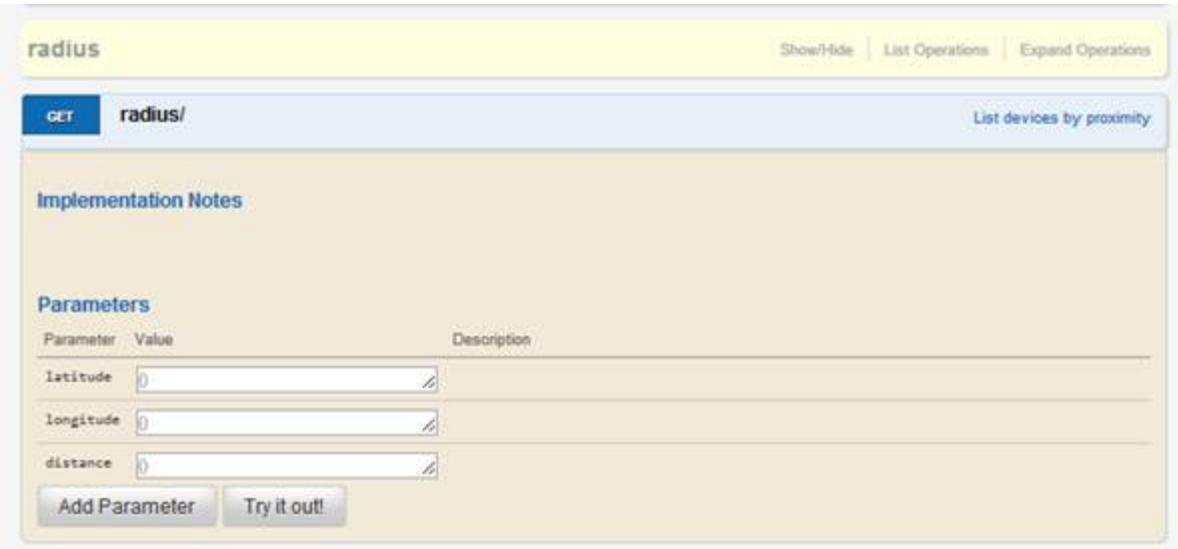

Figure 58 Radius Functionality

- **API in OPEN 311 standard format:** WP4 has implemented this functionality in order to have access to Suggestions & Complains cities information system like BCN (IRIS), LAMIA and Zaragoza.

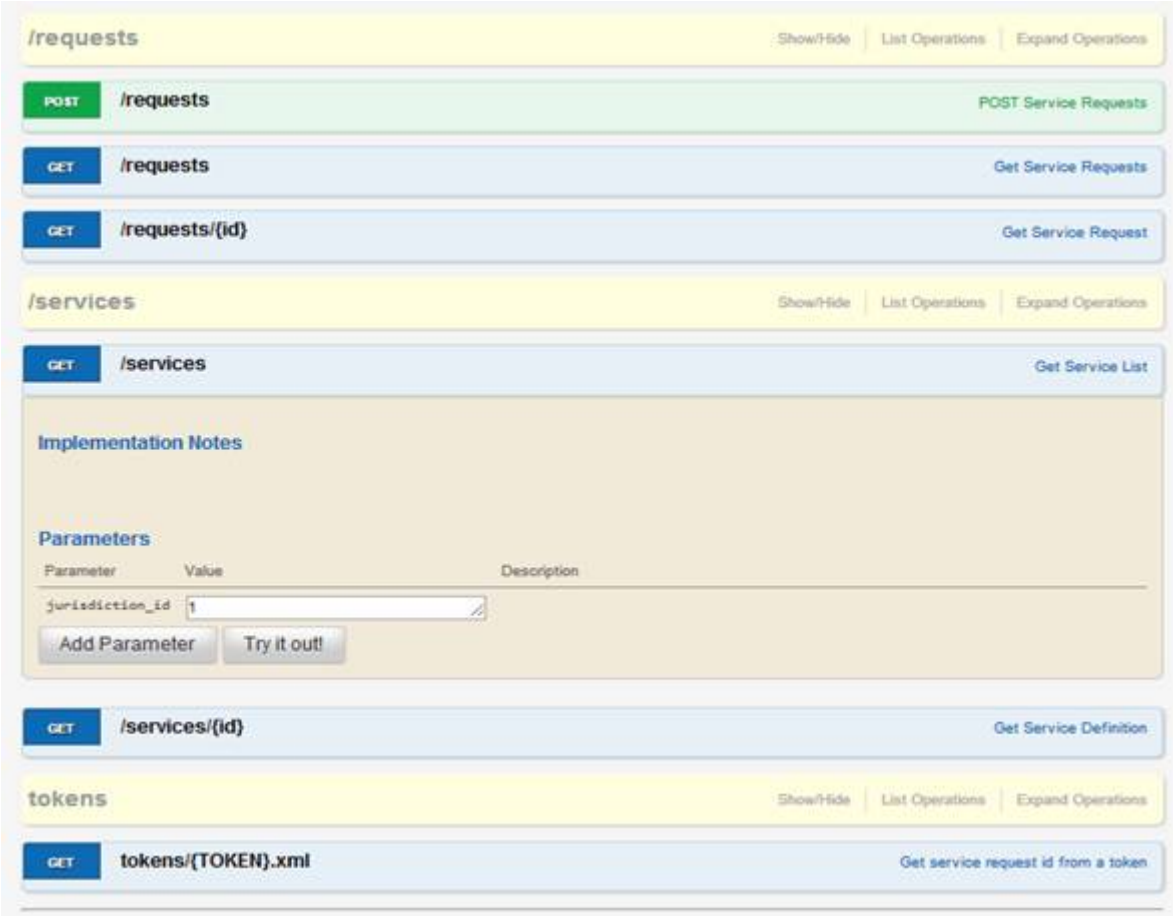

Figure 59 API Open 311

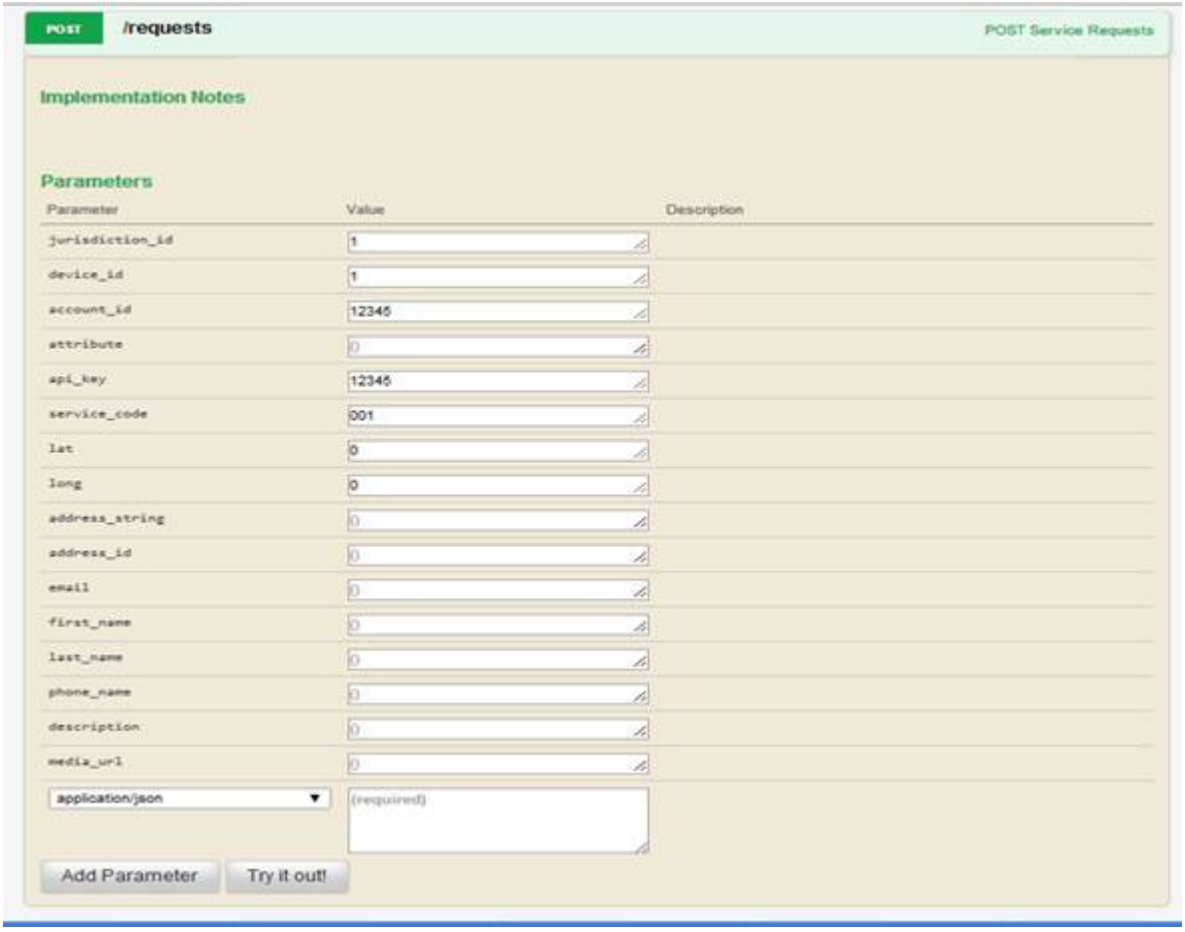

Figure 60 API Open 311 Query Detail

- **Developers Portal:** New Portal Home design, more related with public Portal.

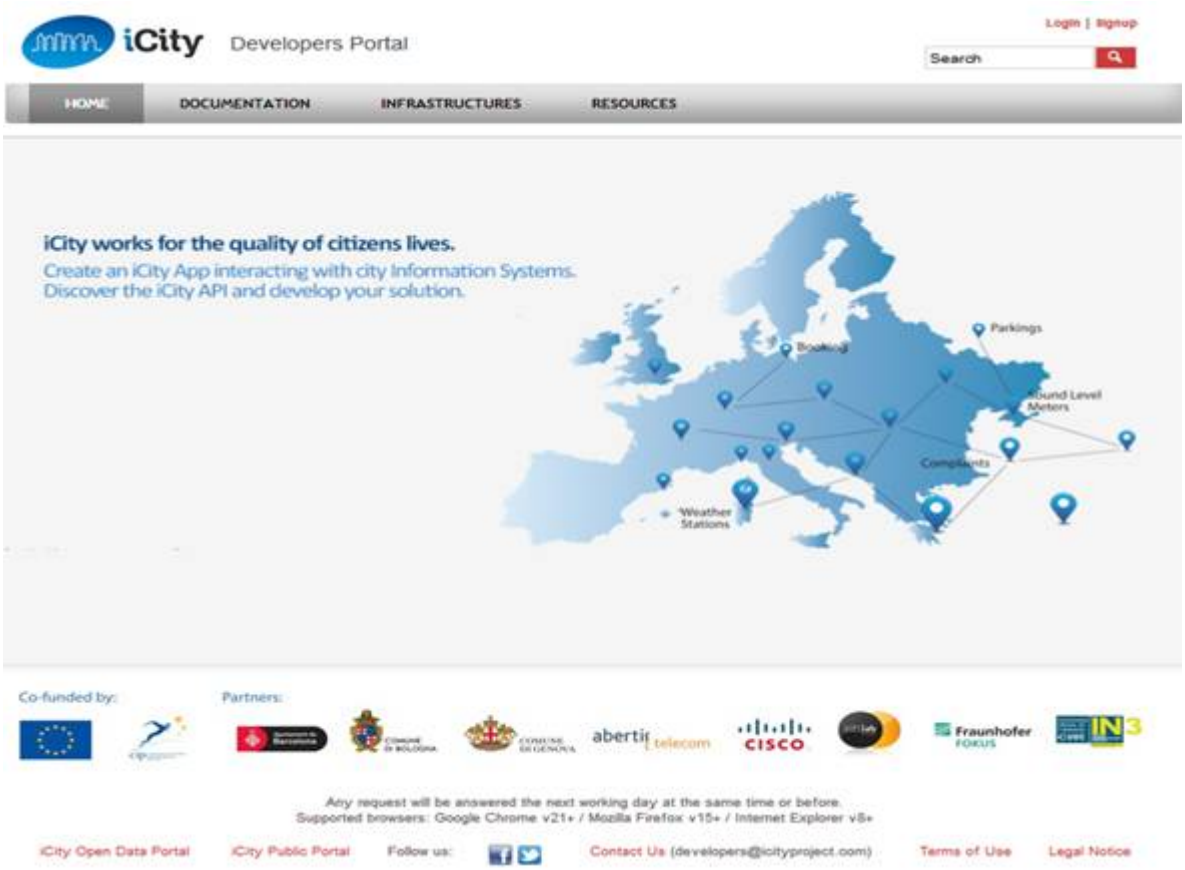

Figure 61 Developers Portal Home

- New **Developers Portal Documentation** area, with a clearer view of concepts.

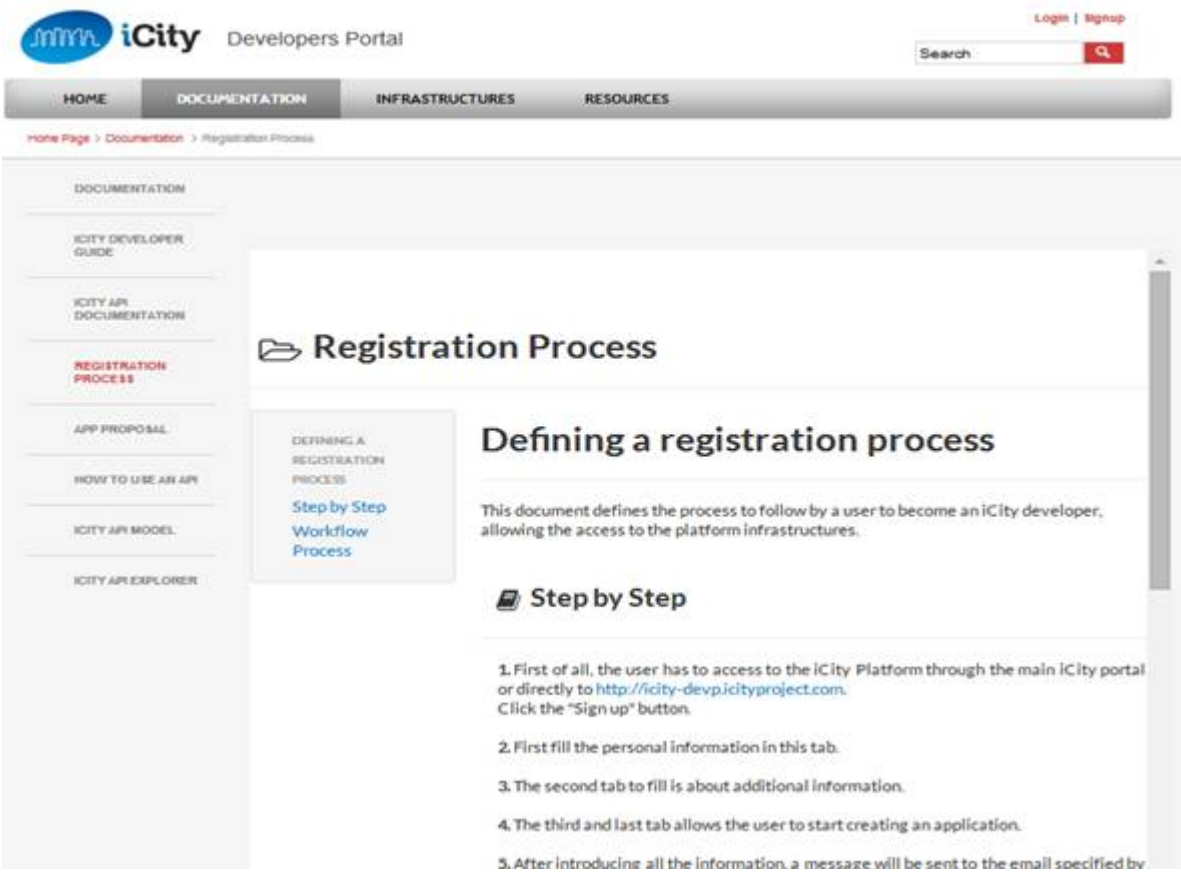

Figure 62 Developers Portal Documentation

In addition, it offers a tool for managing applications and granting access to the Information Systems that was point of concern for cities' managers that were concerned about controlling the access to their open Information Systems.

During the fourth year, from M36 to M42, WP4 have been focused on the integration of new information systems with the iCity platform and the adaptation of the iCity API and its documentation to their requirements. In addition, WP4 has been working in the iCity Platform improvements and its maintenance.

Moreover, WP4 was involved in the deployment and publication process for the iCity APPShowCase.

5.4.1 iCity System Operation

The aim of this section of the deliverable is to set up the second version of system operation prototype, which has the main goal to provide real time information concerning the cities Information Systems integrated in the iCity Platform. Also, the system management has to be able to work with different platforms and has to provide access to them identifying each one of the external authentication processes.

The first points of this section are mainly focused on the different integration systems operation, in order to look for the best operation tool, which covers the iCity requirements. The analysis presents existing solutions and also elaborates comparisons, in order to establish the best guidelines for deploying system operation adaptation to iCity platform. Regarding System operation, not only commercial solutions have been analysed. Thus also open source solutions have been taken into account.

In the next points of this section the different functionalities of the system operation are defined. Finally there is a description of the developments and status of the prototype at M42 (end of June 2015).

5.5 System Management Definition

Positioning vs. Existing solution

Nowadays, there are a huge set of monitoring systems, commercial solutions and open source solutions, and then the analysis covers the needs of iCity architecture defined in WP3 in the most optimal way.

Some of the more remarkable monitoring solution for such systems would be:

- Netcool
- HP NNM
- Pandora FMS
- Nagios
- ZenOSS

NETCOOL is a commercial solution based on Tivoli Netcool Omnibus, which gets alarms and events from different systems, provides a set of features to enrich these events and applies complex rules in order to reduce the amount of events that this type of systems receive. Licensing is established by collector and not for each device. This type of licensing is an important advantage than others, so the price of global solution is lower.

HP NNM is a commercial solution that provides a discovery of Information Systems, monitoring systems, network devices and another type of elements. This system solution provides a SNMP monitoring, which is passive monitoring using a traps reception or active monitoring using MIBS of devices. This system provides a threshold configuration that could be static or dynamic.

PANDORA FMS allows communication within elements through agents, which it is possible to analyse the status and performance of different parameters provided by sources. All communication is done via SSH, FTP, NFS or XML connector to transfer data that it is stored in a MySQL data base. Data is stored in a central server that allows showing in a web interface.

NAGIOS allows alerting message in a proactive way, so sends an email or SMS or instant message. Finally, this system has the enough intelligent to an event can create an automatic action to solve the incident active by an alarm before client or user notice about it. The environment of this solution is based on Linux and is composed about external plugins that can be programmed in bash or Perl giving a high flexibility.

ZENOSS is a monitoring solution based on Open Source Software for monitoring the availability of the components. This solution enables proactive monitoring of elements, thus allows detecting, reporting and resolving problems that affect the correct operation of services. It provides:

- ✓ Monitoring status and performance of the components involved in providing the service and the reception alarms from them.
- ✓ Events are presented and processed in a single console.
- ✓ Monitoring availability, basic inventory/configuration information and managed elements status is provided via SNMP.

As a result, Zenoss has been implemented for iCity platform prototype.

5.6 iCity Operation System Prototype

The operation System is one of the Platform subsystem, which main objective is facilitate the deployment, operation and maintenance of the iCity platform's network resources, urban data, applications and services, providing a global coverage and a more efficient End-To-End management.

This subsystem is based on a modular design which facilitates the individual evolution of each block minimally affecting to the rest of the components, according to the evolution needs and permitting the Platform's sustained future growth based in the addition of Software and Hardware provisions.

It has a robust, reliable, flexible, open, stable and highly scalable design architecture which allows the incorporation of new services and applications and the rising of information volume, devices, data sources, processes, etc., and manages them in an efficient way.

This subsystem will permit the supervision of all the Platform's elements, hardware and software, operating status and other elements belonging to the service, guaranteeing that its operation accomplish the established Service Level Agreement.

It will provide coverage to the operation system different functional modules supporting, at the same time, the different proceeds of incidents, problems, configuration, changes, versions, capacity and availability.

So, the Management System can be divided into the different sub modules:

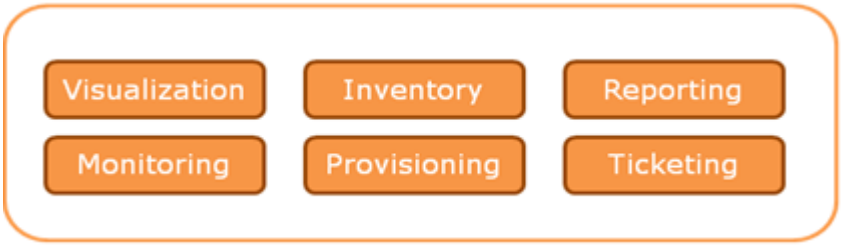

Figure 63 Operation System Prototype

- **Visualization** allows a graphical interface for the system's alerts visualization.
- **Monitoring** allows capture and collection of the data.
- **Inventory** allows a catalogue of elements.
- **Provisioning** defines processes and services associated to each type of element stored in the inventory.
- **Reporting** allows the reports and graphics creation, with the different monitored parameters.
- **Ticketing** provides coverage to the incidents management.

There are two proceeds which allow on the one hand, the sensors' data obtaining as of the SOS service (Configuration Process) and on the other hand, the alerts' obtaining as of the SAS service (Recollection Process).

Configuration Process

The main objective of this process is to save all the information of the Platform elements to enrich the alarms, which are sent to the system. The actualization will be made with two processes:

- GetCapabilities adquires the devices list.
- DescribeSensor obtains the individual information of each device.

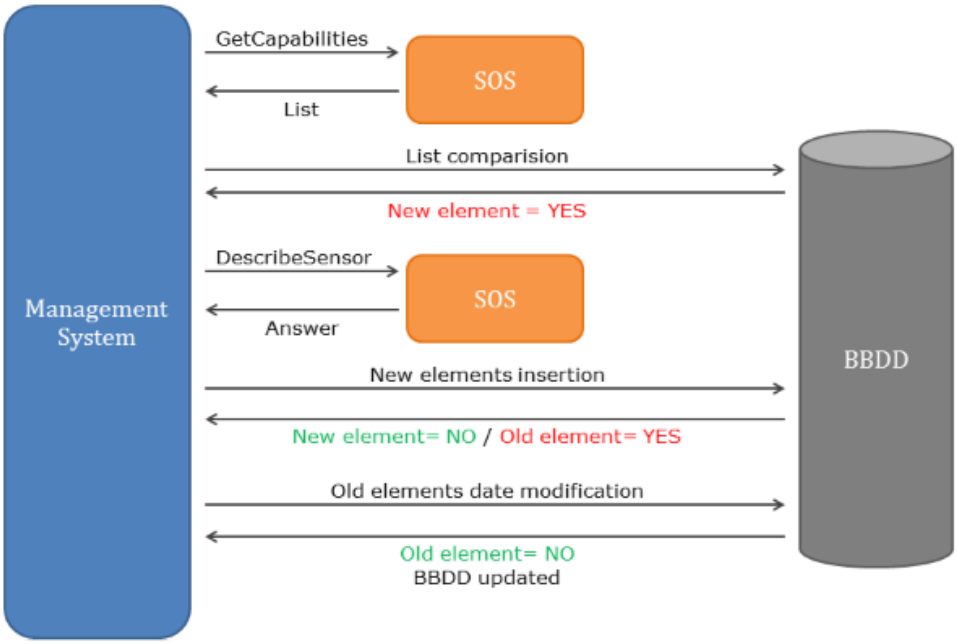

Figure 64 Configuration process

The process will check if the information of each element exists in the Data Base and if they are unsubscribed. Once is checked, it will perform the following operations:

- If the element is in the Data Base and its unregister date is null, it won't perform any operation, because the element will be registered in the system.
- If the element isn't in the Data Base, the Management System will make a *DescribeSensor* request to the SOS Webservice to obtain the necessary information and introduce it in the Data Base.
- If the element is in the Data Base but its unregister date is different from null, the Management System will make a *DescribeSensor* request to the SOS Webservice to obtain the needed information and introduce it in the Data Base. Its unregister date will be updated.

Recollection Process

Once the Data Base information has been updated, the Configuration Process calls the Recollection Process to perform the necessary operation: addAlarmObserver or cancelAlarmOberver.

In the addAlarmObserver process, the collector makes a request to the SAS Webservice with the parameters of the elements that we want to observe. The SAS will return the XMPP channel identifiers where will be the system produced alarms, to enable the subscription to each channel.

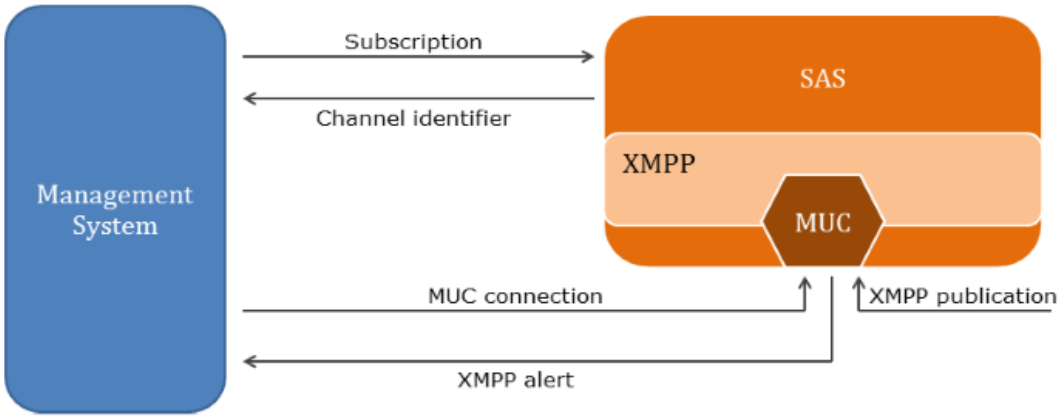

Figure 65 Recollection process

Periodically (configurable time), the process will check the Data Base modifications and will make the necessary management to update the alarm channels of each sensor which its subscribed, doing the following checks:

- If the register date is higher than the last process execution date, the system will make the register of the sensor alarm channel with a *Subscribe* request to the SAS.
- If the unregister date is higher than the last process execution date, the system will make the cancellation of the sensor alarm channel subscription with a *CancelSubscription* request to the SAS.
- In the rest of the cases, the subscription will be renewed with a *RenewSubscription* to the SAS.

When a reception alarm is detected in a subscription channel, this component process, enrich, correlate and send it to the visualization layer.

5.6.1 Visualization

The objective of this module is to visualize the different Platform's elements events: the events detected as a consequence of the monitoring, the events generated by the network devices (SNMP traps) and the events of the Management System.

The visualization/operation module will allow the proactive supervision of the state of all the Platform's devices and components, network interfaces managed, services/processes and Hardware equipment (memory, CPU…). It will be available to show the different devices throughput in a graphical way.

This sub module will have the capacity of generate events considering the established threshold and against an anomalous behaviour, which will be a result of the information analysis during a period of time and its variation regarding a pattern or a previous condition.

In that way, the operator will be able to check the state of the elements in real time, permitting a faster incident detection and resolution of problems in case the elements are affected by some type of failure.

In the picture bellow we can see, as an example, the events console of the monitoring tools with a possible information structure for each event:

Severity	Device	Component	Event Class	Summary	First Seen	Last Seen	Count
...							
	SMARTCITY-SRV6		/Status/Ping	ip 10.1.24.13 is down	08-01 13:28:38	08-13 04:15:50	1644
	SMARTCITY-SRV4	Sos - Servi...	/Status/WinServic...	Windows service 'Sos - Servicio de CaptaciÃ¤Â³n de Datos' is stopped	08-11 00:03:21	08-11 02:48:22	34
	SMARTCITY-SRV6	Applicatio...	/Status/WinServic...	Windows service 'Application Information' is stopped	08-01 11:18:16	08-01 13:27:08	27
	SMARTCITY-SRV6	zeneeventlog	/Status/Wmi	Could not read the Windows event log (NT_STATUS_CONNECTION_REFUSED). Check your username/password setti	08-01 11:18:14	08-01 11:18:14	1
	SMARTCITY-SRV4	Instalador ...	/Status/WinServic...	Windows service 'Instalador de mÃ¤Â³dulos de Windows' is stopped	08-01 10:52:36	08-01 10:52:36	1
	SMARTCITY-SRV6	zeneventlog	/Status/Wmi	Could not read the Windows event log (NT_STATUS_HOST_UNREACHABLE). Check your username/password setting	07-25 12:50:43	08-01 08:30:38	96
	SMARTCITY_SRV	zenmodeler	/Cmd/Fail	User timeout caused connection failure.	07-31 22:51:48	07-31 22:51:48	1
	localhost	/Net/Link/i...	/	Error processing transform/mapping on Event Class /Net/Link/instances/snmp_linkUp	07-09 09:09:15	07-31 12:57:47	59
	SMARTCITY-SRV6	Windows ...	/Status/WinServic...	Windows service 'Windows Modules Installer' is stopped	07-31 10:07:09	07-31 10:52:07	10
	SMARTCITY-BDD	Applicatio...	/Status/WinServic...	Windows service 'Application Experience' is stopped	07-25 15:32:05	07-25 22:52:09	89
	SMARTCITY-BDD	zeneeventlog	/Status/Wmi	Could not read the Windows event log (NT_STATUS_HOST_UNREACHABLE). Check your username/password setting	07-25 11:27:13	07-25 11:29:13	3
	SMARTCITY-SRV4	Host de pr...	/Status/WinServic...	Windows service 'Host de proveedor de detecciÃ¤Â³n de funciÃ¤Â³n' is stopped	07-24 14:37:03	07-24 22:52:01	100
	SMARTCITY-SRV4	zeneeventlog	/Status/Wmi	Could not read the Windows event log (NT_STATUS_IO_TIMEOUT). Check your username/password settings and	07-22 01:56:32	07-24 15:16:46	2
	SMARTCITY-SRV4	zeneeventlog	/Status/Wmi	Could not read the Windows event log (NT_STATUS_CONNECTION_REFUSED). Check your username/password setti	07-24 14:39:33	07-24 14:39:33	1
	SMARTCITY-SRV4	zeneeventlog	/Status/Wmi	Could not read the Windows event log (NT_STATUS_HOST_UNREACHABLE). Check your username/password setting	07-24 13:53:43	07-24 14:30:38	6
	SMARTCITY-SRV6	WinHTTP ...	/Status/WinServic...	Windows service 'WinHTTP Web Proxy Auto-Discovery Service' is stopped	07-24 07:16:59	07-24 10:58:03	45
	SMARTCITY_SRV	zenmodeler	/Cmd/Fail	An error occurred while connecting: 113: No route to host.	07-19 10:54:54	07-19 10:54:54	1
	bcn3020col3	zenmodeler	/Cmd/Fail	TCP connection timed out: 110: Connection timed out.	07-02 14:17:31	07-11 02:22:04	3
	10.16.231.1	zenmodeler	/Cmd/Fail	An error occurred while connecting: [Failure instance: Traceback (failure with no frames):	06-21 02:15:04	07-11 02:17:40	3
	bcn3020col3	telnet	/Status/IpService	IP Service telnet is down	07-02 11:51:01	07-02 11:51:01	1

Figure 66 Event console

- Severity: critic level of the event.
- Device: one higher hierarchical level's device above the element which produced the alarm.
- Component: element which produced the alarm.
- Event class: indicator which produced the alarm.
- Summary: alarm description (completely configurable).
- First seen: first time that the alarm was detected.
- Last seen: last time that the alarm was detected.
- Count: number of times that the system has registered the alarm. When an alarm arrives with the same fields as another one that has been previously detected, the counter increases, to avoid the alarms reduplication.

From the tool interface, the operator of iCity platform will be able to see the complete information that the alarm system has got and the different elements of the Platform, for example, their location or IP direction.

Figure 67 General information related an alarm event

The received elements will be stored in a Data Base, which will have got, at least, the following variables:

- Active events: it will contain a list of all the events associated to the active Platform elements, updating in real time. In that way, the operator will be able to identify them in a fast and visual way.
- Historical: it will contain solved past elements. The operator will be able to establish the rules that he considers to pass to the historical active events automatically.
- Details: it will contain the events defined by the operator. The administrator will be able to create new alarms, remove existent alarms and export showed alarms to different formats.

This module will allow to definite automatic actions that will execute when the system receive certain elements and the SNMP, Telnet, SSH or WMI access credentials to the equipment in their different versions.

From this interface, the administrator will be able to modify and remove existent users and create new users, allocating different roles which will permit or limit the access and execution of determined options.

5.6.2 Monitoring

The monitoring module will facilitate the Platform elements monitoring and supervision (collectors, sensors…), network Information System (switches, routers…), collect and store data services, CRM, portal, etc.

This module will contain the following submodules:

- Data capture and collection.
- Monitoring.

Data capture and collection

This submodule will provide the access to the configuration parameters of the capture functionalities. To speed up the capture configuration of homogeneous devices, the system will allow the use of templates, which will group all the device typology relevant parameters. A device can fit with multiple typologies simultaneously, so it can hold to multiple templates.

It will allow copying one of the templates to modify it after, and will give support to the deployment of the new template to all the existent and linked devices. The operators will be able to import and export to XML format all the available templates.

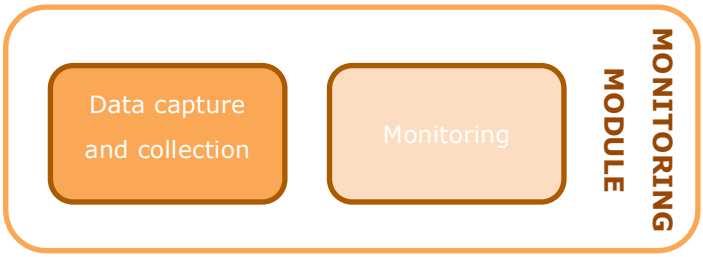

Figure 68 Monitoring module – Data capture and collection

It will have available the following capture functionalities:

- SNMP: capture the devices information with SNMP protocol (SNMP get), SNMP traps and its associated management.
- Poll: specific agents' interrogation in a centralized way.
- IPMI: interrogation of the agents which use the IPMI (Intelligent Platform Management Interface) Protocol.
- Remote services: monitoring of network services published without agent (ping, FTP, HTTP, etc.).
- Web: realization of web pages sequences following the POST and GET events, associated to guarantee the complete availability of themselves, the response times and the expected results.
- Passive: capture of the directories files that the server verifies periodically.

- Log: specific expressions and text sequences (in log files) capture with regular expressions.
- Proxy: system data capture with a group of intermediate servers which gather data from a subgroup of devices and communicate them to the main server.

This submodule will recognize the devices situated in the supervised Information System, will link them automatically to their respective device typology and will start the monitoring.

Monitoring

This module will trigger alarms when the conditions based on the value of the equipment parameters get accomplished, triggering alarms to the operators when the alarms reach determined levels of criticality, offering the possibility of envy with multiple messaging mechanisms (e-mails, SMS, etc.).

It will permit the configuration of the trigger conditions to adapt it to the operator needs and will have an expression definition and value calculating language to define complex alarms.

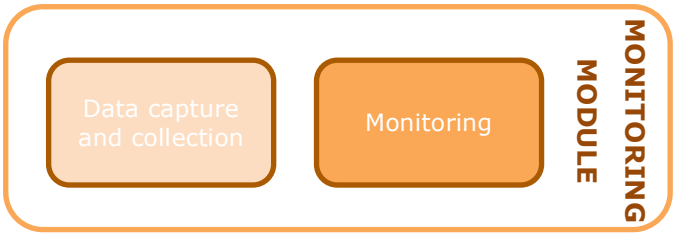

Figure 69 Monitoring module - Monitoring

This submodule will allow visualizing and managing the hierarchy of the alarms associated to different devices and centres and indicators and the gravity or criticality of the alarms depending on multiple parameters (number of similar alarms, alarm duration, number or repetitions of the same alarm in a determined period of time, etc.).

The operator will be able to do the pertinent configurations to make the system take actions automatically against determined alert situations, with local or remote scripts.

This submodule will be the responsible to inform about the disponibility of the monitored Information System failure and its impact on the client services. Having the monitoring information, it can be generated the SLA's (Service Level Agreement) statistics.

5.6.3 Inventory

This module will permit the inventory of the different logical and physical elements (sensors, collectors, transport network Information System, etc.) and the data of each Platform equipment.

In addition, this module will support: service provisioning, operating and management and physical elements life cycle.

Conceptually and, with independency of the network, is proposed the following modelling structure (it can be modified): each device and connection is treated as a typology instance (N1), specified for a family (N2) and a component (N3). In exchange, for the services it's needed just one modelling (N2).

The inventory module will include the following submodules:

- Management consultation.
- Documentation management.
- Network and services management.
- Catalogue management.
- Information quality.

5.6.4 Documentation management

The aim of this submodule is to centralize data inventory of equipment and services, avoiding maintenance of inventory information across different systems, therefore there is only one maintained inventory for the iCity platform.

This module manages the different states of the entities in the documentation module (as planned, build, in service, etc.). Also, it provides historical documents in order to search information in an easy way. Historical save any changes to the configuration and status of equipment, network or services of iCity platform.

Access to external users will be restricted, so that they shall be allowed access to the devices on which the user is responsible for maintenance, so they can document their own networks but they do not have access to other elements of the platform.

5.6.5 Network and service management

This submodule has sufficient management capacity to create, configure and maintain the network and tools that increase efficiency and effectiveness in order to reduce the time for provisioning tasks.

Network and service management allows the representation and creation of networks through different types of elements (nodes, connections, networks, etc.). Also, it provides a functional view of all resources, thereby facilitating the automatic design of circuits and networks.

This submodule collects and stores information about changes in the network resource in order to allow visualization in multiple forms of network planning. In this way, operator can anticipate future capacity needs by extrapolation algorithms relying on the historical.

5.6.6 Catalogue management

This module will provide the submodule ability to evolve and adapt through a master set and customize the types of networks and services in order to assign mandatory attributes, optional attributes, constant values and ranges.

Catalogue management allows management of the structure of services, equipment, connections, routes and cards, that it offers functionalities for managing entities (cards, equipment, connections, groups, routes, and services) and to establish relationships between these groups and the inventory.

This module is aligned with TMN standard (Telecommunications Management Network) and with SID standard (Shared Information Data Model).

5.7 Operation System Adaptation M42

In the first year prototype, a preliminary system was included in the prototype in order to allow, in a short time, the deployment of pilots and also the development of apps, with an operation system that would manage real time information.

Operation system offers the following benefits:

- Openness allows the integration of the whole system is based on universal platform.
- Adaptability provides confront the swelling and much more complex services.
- Expansibility allows the adaptation to future complex situation.
- Flexible provides an easy for correcting, maintenance, update and upgrade.
- Friendly user interface.

Over the last three years, the prototype has included the inputs coming from other WPs as WP3 and WP5. Due to those inputs WP4 has worked in an operation model to follow in the provision and integration of new Information Systems in the iCity Platform.

5.7.1 Integration of new Information Systems

The aim of iCity's project is to integrate as many Information Systems as possible and also to offer developers an attractive source of data that will provide information coming from these Information Systems, ensuring the security and the properly use of them.

In order to achieve this objective, iCity Project needs information related to these Information Systems. To obtain this information WP4 defined a table to be completed by cities once an Information System is open to facilitate its integration. This table includes all the information needed to integer in iCity Platform including the policies & rules of use.

So the first step to integrate one Information System is to work hand by hand with cities to obtain as much as possible information in order to integrate and the define how the access to the data will be done.

General Features

City	*City*
Name	*Information System's Name.*
Description	*A description of the Information System including some details.* *It is also recommended to include the following information:* • *Why have we chosen this Information System?* • *Does it answer to an application's need?*
About Communications	
Technical Contact	*It's necessary to have a technical contact to solve issues related to communication between the iCity platform and the proposed Information System:* • *Name* • *Telephone Number* • *e-mail* • *Preferred communication channel*
Physical Connection	*Detail on how to physically connect both Information Systems. (iCity and the proposed one)*
Logical Connection	*Detail on how to establish a logical connection. (VPN, Telnet, Socket ,….) It is possible that this connection is directly linked to previous section.*
Integration	*Detail on how to access to information.*
About Use	
Detailed Primary Functions	*Information about API or commands. Please attach the*

	document or link. Information required: (if available) • *Detailed functions list and its parameters.* • *Information obtained in each case* • *Errors* • *Data types*
Functionalities	Owners are encouraged to propose the main functionalities that have to be transferred to the service layer (application layer). *iCity platform will use the original functions to execute the new ones.*
Polices	*Define polices and rules in order to use correctly the Information Systems.*
Planning	
Dates	*Provide a delivery date for each previous item*

The second step is to study the method for accessing the Information System. Usually each platform has its own API or method for accessing to the data, so it's important to study the documentation facilitated and to ask for the credentials in case is needed. After its clear how it works, the following step is to integrate and ask for help to the cities with the issues occurred during the process of integration.

Before publishing the access to the Information System in the iCity developers Portal we have checked and tested the access to the data. Once the Information Systems opened by cities are accessible the information about the data available and how to access to it is available in the iCity Public Portal (www.icityproject.eu) and iCity Developers portal (http://icity-devp.icityproject.com/documentation/INFORMATION SYSTEMS).

During the project, WP4 has integrated in the iCity Platform the following Information Systems Opened by cities:

LONDON:

The Information Systems integrated in the iCity platform by the City of London are mainly related with the environmental data and mobility for public transportation services (TFL).

- **Air quality sensors** (King's College London Environmental Research Group of Strand (http://www.londonair.org.uk/LondonAir/guide/default.aspx) Offers information about environmental air quality sensors placed in the city of London.

- **TFL Journey Planner** Offers information about the best route to arrive to a defined place by introducing some parameters as well as the starting point.

- **Alert Me** is a Smart Home platform. In this case, we will have an instance of "AlertMe Smart Energy" (see www.alertme.com) in the home of each triallist based in London.

BOLOGNA:

The Information Systems integrated in the iCity platform by the City of Bologna are mainly related with the Town's mobility management system and public transportation services. Here is the current list:

- The **"TPER — QueryHellobus"** service is the City's public transportation arrival time information management system. Offers information about the expected arrival time for a bus introducing the stop and the line.

- The **"QueryHellobus- 4ivr"** service is the City's public transportation arrival time information management system. . Offers information about the expected arrival time for a bus introducing the stop and the line for IVR services format.

- The **"QueryResale"** service is the City's public transportation ticket sales point. provides the list of resellers of bus tickets allocated in the nearby of a specific bus stop.

- The **"CISIUM – Metropolitan traffic"** service provides the measurement of traffic on the main metropolitan streets. Data are sampled every 5 minutes.

- The **"CISIUM – Events"** service provides the list of events, such as accidents and workings, that may have an impact on mobility.

- The **"CISIUM – Parking"** is related to the parking areas of the city. It offers the list of available parkings and the number of free spots and also additional information like opening hours and telephone number.

- The "**CINETECA – Catalogue of DVDs and VHS"** offers the possibility to query the database of their DVDs and VHSs available for viewing and/or loaning.

- The "**CINETECA – Events"** allows to query Cineteca di Bologna's projections and events programme.

- The **"Air Quality"** allows developing applications that shows the trend of air quality along the time for the available locations.

- The **"Wifi Location and Live Monitoring"** allows both location and live monitoring of the number of accesses throughout the whole Iperbole Wireless Network.

GENOA

The Information Systems integrated in the iCity platform by the City of Genoa are mainly related with environmental and Citizens Information. This is the list of integrated Information Systems:

- **"Genoa Citizen's Desk":** This Information System is mainly based on the information stored on a database and managed through web and mobile applications. Through this system citizens may request information about department or work processes, receive documentation or forms by mail or fax, check the opening hours of the offices. There is also information about tourist and cultural points of interest, or security and public health structures (police stations, hospitals, embassies, etc.).

  The system is managed and used by various offices spread on the municipal territory but it will be expanded and will also supply information of other surrounding areas in an integrated way. The structure is already designed for distributed gathering of information from different sources.

  The information offered to developers is

- List of items and offices available
- Request the last available data from one or more items and/or office.

- **"Genoa Weather Stations"**: The city of Genoa has a network of weather stations that provide information about temperature, humidity and wind speed from many providers. These Information Systems allow developing applications that show real time information about local weather. These data are also used by citizens but it's also among the information used by its local civil protection.

- **"Traffic Webcam System":** is composed by about 25 Webcams, located on the municipality territory. They have a resolution equal to 352 × 288. This Information System will be integrated with new webcams during the next years. This Information System offers images obtained from these traffic cameras. The information provided by this Information System are images obtained from traffic cameras located around the municipality.

- **"Wifi Hotspots":** FreeWiFiGenova is the name of the project of the Municipality of Genoa for the free internet navigation via wi-fi network. Among the objectives of the service there is an increased accessibility of information for citizens and tourists. The service is available in the main city squares, libraries and museums with over 130 hot-spots that allow free navigation for 300 MB per day (no time limits) on the Internet.

- **"Toursim Webcams":** The Tourism Webcam System is composed by a few Webcams, located on the municipality territory. They have a resolution equal to 1024 × 768. This Information System makes accessible to developers the list of webcams data available in Genova.

- **"Air Sensors":** In our city and Province, we have a network of air sensors that provide information about various data regarding different pollutants. These data are reported in conjuction with the Province of Genova.

BARCELONA

The Information Systems integrated in the iCity platform by the City of Barcelona are mainly related with environmental and weather data. Barcelona has also opened an Information

System for citizens' complaints and incidences. The list of integrated Information Systems is the following:

- **Sentilo:** Barcelona City Council offers a platform to access to sensors data which are distributed around the city. BSP includes data coming from these kind of sensors:
    - Environmental sensors (temperature, NO2, CO2, noise).
    - Sustainability (level of capacity of the container waste).
    - Traffic management (parking sensors).
    - Walkers flows (number of pedestrian).
    - Irrigation control (ground humidity, wind, rain, temperature).
- **Smart Citizen Platform** (http://www.smartcitizen.me/): This Information System is a platform which purpose is to generate participatory processes of people in the cities. Connecting data, people and knowledge, the objective of the platform is to serve as a node for building productive and open indicators, and distributed tools, and thereafter the collective construction of the city for its own inhabitants.

  The Smart Citizen project is based on geolocation, Internet and free hardware and software for data collection and sharing, and (in a second phase) the production of objects; it connects people with their environment and their city to create more effective and optimized relationships between resources, technology, communities, services and events in the urban environment. Currently it is being deployed as initial phase in Barcelona city.

- **IRIS**: Barcelona City Council offers different attention channels aimed to citizens (telematics, telephonic and face-to-face channel) with the purpose to allow citizens communicating incidences, complaints and suggestions about municipal services or city functioning. Furthermore, it is possible to consult the petition status by means of the three possible channels as well as claim it.

  To ensure the fastest resolution of each request, it is essential to classify correctly the requests.

iCity offers access to the complaints IRIS Service through iCity API. Developers could build applications to insert complaints as well as consult the status of its incidence.

- **AGENDA**: Agenda Barcelona contains the leisure activities taking place in the city, with information about dates, places and prices. This information system contains very useful information that can provide a richest experience of the city to locals and tourists.

- **FACILITIES**: Facilities of Barcelona contain the inventory of facilities of the city, including public and private, that offer services to the citizens and tourists. This list includes information about hospitals, hotels, sport centres, taxi stations, etc.This information system contains very useful information that can provide a richest experience of the city to locals and tourists.

CORNELLÀ

- **Agenda**: The agenda is a webservice that offers very valuable information about events of Cornellà for its citizens and visitors. The Cornellà de Llobregat City council has developed this service to promote third parties to use this daily information in their apps. There's already an app that uses it.

LAMIA

- **Issue Reporting**: Issue reporting is a basic need to every local community. Citizens need to interact with authorities not only using phone but using mobiles, tablets, etc. Citizens can submit issues regarding city's Information System such as potholes, graffiti removal, etc.

ZARAGOZA

- **Suggestions & Complaints**: Complaints and suggestions, which according to different administrative categories, citizens send to the Zaragoza City Council authorizing their publication. More information about Complaints and Suggestions can be found in: http://www.zaragoza.es/ciudad/risp/open311.html

ABERTIS TELECOM

- **Smart Zone – Urbiotica Sensors**: Urbiotica offers a platform to access to sensors data distributed in Abertis Smartzone nearby Abertis offices. Urbiótica offers an open API based on Representational State Transfer interfaces type (REST). REST is a style of architecture that exploits existing technologies and protocols of the World Wide Web (WWW).

- **Smart Zone – Parkare Sensors**: Parkare offers an open API based on SOAP Web services. The Parkare will be through communication with the HTTP (Hypertext Transfer Protocol). Parkare includes information obtained from parking meter like:

    - Transaction tariff
    - Car Plate Number
    - Total paid cash
    - Total paid card
    - Total paid time
    - Ticket Number

The prevision for the end 4th year WP4 will work on the integration of the following Open Information Systems although we expect to integrate some more pending to be defined:

City	Information System
Barcelona	Wifi Network – MSE Service
Bologna	Geocoding
Bologna	Agenda Cultural
Genoa	Geographical Information
Genoa	Hydrometers
Genoa	Toponyms

5.7.2 Partner's proposals Portal

As a result of the inputs received during this period from pilots and cities, it has been created a Governance structure document which contains the requested information to the cities like provisioning requirements, policies and rules, etc, for enabling the access to each Open Information System. Responding to the needs of the Governance workflow, it is being developed a portal for managing new proposals for changes, from now called Partner's Proposals' Portal.

Only partners will have access to this portal by login with the use of credentials. The Portal will have some sections. Once the partner accesses to the Portal they will see a list of existing Proposals and see the information related. For example the name or some details and the status of the proposal.

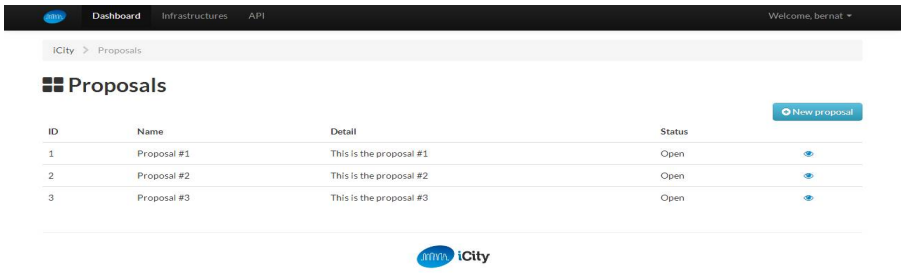

Figure 70 Partner's Proposals' Portal List

It is also possible to see more details about one of those Proposals by accessing to the Proposal's Profile.

- Proposal Information:
    - ID: identification number of the proposal
    - Name: Name of the proposal
    - Type: Defined Type of proposal (New Future/Task/Improvement/Change)
    - Detail: Brief description of the proposal
    - Status: Open/ In Progress/ Reopened/ Solved/ Closed

- Creation date: Date when the proposal was created

- Status History: Displays information about the changes that have been made throughout the lifecycle of the proposal as well as the partners involved in the changes.

- Comments: Displays the comments done by the partners involved in the changes.

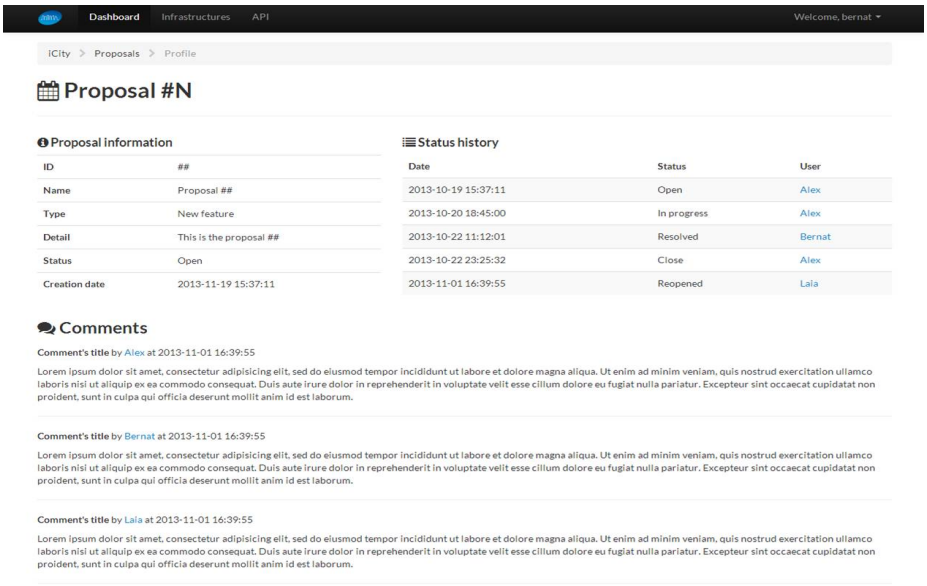

Figure 71 Partner's proposals' Portal Info

To add a new proposal for change they should complete some fields. Every proposal for change has to include the following points:

- Title of the Proposal
- Type of Proposal (select): New Future/Task/Improvement/Change
- Proposal Details (combo)
- Comments (combo): including Date and User
- Status (checkbox):Open/ In Progress/ Reopened/ Solved/ Closed including Date and User
- Image: to upload an image in format .jpg, jpeg or gif which cannot exceed 500KBytes.

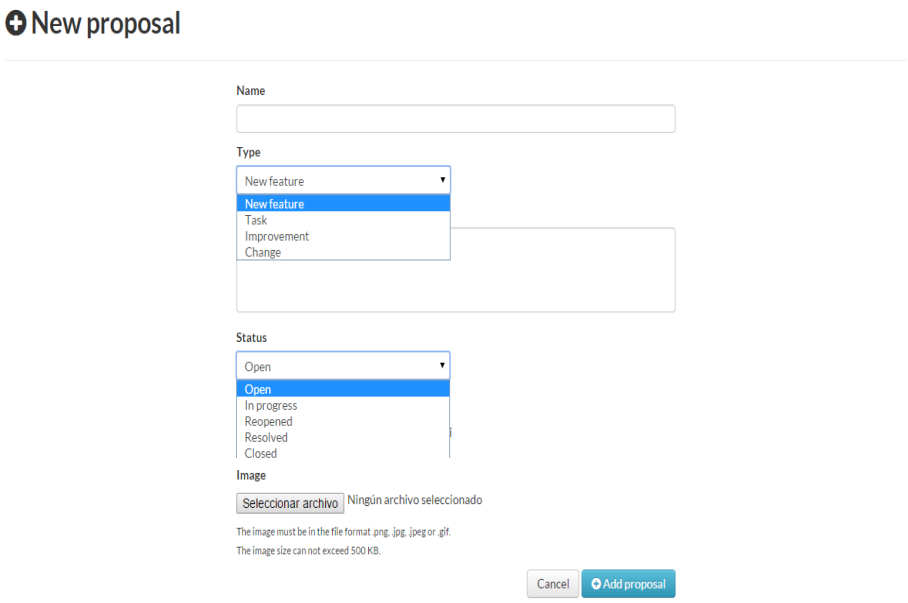

Figure 72 Partner's Proposals' Portal new proposal

6. Data Warehouse & Business Intelligence

This section reports the work achieve in WP4 regarding Data Warehouse adaptation and Business Intelligence.

The aim of this section is to set up the Data Warehouse prototype, which has the main goal to homogenize the storage data, although data is coming from heterogeneous open Information Systems. Besides, Data Warehouse will allow future deployments of Business Intelligence, based on WP3 design requirements, WP5 pilots' needs and WP7 exploitation models definition.

This activity is alive until the end of the project and the DWH & BI prototype will be enriched following the inputs coming from WP3, WP5 and WP7.

The first points of these sections are mainly focused on the state of the art related to Data Warehouse and Business Intelligence solutions, in order to establish the implementation guidelines for both topics, covering the iCity requirements.

Thanks to the inputs received from WP3 and the pilots done in WP5 in the last points of this section it's explained the decision taken related Data Warehouse & Business Intelligence. The experience acquired during this period has proven that there's no need to have Business Intelligence module or any Data warehouse although cities may have one to store its information.

6.1 Overview

First part of the deliverable is an analysis of data warehouse and business intelligence solutions based on the state of the art. The analysis presents existing solutions and also elaborates comparisons, in order to establish the best guidelines for deploying data warehouse and BI to iCity platform.

Regarding BI, not only commercial solutions have been analysed. Thus also open source solutions have been taken into account. Regarding DWH, as a large amount of data is expected to be managed by iCity platform, the analysis presents a discussion about the use of relational or non-relational data base and the management of video content.

Second part of the deliverable is focused on the description of the data warehouse adaptation to iCity prototype.

6.2 Analysis of DWH & BI

6.2.1 Existing Solutions

In this chapter, we should explain the following items:

- BI Analysis Software for Enterprises.
- BI Analysis Software for SME.
- Analysis of advantages and disadvantage of implantation an Open Source vs. Commercial Product.

The main concepts in order to explain BI existing solutions are listed below[5].

6.2.2 ETL: Extract, Transform and Load

Extract, Transform and Load (ETL) refers to a process in database usage and especially in data warehousing that involves:

1) Extracting data from outside sources
2) Transforming it to fit operational needs (which can include quality levels)
3) Loading it into the end target (database, more specifically).

[5] Most of the information in this section is taken from Wikipedia.

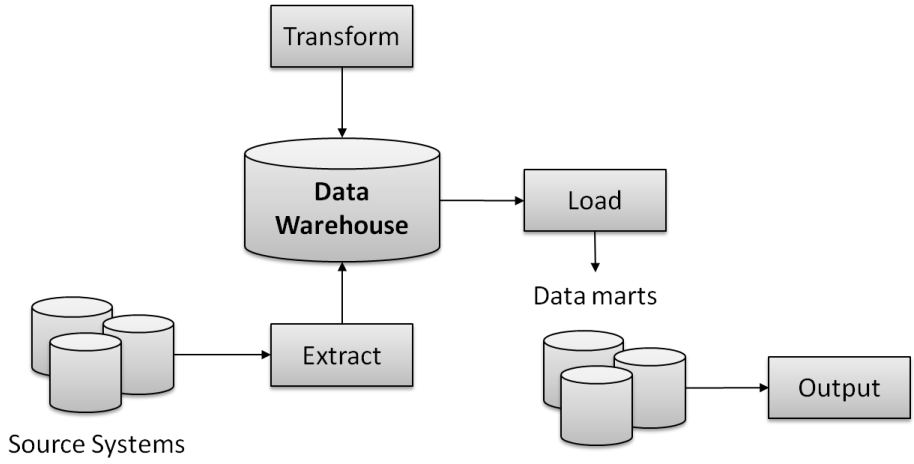

Figure 73 ETL

Extract

The first part of an ETL process involves extracting the data from the source systems. In many cases this is the most challenging aspect of ETL because data comes from different source systems, different formats, different structures, different storages, etc., in fact extracting data correctly will set the stage for how subsequent processes will go.

Transform

The transform stage applies a series of rules or functions to the extracted data from the source to derive the data for loading into the end target.

Load

The load phase loads the data into the end target, usually the data warehouse (DW). Depending on the requirements of the organization, this process varies widely. Some data warehouses may overwrite existing information with cumulative information; frequently updating extract data is done on daily, weekly or monthly basis. Other DW (or even other parts of the same DW) may add new data in a historical form, for example, hourly. To understand this, consider a DW that is required to maintain sales records of the last year. Then, the DW will overwrite any data that is older than a year with newer data. However, the entry of data for any one year window will be made in a historical manner. The timing and scope to replace or append are strategic design choices dependent on the time available and the business needs.

More complex systems can maintain a history and audit trail of all changes to the data loaded in the DW.

As the load phase interacts with a database, the constraints defined in the database schema — as well as in triggers activated upon data load — apply (for example, uniqueness, referential integrity, mandatory fields), which also contribute to the overall data quality performance of the ETL process.

For example, a financial institution might have information on a customer in several departments and each department might have that customer's information listed in a different way. The membership department might list the customer by name, whereas the accounting department might list the customer by number. ETL can bundle all this data and consolidate it into a uniform presentation, such as for storing in a database or data warehouse.

Another way that companies use ETL is to move information to another application permanently. For instance, the new application might use another database vendor and most likely a very different database schema. ETL can be used to transform the data into a format suitable for the new application to use.

6.2.3 Data Warehouse

In computing, a data warehouse or enterprise data warehouse (DW, DWH, or EDW) is a database used for reporting and data analysis. It is a central repository of data which is created by integrating data from multiple disparate sources. Data warehouses store current as well as historical data and are used for creating trending reports for senior management reporting such as annual and quarterly comparisons.

The data stored in the warehouse are uploaded from the operational systems (such as marketing, sales etc., shown in the figure to the right). The data may pass through an operational data store for additional operations before they are used in the DW for reporting.

The typical ETL-based data warehouse uses staging, data integration, and access layers to house its key functions. The staging layer or staging database stores raw data extracted from each of the disparate source data systems. The integration layer integrates the disparate data sets by transforming the data from the staging layer often storing this transformed data in an operational data store (ODS) database. The integrated data are then moved to yet another database, often

called the data warehouse database, where the data is arranged into hierarchical groups often called dimensions and into facts and aggregate facts. The combination of facts and dimensions is sometimes called a star schema. The access layer helps users retrieve data.

A data warehouse constructed from an integrated data source system does not require ETL, staging databases, or operational data store databases. The integrated data source systems may be considered to be a part of a distributed operational data store layer. Data federation methods or data virtualization methods may be used to access the distributed integrated source data systems to consolidate and aggregate data directly into the data warehouse database tables. Unlike the ETL-based data warehouse, the integrated source data systems and the data warehouse are all integrated since there is no transformation of dimensional or reference data. This integrated data warehouse architecture supports the drill down from the aggregate data of the data warehouse to the transactional data of the integrated source data systems.

Data warehouses can be subdivided into data marts. Data marts store subsets of data from a warehouse.

This definition of the data warehouse focuses on data storage. The main source of the data is cleaned, transformed, catalogued and made available for use by managers and other business professionals for data mining, online analytical processing, market research and decision support (Marakas & O'Brien 2009). However, the means to retrieve and analyse data, to extract, transform and load data, and to manage the data dictionary are also considered essential components of a data warehousing system. Many references to data warehousing use this broader context. Thus, an expanded definition for data warehousing includes business intelligence tools, tools to extract, transform and load data into the repository, and tools to manage and retrieve metadata.

Benefits of a data warehouse

A data warehouse maintains a copy of information from the source transaction systems. This architectural complexity provides the opportunity to:

- Maintain data history, even if the source transaction systems do not.
- Integrate data from multiple source systems, enabling a central view across the enterprise. This benefit is always valuable, but particularly so when the organization has grown by merger.
- Improve data quality, by providing consistent codes and descriptions, flagging or even fixing

bad data.

- Present the organization's information consistently.
- Provide a single common data model for all data of interest regardless of the data's source.
- Restructure the data so that it makes sense to the business users.
- Restructure the data so that it delivers excellent query performance, even for complex analytic queries, without impacting the operational systems.
- Add value to operational business applications, notably customer relationship management (CRM) systems.

6.2.4 Business Intelligence

Business intelligence (BI) is the ability of an organization to collect, maintain, and organize data. This produces large amounts of information that can help develop new opportunities. Identifying these opportunities, and implementing an effective strategy, can be provided a competitive market advantage and long-term stability.

BI technologies provide historical, current and predictive views of business operations. Common functions of business intelligence technologies are reporting, online analytical processing, analytics, data mining, process mining, complex event processing, business performance management, benchmarking, text mining, predictive analytics and prescriptive analytics.

The goal of modern business intelligence deployments is to support better business decision-making. Thus a BI system can be called a decision support system (DSS). Though the term business intelligence is sometimes a synonym for competitive intelligence (because they both support decision making), BI uses technologies, processes, and applications to analyse mostly internal, structured data and business processes while competitive intelligence gathers, analyses and disseminates information with a topical focus on company competitors. If understood broadly, business intelligence can include the subset of competitive intelligence.

Business intelligence and data warehousing

Often BI applications use data gathered from a data warehouse or a data mart. However, not all data warehouses are used for business intelligence, nor do all business intelligence applications require a data warehouse.

To distinguish between the concepts of business intelligence and data warehouses, Forrester Research often defines business intelligence in one of two ways:

Using a broad definition: "Business Intelligence is a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical, and operational insights and decision-making". When using this definition, business intelligence also includes technologies such as data integration, data quality, data warehousing, master data management, text and content analytics, and many others that the market sometimes lumps into the Information Management segment. Therefore, Forrester refers to data preparation and data usage as two separate, but closely linked segments of the business intelligence architectural stack.

Forrester defines the latter, narrower business intelligence market as, "...referring to just the top layers of the BI architectural stack such as reporting, analytics and dashboards."

6.3 BI Software Benchmarking

An extended analysis on Business Intelligence is The Magic Quadrant for Business Intelligence Platforms[6] published by Gartner.

This study presents a global view of Gartner's opinion of the main software vendors that should be considered by organizations seeking to develop business intelligence (BI) applications. To be included in the Magic Quadrant, vendors must generate at least $15 million in BI-related software license revenue annually. The Magic Quadrant is based on a customer survey including vendor-provided references, as well as survey responses from BI users from Gartner's BI Summit. On 2011 there were 1,364 survey responses.

[6] Published: 6 February 2012 ID:G00225500

Analyst(s): John Hagerty, Rita L. Sallam, James Richardson

http://businessintelligence.info/docs/estudios/Magic-Quadrant-for-Business-Intelligence-Platforms-2012.pdf

6.3.1 Software platform capabilities

As defined by Gartner, business intelligence (BI) platforms enable all types of users — from IT staff to consultants to business users — to build applications that help organizations learn about and understand their business. Gartner defines a BI platform as a software platform that delivers the 14 capabilities listed below. These capabilities are organized into three categories of functionality: integration, information delivery and analysis. Information delivery is the core focus of most BI projects today, but there is an increased interest in deployments of analysis to discover new insights, and in integration to implement those insights.

6.3.1.1 *Integration*

BI Information System — All tools in the platform use the same security, metadata, administration, portal integration, object model and query engine, and should share the same look and feel.

Metadata management — Not only should all tools leverage the same metadata, but the offering should provide a robust way to search, capture, store, reuse and publish metadata objects such as dimensions, hierarchies, measures, performance metrics and report layout objects.

Development tools — The BI platform should provide a set of programmatic development tools and a visual development environment, coupled with a software developer's kit for creating BI applications, integrating them into a business process, and/or embedding them in another application. The BI platform should also enable developers to build BI applications without coding by using wizard-like components for a graphical assembly process. The development environment should also support Web services in performing common tasks such as scheduling, delivering, administering and managing. In addition, the BI application can assign and track events or tasks allotted to specific users, based on predefined business rules. Often, this capability can be delivered by integrating with a separate portal or workflow tool.

Collaboration — This capability enables BI users to share and discuss information, BI content and results, and/or manage hierarchies and metrics via discussion threads, chat and

annotations, either embedded in the BI platform or through integration with collaboration, social software and analytical master data management (MDM).

6.3.1.2 *Information Delivery*

Reporting — Reporting provides the ability to create formatted and interactive reports, with or without parameters, with highly scalable distribution and scheduling capabilities. In addition, BI platform vendors should handle a wide array of reporting styles (for example, financial, operational and performance dashboards), and should enable users to access and fully interact with BI content delivered consistently across delivery platforms including the Web, mobile devices and common portal environments.

Dashboards — This subset of reporting includes the ability to publish formal, Web-based or mobile reports with intuitive interactive displays of information, including dials, gauges, sliders, check boxes and traffic lights. These displays indicate the state of the performance metric compared with a goal or target value. Increasingly, dashboards are used to disseminate real-time data from operational applications or in conjunction with a complex event processing engine.

Ad hoc query — This capability enables users to ask their own questions of the data, without relying on IT to create a report. In particular, the tools must have a robust semantic layer to allow users to navigate available data sources. These tools should include a disconnected analysis capability that enables users to access BI content and analyse data remotely without being connected to a server-based BI application. In addition, these tools should offer query governance and auditing capabilities to ensure that queries perform well.

Microsoft Office integration — In some use cases, BI platforms are used as a middle tier to manage, secure and execute BI tasks, but Microsoft Office (particularly Excel) acts as the BI client. In these cases, it is vital that the BI vendor provides integration with Microsoft Office applications, including support for document and presentation formats, formulas, data "refreshes" and pivot tables. Advanced integration includes cell locking and write-back.

Search-based BI — This applies a search index to both structured and unstructured data sources and maps them into a classification structure of dimensions and measures (often, but not necessarily leveraging the BI semantic layer) that users can easily navigate and explore

using a search (Google-like) interface. This capability extends beyond keyword searching of BI platform content and metadata.

Mobile BI — This capability enables organizations to deliver report and dashboard content to mobile devices (such as smartphones and tablets) in a publishing and/or interactive (bidirectional) mode, and takes advantage of the interaction mode of the device (tapping, swiping and so on) and other capabilities not commonly available on desktops and laptops, such as location awareness.

6.3.1.3 *Analysis*

Online analytical processing (OLAP) — This enables end users to analyse data with extremely fast query and calculation performance, enabling a style of analysis known as "slicing and dicing." Users are (often) able to easily navigate multidimensional drill paths. And they (sometimes) have the ability to write-back values to a proprietary database for planning and "what if" modelling purposes. This capability could span a variety of data architectures (such as relational or multidimensional) and storage architectures (such as disk-based or in-memory).

Interactive visualization — This gives users the ability to display numerous aspects of the data more efficiently by using interactive pictures and charts, instead of rows and columns. Over time, advanced visualization will go beyond just slicing and dicing data to include more process-driven BI projects, allowing all stakeholders to better understand the workflow through a visual representation.

Predictive modelling and data mining — This capability enables organizations to classify categorical variables and to estimate continuous variables using advanced mathematical techniques. BI developers are able to integrate models easily into BI reports, dashboards and analysis, and business processes.

Scorecards — These take the metrics displayed in a dashboard a step further by applying them to a strategy map that aligns key performance indicators (KPIs) with a strategic objective. Scorecard metrics should be linked to related reports and information in order to do further analysis. A scorecard implies the use of a performance management methodology such as Six Sigma or a balanced scorecard framework.

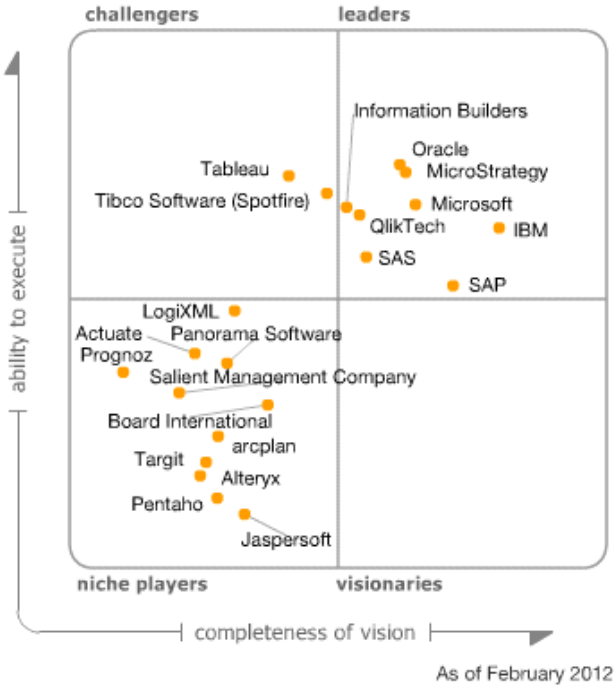

Figure 74 Magic Quadrant for Business Intelligence Platforms. Source: Gartner (Feb. 2012)

6.3.2 Evaluation Criteria

6.3.2.1 *Ability to Execute*

Vendors are judged on their ability and success in making their vision a market reality. In addition to the opinions of Gartner's analysts, the scores are based on three sources: customer perceptions of each vendor's strengths and challenges derived from BI-related inquiries with Gartner; an online survey of vendor customers conducted in late 2011, yielding 1,364 responses; and a vendor-completed questionnaire about the vendor's BI strategy and operations.

- **Product/Service:** How competitive and successful are the goods and services offered by the vendor in this market? This includes current product/service capabilities, quality, feature sets and skills, whether offered natively or through OEM agreements/partnerships.

- **Overall Viability:** What is the likelihood of the vendor continuing to invest in products and services for its customers? Viability includes an assessment of the overall organization's financial health, the financial and practical success of the business unit, and the likelihood of the individual business unit to continue to invest in the product, continue to offer the product and advance the state of the art within the organization's portfolio of products.
- **Sales Execution/Pricing:** Does the vendor provide cost-effective licensing and maintenance options? This covers the technology provider's capabilities in all presales activities and the structure that supports them. This includes deal management, pricing and negotiation, presales support and the overall effectiveness of the sales channel.
- **Market Responsiveness and Track Record:** Can the vendor respond to changes in market direction as customer requirements evolve? This covers the ability to respond, change direction, be flexible and achieve competitive success as opportunities develop, competitors act, customer needs evolve and market dynamics change. This criterion also considers the provider's history of responsiveness.
- **Marketing Execution:** Are customers aware of the vendor's offerings in the market? This assesses the clarity, quality, creativity and efficacy of programs designed to deliver the organization's message in order to influence the market, promote the brand and business, increase awareness of the products and establish a positive identification with the product/brand and organization in the minds of buyers. This mind share can be driven by a combination of publicity, promotional, thought leadership, word-of-mouth and sales activities. This criterion was not rated separately this year and therefore was given a "no rating" in the Magic Quadrant model. Instead, our assessment of Market Execution was combined with Market Responsiveness and Track Record into one criterion on this year's Magic Quadrant.
- **Customer Experience:** How well does the vendor support its customers? How trouble-free is the software?
- **Operations:** What is the ability of the organization to meet its goals and commitments? This criterion was given a "no rating." Assessment of a vendor's ability to meet its goals and commitments is incorporated into the Market Responsiveness and Track Record criterion.

Evaluation Criteria	Weighting

Product/Service	high
Overall Viability (Business Unit, Financial, Strategy, Organization)	high
Sales Execution/Pricing	high
Market Responsiveness and Track Record	standard
Marketing Execution	no rating
Customer Experience	high
Operations	no rating

6.3.2.2 ***Completeness of Vision***

Vendors are rated on their understanding of how market forces can be exploited to create value for customers and opportunity for themselves. The scores are based on the same sources like the *Ability to Execute* criteria.

- **Market Understanding:** Does the vendor have the ability to understand buyers' needs, and to translate those needs into products and services?
- **Marketing Strategy:** Does the vendor have a clear set of messages that communicate its value and differentiation in the market?
- **Sales Strategy:** Does the vendor have the right combination of direct and indirect resources to extend its market reach?
- **Offering (Product) Strategy:** Does the vendor's approach to product development and delivery emphasize differentiation and functionality that maps to current and future requirements? The major business analytics market growth drivers described in the Market Overview section of this report were used as a rubric to assess both the Offering (Product) Strategy and Innovation criteria, which are combined into one score this year.
- **Business Model:** How sound and logical is the vendor's underlying business proposition? Note that this criterion has been given a "no rating" because all vendors in the market have a viable business model.
- **Vertical/Industry Strategy:** How well can the vendor meet the needs of various industries, such as financial services or the retail industry?
- **Innovation:** How well does the vendor direct related, complementary and synergistic layouts of resources, expertise or capital for investment, consolidation, defensive or pre-

emptive purposes? How well does the vendor exploit current or new technologies and combine them in a novel way to address a market need? Innovation and Offering (Product) Strategy are combined into one score for the purpose of this year's Magic Quadrant.

- **Geographic Strategy:** How well can the vendor meet the needs of locations outside its native country, either directly or through partners?

Evaluation Criteria	Weighting
Market Understanding	high
Marketing Strategy	high
Sales Strategy	high
Offering (Product) Strategy	high
Business Model	no rating
Vertical/Industry Strategy	standard
Innovation	no rating
Geographic Strategy	standard

6.3.3 Quadrant Descriptions

6.3.3.1 *Leaders*

Leaders are vendors that are reasonably strong in the breadth and depth of their BI platform capabilities and can deliver on enterprise wide implementations that support a broad BI strategy. Leaders articulate a business proposition that resonates with buyers, supported by the viability and operational capability to deliver on a global basis.

6.3.3.2 *Challengers*

Challengers offer a good breadth of BI platform functionality and are well positioned to succeed in the market. However, they may be limited to specific use cases, technical environments or application domains. Their vision may be hampered by a lack of coordinated strategy across the various products in their BI platform portfolio, or they may lack the marketing effort, sales channel, geographic presence, industry-specific content, and awareness offered by the vendors in the Leaders quadrant.

6.3.3.3 *Visionaries*

Visionaries are vendors that have a strong vision for delivering a BI platform. They are distinguished by the openness and flexibility of their application architectures, and they offer depth of functionality in the areas they address, but they may have gaps relating to broader functionality requirements. A Visionary is a market thought-leader and innovator. However, it may have yet to achieve sufficient scale — or there may be concerns about its ability to grow and provide consistent execution.

6.3.3.4 *Niche Players*

Niche Players are those that do well in a specific segment of the BI platform market — such as reporting or dashboarding — or that have limited capability to innovate or outperform other vendors in the market. They may focus on a specific domain or aspect of BI, but are likely to lack depth of functionality elsewhere. Or they may have gaps relating to broader BI platform functionality. Alternatively, Niche Players may have a reasonably broad BI platform, but have limited implementation and support capabilities or relatively limited customer bases, such as in a specific geography or industry. Or they may not yet have achieved the necessary scale to solidify their market positions.

6.3.4 Major Vendor Strengths and Cautions

We have considered the principal vendors to analyse as:

- IBM
- Oracle
- SAP
- Microsoft
- Pentaho
- SAS
- MicroStrategy
- QlikTech
- Tableau

The conclusions of the Gartner analysis about these vendors are:

IBM

Strengths

- ✓ IBM maintains its leading position on the Completeness of Vision axis for 2012 Magic Quadrant. The company takes a holistic approach to what it calls Business Analytics and Optimization (BAO), combining comprehensive software, hardware and services in a coordinated market offering. IBM's business analytics software portfolio includes a unified BI, analytics and performance management platform, and is complemented by IBM information management software and appliances.
- ✓ In 4Q10, IBM introduced its latest business analytics platform, IBM Cognos 10. Throughout 2011, additional capabilities have been released and customer adoption has begun in earnest.
- ✓ Advanced analytics is a particular IBM strength. The company's SPSS software continues to advance nicely, readily allowing IBM to bid for predictive analytics and statistical use cases.
- ✓ The top reasons why customers select IBM are functionality, ease of use for end users, and data access and integration. IBM's road map and future vision weighed heavily in reference decisions. In 2011, IBM delivered a new Cognos 10 mobile application for the iPad that is included free in existing user roles. In early 2012 the company will introduce Cognos Insight, a personal, desktop BI product that enables independent discovery and "what if" modeling, while also providing full interoperability with the larger workgroup and enterprise solutions.

Cautions

- Again this year, references consider the Cognos products more difficult to implement and use than those of competitors. References indicate that Cognos software is used largely by a consumer/casual user population. Reporting is the most extensively deployed component, followed by ad hoc query and OLAP analysis.
- IBM's customers also continue to have less than optimal customer experiences, with support and sales interactions, along with product quality..

- License cost continues to be another source of customer concern across all products in the IBM business analytics portfolio. Higher than expected costs to upgrade from Cognos 8 to Cognos 10 have stalled some projects, but changes in configuration, user roles, and/or support costs appear to drive the increase. As a counterpoint, existing Cognos 10 users did not identify license cost as a concern.

Microsoft

Strengths

- ✓ Microsoft offers a competitive set of BI capabilities, packaging and pricing that appeal to Microsoft developers and its independent distributor channel. The company has consistently invested in building and enhancing BI capabilities into three of its core offerings — Microsoft Office (specifically Excel), Microsoft SQL Server and Microsoft SharePoint — in order to increase their value and drive upgrades. By incorporating BI capabilities into its most ubiquitous products, Microsoft virtually guarantees its BI offering's continued adoption, particularly in organizations with a Microsoft-centric information Information System. As a result of this strategy, since the company's serious entry into the market in 2000, Microsoft's BI market share has grown steadily to take the No. 3 spot in 2010.
- ✓ Microsoft's low-license-cost bundling strategy for BI platforms makes it a compelling license-cost value proposition for organizations that want to deploy BI to a wider range of users, or that want to lower overall BI portfolio license costs by using lower-cost BI tools for basic BI functions. Its license cost profile is comparable to open-source BI vendors, and is considerably less than its commercial competitors.
- ✓ Microsoft's market success is also driven in part by its IT-oriented, BI authoring tools within SQL Server, which are based on Visual Studio, the broadly adopted development environment. This approach, along with targeted marketing efforts and programs for building strong developer communities and support, has helped Microsoft lower the cost and expand the availability of its BI skills.
- ✓ While Microsoft has traditionally focused on the developer, it continues to enhance reporting, dashboarding and data discovery capabilities in Excel with the intention of making Excel not only the most widely deployed BI tool, but also the most functional for business users.

- ✓ Use of OLAP functionality by Microsoft customers is among the highest when compared to other vendors. This can be attributed to the success and adoption of Microsoft SQL Server Analysis Services functionality bundled with Microsoft SQL Server and its optimizations with Microsoft front-end tools.
- ✓ Microsoft's cloud-based DataMarket offering, which makes external data easier to consume, analyse and integrate with internal data, is a unique enhancement to Microsoft's portfolio of BI capabilities. DataMarket is an online data market that enables ISVs and business users to access, purchase and analyse trusted, public-domain and commercial premium data. ISVs can use this data to build new analytic applications. Business users can incorporate and analyse this external data with internal data sources using Microsoft Excel and PowerPivot, or with partner tools, such as those from Tableau Software.

Cautions

- Since Gartner began surveying BI platform customers for this Magic Quadrant research five years ago, this is the first year that Microsoft has scored below the survey average on key Ability to Execute measures, including overall product functionality, support and customer experience.
- Multiproduct complexity is a challenge. Because Microsoft's BI platform capabilities exist across three different tools (Office, SQL Server and SharePoint) that also perform non-BI functions, integrating the necessary components and building the applications is left to the organization. Microsoft's do-it-yourself approach puts more of the BI solutions development and integration onus for the platform components on customers, compared with the all-in-one purpose-built BI platforms offered by most other vendors in the BI market. Microsoft's road map for Office, which features the consolidation of more and more front-end reporting, dashboard and analysis capabilities in Excel, should begin to address some of this complexity over time.
- Microsoft lags behind most other BI vendors in delivering mobile BI capabilities. It has, instead, relied on partners to build mobile solutions for Apple iOS that integrate with Microsoft BI components. Microsoft BI assets can run in a browser today, but they are not optimized for iOS, Android or Windows devices.

- There is currently no single business metadata layer or capability that spans Microsoft's BI platform components, and there are limited capabilities for sophisticated metadata modeling, impact analysis, data lineage and change management.

MicroStrategy

Strengths

- ✓ MicroStrategy specializes in enterprise BI deployments running on top of large enterprise data warehouses. Its customers cite functionality, performance and support for large data volumes as top reasons for selecting it as a vendor. Its deployments are among the most complex in terms of large numbers of users, the highest data volume, broad product functionality use, wide deployment across an enterprise, and complexity of analytic workload, and its customers have a high level of satisfaction with product functionality.
- ✓ MicroStrategy has a focused vision that maps to key high-value market requirements, particularly for mobility, and large and diverse data, including social media data sources. The company was one of the first vendors to invest heavily in deploying BI applications on mobile devices, with earlier successes than its competitors in accumulating a respectable number of large production mobile deployments. Free trials and online training make it easy for developers to try and succeed with mobile development. Beyond mobility, MicroStrategy continues to reinforce its enterprise-scale pedigree through initiatives for high performance across all layers of its platform and against extremely large and diverse datasets. MicroStrategy has invested heavily in creating a cloud offering that includes its platform and complementary technologies, including ETL and data warehousing. Social data is another forward-looking area of focus for MicroStrategy. This past year, the company delivered a Facebook connector to enable organizations to integrate Facebook profile data, with user permissions, into a MicroStrategy analytic application.
- ✓ Developer productivity for building complex analytic applications is another of MicroStrategy's strengths. Its efficient, parameterized report development paradigm and object-oriented report development environment support centralized management, in which a small number of administrators can support big BI projects with many users, complex reporting and analysis requirements, and a large amount of data. With an

extensive library of prebuilt objects, including metrics, prompts, filters and statistical functions, developers can create reports and other analytic content with high degrees of formatting and analytical sophistication, but with less effort and cost than many other platforms.

- ✓ In March 2011, MicroStrategy introduced a data discovery capability, Visual Insight, this complements and fully integrates with its enterprise, report-centric architecture. Visual Insight is available as a feature of Report Services reducing the need for most customers to purchase stand-alone interactive visualization/data discovery products. Visual Insight is also available in a free personal cloud-based version.
- ✓ MicroStrategy has built its BI platform from the ground up through completely organic development. The high level of integration of the individual platform components and the reusability of MicroStrategy's well architected and object-oriented semantic layer are the result of this strategy.

Cautions

- While the MicroStrategy development environment is robust and flexible, there is a steep learning curve, even for seasoned report developers building any level of analytic complexity into parameterized reports that simulate ad hoc analysis and interactive dashboards for business users. The need for interactivity beyond parameterized reports and dashboards will only increase with broader mobile BI application user adoption.
- Even though MicroStrategy has comparatively moderate administration costs per user compared to its competitors, its customers report above average license and implementation costs per user.
- While MicroStrategy Mobile, its new social data capabilities and its personal cloud offering will increase its appeal to business users and line of business owners, the company currently sells predominantly to IT, which has a stack-centric buying tendency.

Oracle

Strengths

- ✓ In 2011, Oracle Business Intelligence Foundation Suite, with its principal component Oracle Business Intelligence Enterprise Edition (OBIEE), continued to execute on its

stated top-to-bottom BI vision. This year, the products have the highest aggregate *Ability to Execute* scores.

- ✓ References select Oracle primarily for functionality, enterprise application integration, and data access capabilities. Additionally, customers indicated that they valued the products' ability to support large numbers of users. Like other megavendors, the product road map plays an important role in the evaluation process. Ease of use and cost do not factor significantly into the selection process.
- ✓ Oracle Business Intelligence Applications (OBIA) are predefined analytic applications for horizontal business processes such as finance, procurement and sales analysis. Additionally, the company also delivers vertical-specific analytic data models for industries such as retail and financial services for IT buyers looking to establish a common data model standard as the foundation for analytics.

Cautions

- References rate OBIEE as difficult to implement.
- Product functionality evaluation scores remain below average again this year, a trend that appeared in last year's report. Additionally, customer support and product quality issues are rated below the average (in the fourth and third quartiles respectively) for all vendors in this report. In fact, both support and product quality were also noted as issues that blocked further deployments within customer organizations.
- Oracle customers use the product mostly for static report viewing, parameterized reporting and scorecard capabilities, leading to below average user complexity ratings.

Pentaho

Strengths

- ✓ Pentaho makes its debut on the Magic Quadrant this year. It provides a comprehensive open-source BI platform composed of ETL, OLAP, reporting, dashboards, ad hoc analysis and data mining components, all managed from a central BI server deployed either on-premises or in the cloud, with end-user access via the Web or mobile devices such as the iPad.
- ✓ Low license cost is central to Pentaho's value proposition. The No. 1 reason that customers choose Pentaho is for its perceived low license cost and TCO.

- ✓ Pentaho's lightweight footprint, in which the complete platform can be deployed in a small environment on a laptop, or can be integrated into an existing scalable architecture such as a grid for much larger deployments, makes it very flexible in meeting a broad range of deployment requirements. Moreover, Pentaho is an embeddable platform, making it very attractive to ISVs and internal IT shops for embedded use cases to deploy both on-premises and in the cloud.
- ✓ While open-source vendors, including Pentaho, tend to invest more heavily to achieve feature parity with the core BI functionality of commercial competitors rather than in innovation, Pentaho does have focused areas of forward-looking investment, on which it has been able to deliver quickly.

Cautions

- Pentaho's below average aggregate product scores (with the exception of predictive modelling) are still an indication that functional gaps in the platform remain. Moreover, Pentaho needs to continue to improve on both its business user tools, to meet growing requirements for intuitive and interactive analysis, and the usability and efficiency of its developer-oriented tools. Moreover, ease of use goes hand in hand with the effort to develop content. Despite the perception of low TCO as a primary reason for purchasing Pentaho, users report among the longest BI content development times of all vendors in the Magic Quadrant survey.
- Given that Pentaho's subscription-based model hinges on providing superior support, Pentaho's below average product support scores (particularly related to level of expertise) are a concern, especially since the company's product is also rated below average in terms of product quality, which tends to result in more customer interaction with support.
- Although Pentaho claims a single unified platform managed from a single server, the repositories and authoring environments remain separate, with migration to a single repository in process.

QlikTech

Strengths

- ✓ QlikTech is a marketing juggernaut; it has brand recognition many times more prominent than a firm with its current market share would expect.
- ✓ QlikTech's QlikView product is a self-contained BI platform, based on a wholly in-memory data store, with a set of well integrated BI tools.
- ✓ Gartner frequently sees companies deploy QlikView for prototyping and requirements gathering, leveraging its flexibility to engage end users, usually alongside a more traditionally modeled BI platform.
- ✓ QlikTech's customers report strong delivery of business benefits, particularly in making better information available to more users and expanding the type of analysis undertaken.
- ✓ Customers' rating of QlikView's functionality is very positive in nine out of 14 functional capabilities: dashboards, interactive visualization, mobile BI, search-based BI, scorecards, ad hoc query, Microsoft Office integration, OLAP, and development tools.

Cautions

- QlikTech's growing pains are more evident. For the first time, QlikTech's customers reported having a poor overall customer experience, and below average ratings for product quality and support.
- Gartner continues to hear rumblings of discontent from QlikTech customers about the structure of its pricing model and its high license cost.
- QlikTech faces increasing competition from larger BI vendors offering in-memory offerings and interactive visualization (particularly Microsoft SQL Server PowerPivot/Power View and MicroStrategy Visual Insight), all of which are intent on narrowing QlikView's opportunities for expansion by offering cheaper alternatives.
- QlikTech offers limited metadata management. Filling this gap requires additional cost and effort in the management of metadata to lockdown common definitions and calculations, and to conform dimensions for cross-functional analysis across QlikView applications.
- Although quick to develop simple or moderately complex dashboards, when it comes to building large, complex reports from various data sources, involving detailed logic or calculations, QlikView users reported the second slowest turnaround.

SAP

- ✓ The combination of SAP BusinessObjects and SAP NetWeaver BW revenue accounts for the largest share of the BI platform market, with both SAP platforms continuing to support large enterprise deployments (more than twice the average for both data size and number of users). Similarly, a higher percentage of SAP cite "corporate standards" and "integration with enterprise applications" as among the top reasons why they chose SAP for BI.
- ✓ SAP has one of the largest global direct sales, support, and channel and services ecosystems. Moreover, the combination of SAP and BusinessObjects constitutes the largest installed base in the BI platforms market, which represents a significant and captive cross-sell and upsell market opportunity for SAP.
- ✓ SAP has a compelling and comprehensive product vision that addresses many key future trends including mobile, collaborative analytics, and analytics on big data. SAP complements its BI platform with forward-looking capabilities in the areas of collaboration and decision support (with its StreamWork product), text analysis integrated with its enterprise information management products, and search-based data exploration with its SAP BusinessObjects Explorer product.
- ✓ SAP is investing in industry- and domain-specific packaged applications built with SAP BusinessObjects that include a data model, ETL and business content.

Cautions

- Migration, implementation and integration choices can be confusing.
- While SAP's customers tend to have very large and global deployments, poor performance is mentioned as a problem limiting broader deployment.
- At the end of August 2011, SAP implemented its third license model change (Concurrent Session-Based Licenses [CSBLs] and Named User licenses) for SAP BusinessObjects since the Business Objects acquisition in January 2008. While there are many advantages for users in using CSBL, changing license models have contributed to confusion, and concern. Also, some customers are charged for upgrades when they expected to be provided with product at no/low cost.

SAS

Strengths

- ✓ SAS gets high marks for its global footprint and broad industry initiatives. Unlike some other BI platform vendors, SAS focuses on advanced analytical techniques, such as data mining and predictive modelling, where references acknowledge it as a leader of the pack. SAS's clients also have above average complexity scores (for the depth of use of different BI use cases) on larger than average data sources. SAS customers also access and interpret unstructured internal and external data more often than any other vendor's clients surveyed for this Magic Quadrant.
- ✓ SAS's solution-oriented analytic application approach to the market is a differentiator, giving the company the advantage of having a wide variety of cross-functional and vertically specific analytic applications out of the box for a variety of industries, including financial services, life sciences and manufacturing. While others are also adopting this approach, SAS remains in the lead. Customers also report an above average sales experience.
- ✓ The primary drivers for customers choosing SAS remain functionality and data integration. In addition, references reported that they select SAS because of availability of skills.
- ✓ On the software partnership front, SAS has partnered with a number of database vendors to push the execution of its models directly into the database management system without moving the data. Not only does this reduce data duplication and movement, it also allows SAS users to leverage the power and scalability features of the database to run predictive models against very large datasets with high performance.

Cautions

- References report that SAS is very difficult to implement and companies also indicate that the product is considered difficult to use for business users.
- SAS's dominance in predictive analytics and statistics continues to be challenged on many fronts.
- Customer references report that cost is the most common factor blocking further adoption.

- Despite SAS's success and awareness as a leader in the predictive analytics space, the company is still challenged to make it onto BI platform shortlist evaluations when predictive analytics is not a primary business requirement.

Tableau

Strengths

- ✓ For the third year in a row, Tableau is the "sweetheart" of the Magic Quadrant, with customers even more enamoured with it this year than in the last two. It gained overwhelmingly positive customer survey feedback across the board in all measures in the survey, including ease of use, functionality, product quality, product performance, support, customer relationship, success, achievement of business benefits and view of the vendor's future. These stellar results in part contributed to Tableau's strong *Ability to Execute* position, despite its relatively small size.
- ✓ Tableau is one of a number of stand-alone BI vendors delivering strong interactive visualization for analysis, dashboards, information delivery and managed analytic applications. Tableau's strong performance, even with an increasingly crowded competitive landscape, is evidence of its ability to meet the increased market demand for easy-to-use and intuitive interactive BI tools that are easy to deploy without IT assistance.
- ✓ Tableau's self-contained BI platform provides purpose-built, business-oriented data mashup ETL capabilities with data connectors that leverage Tableau's own VizQL technology (drag-and-drop operations in Tableau create a query in VizQL, which interprets and packages an SQL or MDX query to the database and then expresses the response graphically). Its columnar, in-memory data engine, which can be used as an alternative to its direct query access, enables fast performance on large and multisource datasets and on complex queries, such as very large multidimensional filters or complex co-occurrence or multipass queries. Zero programming data mashup capability, combined with an in-memory database, allows users to blend and visually analyse large amounts of diverse datasets with auto-detect relationships between multiple sources (of any format). This allows users to connect to any data source and produce a series of interactive dashboards, and highlight and visually filter and pass parameters directly from a graphic; or use filters (for example, check boxes, sliders, relative date filters and

drop-down menus); or build in geographic intelligence to analyse their data. Interactive analysis can be shared with a report consumer equipped with a Web browser. The combination of exceptional ease of use with the ability to conduct sophisticated analysis, is a key reason users are exuberant with the platform.

Cautions

- Tableau's product functionality is more narrowly defined around analysis and interactive visualization. It lacks broader BI platform capabilities, such as production reporting and predictive analytics. Tableau has introduced a shareable semantic layer — a key enterprise feature — in its 7.0 release.
- Although Tableau's user counts remain below the survey average (albeit growing from last year), it is still largely departmentally deployed with smaller user counts. Tableau's products often fill an unmet need in organizations that already have a BI standard, and are frequently deployed as a complementary capability to an existing BI platform. Tableau is still less likely to be considered an enterprise BI standard than the products of most other vendors.

6.4 Open Source BI Software vs. Commercial BI Software

In the Open Source Business Intelligence context, we can see that most of the major vendors have free-of-cost community editions and enterprise versions, even some with free trials so you can try before you buy.

There is a community open source version with a well-defined set of functions, bounded and fully operational; and a professional version that presents more features or an enhanced version of the same features.

It may be that the free versions will meet your needs, if you have programming talent in-house able to customize your chosen BI software, but will need a big maintenance effort.

For instance, Pentaho Dashboards supports creating, but only the professional version has a Dashboard Designer Ad-hoc. These (premium) functionalities can be accessed only by purchasing a subscription or support.

6.4.1 What is open source business intelligence?

Open source BI are BI software that can be distributed for free and permits users to modify the source code. Open source software is available in all BI tools, from data modelling to reporting to OLAP to ETL.

Because open source software is community driven, it relies on the community for improvement. As such, new feature sets typically come from community contribution rather than as a result of dedicated R&D efforts.

6.4.2 Advantages of open source BI tools

- ✓ Easy to get started:

  With traditional BI software, the business model typically involves a hefty start-up cost, and then there is an annual fee for support and maintenance that is calculated as a percentage of the initial purchase price. In this model, a company needs to spend a substantial amount of money before any benefit is realized. With the substantial cost also comes the need to go through a sales cycle, from the RFP process to evaluation to negotiation, and multiple teams within the organization typically get involved. These factors mean that it's not only costly to get started with traditional BI software, but the amount of time it takes is also long.

  With open source BI, the beginning of the project typically involves a free download of the software. Given this, bureaucracy can be kept to a minimum and it is very easy and inexpensive to get started.

- ✓ Lower cost:

  Because of its low start-up cost and the typically lower ongoing maintenance/support cost, the cost for open source BI software is lower (sometimes much lower) than traditional BI software.

- ✓ Easy to customize:

  By definition, open source software means that users can access and modify the source code directly. That means it is possible for developers to get under the hood of the open

source BI tool and add their own features. In contrast, it is much more difficult to do this with traditional BI software because there is no way to access the source code.

6.4.3 Disadvantages of open source BI tools

- Features are not as robust:

  Traditional BI software vendors put in a lot of money and resources into R&D, and the result is that the product has a rich feature set. Open source BI tools, on the other hand, rely on community support, and hence do not have as strong a feature set.

- Consulting help not as readily available:

  Most of the traditional BI software - MicroStrategy, Business Objects, Cognos, Oracle and so on, have been around for a long time. As a result, there are a lot of people with experience with those tools, and finding consulting help to implement these solutions is usually not very difficult. Open source BI tools, on the other hand, are a fairly recent development, and there are relatively few people with implementation experience. So, it is more difficult to find consulting help if you go with open source BI.

6.5 Description of Barcelona and Genoa BI interfaces

6.5.1 Barcelona BI Interface

The municipality of Barcelona is now working with many BI softwares (Cognos, Microsoft, QlikView, Pentaho...) as a consequence of different strategies in different areas along the last period.

The last years have been of Cognos consolidation, although many departmental suites (as QlikView) have been maintained in order to supply specific needs with a fast-deployment solution.

Barcelona is now developing the future BI strategy at city level, analysing which are the best solutions to implement in order to choose the BI software that fits to the whole organisation with

a broad vision, despite some departmental suites might still be maintained. It is expected to have a decision taken in the next 2-3 months.

The analysis being developed includes many comparative criteria that may be useful to share them with iCity partners.

- Gartner's Magic Quadrant: The vendors in the "Leaders" quadrant usually have the ability to combine good performance with a flexible response to market demands
- The most comprehensive tool for us: It is important to differentiate tools supporting all BI functionalities from specific products focused in an area (for example data analysis)
- Licensing costs - price to quality equilibrium: Analysis of functionalities needed and users to provide the solution to find the best balance for the organisation.
- Integration & scalability - previous platform integration: Look for the existing product that allows maximum integration with the existing platform providing scalability.
- Maintenance cost: Apart from the cost of licenses there is another cost to bear in mind as are the resources consumed by each one of the platforms (memory...)
- Market knowledge - Price negotiation advantages: The more knowledge has the market about a tool, the easier may be to find suppliers with expertise and best price competition
- Importing external data: Validate if the chosen tool has some complexity to import external data that may require technical background

6.5.2 Genoa BI Interface

The municipality of Genoa is now working with two different BI software:

- The first and most consolidated is based on the instruments supplied by Microsoft technology. This Information System has been used since 2006, and refers to Analysis Services for the multidimensional DB and Reporting Services for the presentation layer.

  The choice of these products is due to the fact that Municipality of Genoa uses Microsoft sqlserver as a standard.

- Actually, to provide fast-deployment and distributed solutions we are implementing new environments based on Qlik View. This tool will become the standard BI solution provided to all the Municipality.

Both instruments will continue to be used by Municipality, relating to the different users needs.

6.6 DWH key points

6.6.1 Relational or non-relational Data Base

We need more than just the traditional relational SQL database functionality (RDBMS) for our iCity Platform.

NoSQL can service heavy read/write workloads compared to traditional RDBMS (relational database management system), and scale up to Terabyte (TB) size across replicated, commodity hardware. The primary driver for NoSQL is achieving massive, redundant distributed storage for large scale iCity applications.

If you want to make fast data analysis, you need to use a 'schema free' database architecture, designed for large data warehousing and high frequented data base changes.

A good example of a NoSQL database is the **MongoDB**. It can track and store tons of data about user interactions.

For example, you take an action on Meet Up, it will update your user references and update all your friends**. Non-relational stores** are really good at that and you can afford to keep that data in multiple places.

MongoDB (from "hu**mongo**us") is an open source document-oriented database system developed and supported by 10gen. It is part of the NoSQL family of database systems. Instead of storing data in tables as is done in a "classical" relational database, MongoDB stores structured data as **JSON-like** documents with dynamic schemas (MongoDB calls the format **BSON**), making the integration of data in certain types of applications easier and faster. For transactions, we still prefer a relational database, for example MySQL.

In a high scalable web architecture, a widely used open source memory object caching system is **memcached.** This is intended to speed up dynamic web applications by alleviating database load. It's a short term memory for the applications. (http://memcached.org)

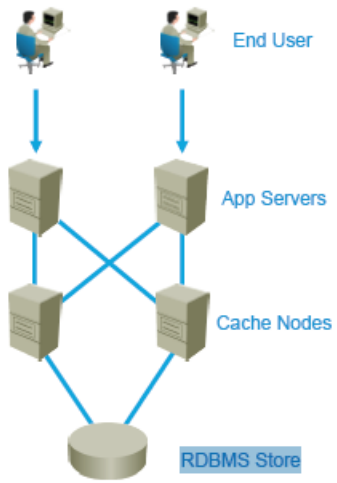

Figure 75 Dedicated memcached machines

An **Index store** is used as a cache for the meta-data: for example uploaded videos.

The index store reduces the need for complex and resource intensive queries to the RDBMS Store. (Multiple instances are required per deployment for redundancy)

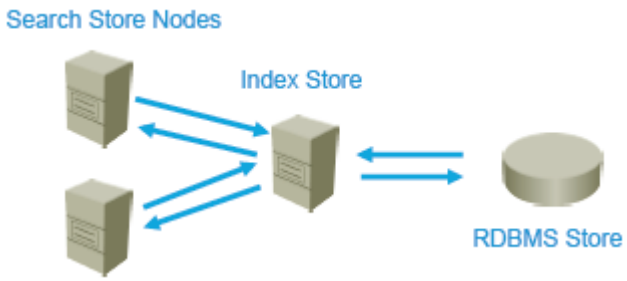

Figure 76 Index Store

6.6.2 Video management

This section presents a proposal of video solution based on Cisco Show and Share[7] product, that could be included in future prototype versions.

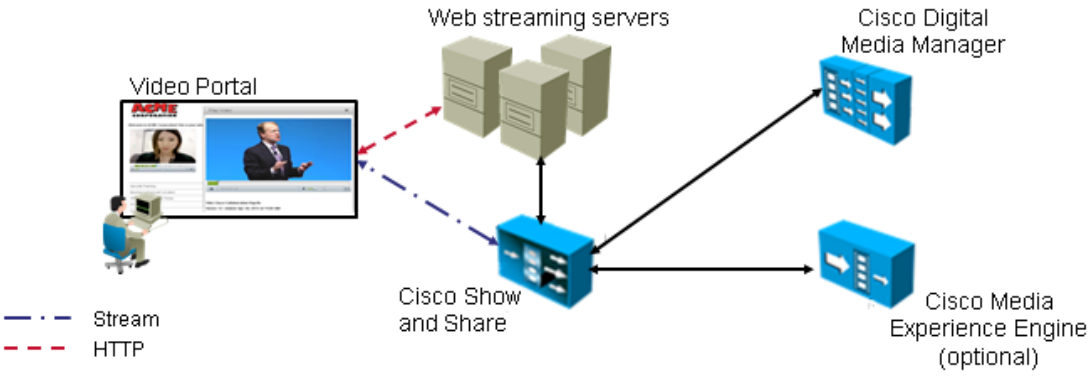

Figure 77 Show and Share components

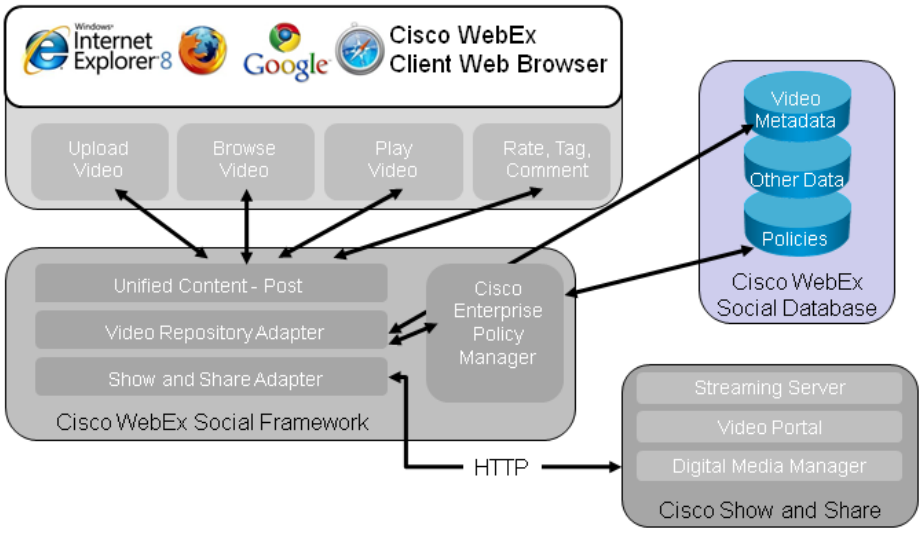

Figure 78 Show and Share architecture from client perspective

[7] http://www.cisco.com/en/US/prod/collateral/video/ps9339/ps6681/data_sheet_c78-565776.html

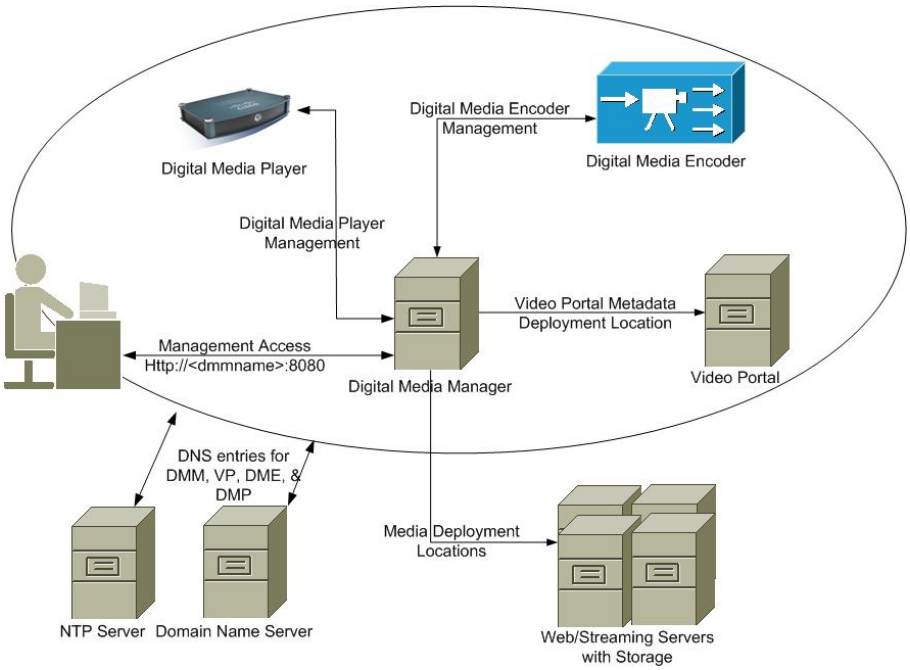

Figure 79 Video management

Used for videos embedded via the Webex Social Editor (e.g.: Posts, Message Boards). There will be APIs to integrate with other systems.

6.7 iCity DWH & BI Prototype

6.7.1 DWH adaptation

The iCity platform prototype adapts a simple version of data warehouse which overspreads the following requirements:

- ✓ Provide a standard protocol to access the data
- ✓ Resolve queries related historical data
- ✓ Be able to support future business intelligence deployments

The DWH adaptation prototype has been done to provide the following features, which will allow easy developments in the future:

- **Flexible** allowing the integration of different types of Information Systems located in the cities. The information provided by cities Information System is stored in homogeneous system that provides standard interactions for any service provided by iCity Platform. Finally, add new Information System could be done in an easy way.
- **Scalable** allowing to be improved and addition of new features in a easy and optimal way.
- **Dynamic** allowing the adaptation of different environment in order to resolve new requirements and needs.
- **Homogenized** allowing the iCity service to have a standard view and normalized access to data which is coming from resource layer.
- **Interoperability** providing technological transparency and allowing the communication of different technologies.
- **Modularity** based on quality software policies where different parts and procedures of the system are able to be reused, updated and replaced in a efficient and optimal way in order to facilitate the management.

Thus, first version of data warehouse is related with the storage layer and the catch up services module of iCity platform prototype as it is showed in the following picture:

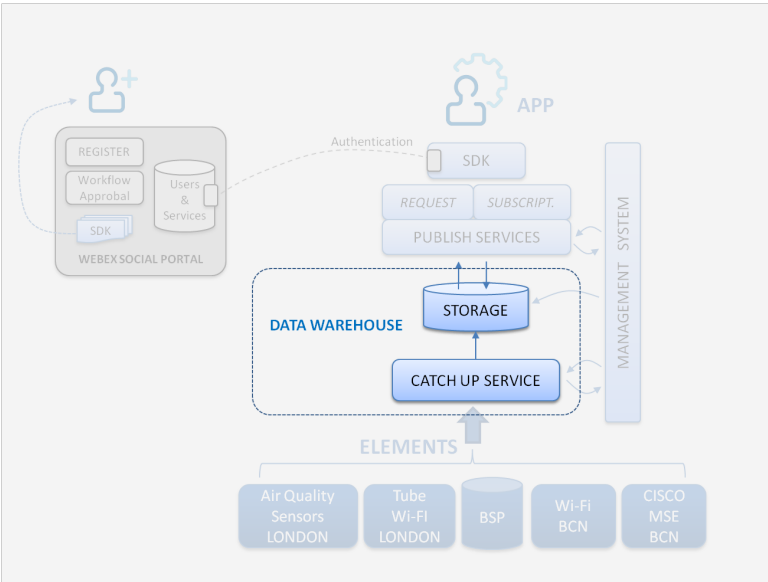

Figure 80 Data Warehouse prototype

Catch Up Services module:

This module is in charge of transferring the incoming data, which comes from the heterogeneous Information Systems, to the homogeneous storage layer of the iCity platform.

Cath up services acquire and recollect data from different information sources, allowing different communication protocols and offering a huge range of possibilities to integrate different types of Information System. Catch up service is composed of different modules, each module translates data coming from the Information System to a normalized model, reducing the integration and providing a generic communication of upper layers.

Finally, this module feeds the storage module with normalized data.

Storage module:

The storage module has the data structure based on OGC[8], in order to provide homogeneous data to the upper layers of iCity platform prototype.

[8] http://www.opengeospatial.org/

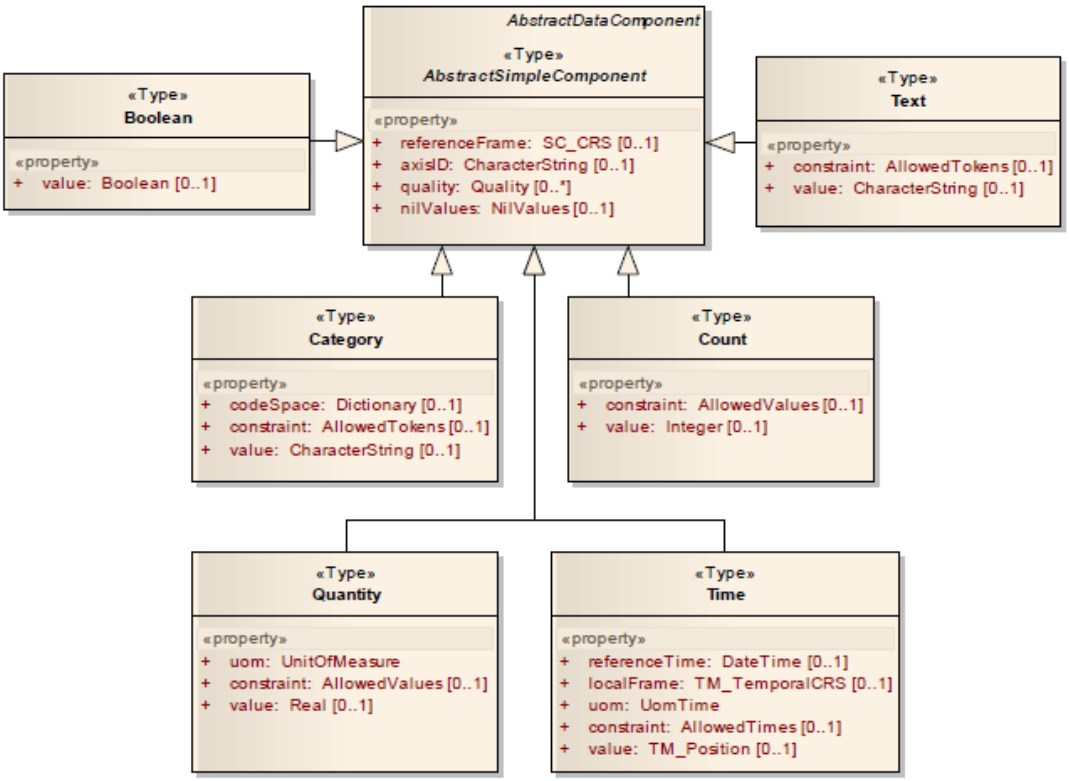

Figure 81 Example of data base model

Data coming from catch up services is managed by this module. Main features of this module are:

- To enable enrichment process that generates information from data, converting raw data to a cooked data.
- To provide normalized access to data for service layer in the iCity Platform.
- To offer a repository of historical data in order to reduce cost acquiring data from cities Information System.

At this stage of the project, as there isn't large amount of data to be managed, the storage module has been now developed using a relational data base.

6.8 Conclusions

This year it has been deployed pilots and apps, with a BI that enriches and enables the validation of different exploitation business models and new business services.

Data Warehouse offers the following benefits:

- To integrate data from multiple sources.
- To perform new types of analyses.
- To reduce cost to access historical data.
- To normalize data across the iCity Platform.
- To share and allow others to easily access data.

During the fourth year, from M36 to M42, we have been focused on the integration of new information systems with the iCity platform and the adaptation of the iCity API and its documentation to their requirements. In addition, we have been working in the iCity Platform improvements and its maintenance.

We are now involved in the deployment and publication process for the iCity APPShowCase.

7. iCity OPEN DATA

7.1 Introduction

This section presents the work on Open Data performed in the second working period in the task 4.5. The outcome of this work is the iCity Open Data portal. The emphasis was on the development of the portal itself and the population with existing metadata from three cities participating the iCity project.

The portal is an instance of the Open Data platform developed by Fraunhofer FOKUS, which was customised for iCity needs.

The cities involved in the iCity project are very advanced regarding their ICT (Information and Communication Technologies) Information System and already have - or will have very soon - their own Open Data platform. Therefore, the iCity Open Data platform plays the role of a single point of access to Open Data portals of the involved cities. Some cities in perspective can consider migrating their Open Data registries to it.

7.2 Architecture

The Open Data platform is built from two main components: The Open Data Portal and the Open Data Registry as it is depicted in Figure 82. The Open Data Portal consists of a publicly accessible part and a non-publicly accessible part. The publicly accessible part provides users with a web front-end that facilitates data browsing, search, and consumption. The non-publicly accessible part is reserved for civil servants or authorized third parties and allows them to publish metadata for data sets or edit existing metadata in the registry.

The implementation of the Open Data portal is based on the community edition of Liferay (licensed under the GNU Lesser General Public License), a portal server by Liferay, Inc. A portal server aggregates several web applications, called portlets, into one web page. Each portlet is developed to handle a specific job. The aggregation of specialised portlets results in a high-scalable modular system. The Liferay and the Open Data Portal portlets are written in Java and run on Apache Tomcat.

The Open Data Registry is a major component of the Open Data Platform. It stores the metadata associated with data sets that are catalogued in the data portal. In order to register new metadata, one can either do it by using the Open Data Portal or by using the Open Data Registry import API, which can be accessible via the department CMS or other third party's application. Furthermore, the Open Data Registry provides a comprehensive API that allows to read and search through metadata. The Open Data Registry is instantiated by using the CKAN[9] (Comprehensive Knowledge Archive Network) data catalogue portal software, the de-facto European standard for metadata registries in the Public Sector Information (PSI) domain. CKAN is a free software suite maintained by the Open Knowledge Foundation. CKAN is a metadata registry that enables publishing, sharing and finding metadata entries, called data packages. It uses a predefined cataloguing schema built on a set of metadata terms. Among core features of CKAN are customizable metadata registry schema, which enables adding extra fields, and multiple ways for maintaining metadata entries (i.e., via the CKAN admin web page or via the CKAN API). Open interfaces of CKAN enable seamless integration and federation with other open data portals.

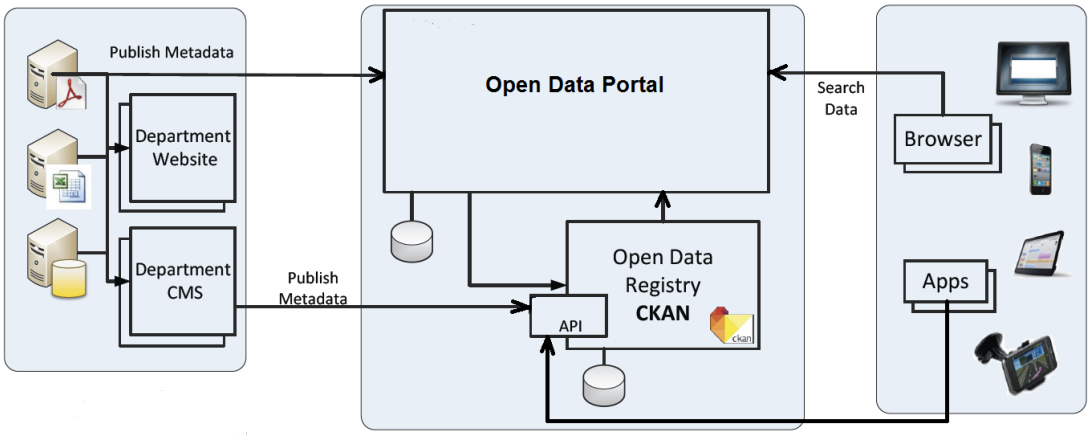

Figure 82 Architecture of the Open Data Portal

[9] http://ckan.org/

7.3 Theming and Static Content

In order to adapt the appearance of the iCity Open Data Portal to the entire iCity project the Fraunhofer FOKUS Open Data platform got a special customized theming. A new colour scheme was developed and integrated together with a newly created icon set.

Furthermore the static content was adjusted to the needs of the iCity project and the whole user interface was translated from German to English.

Additionally, a library portal section was introduced to present an overview of the legal framework of important aspects regarding the iCity project. The content was prepared for the portal and shows the legislation on publishing of public data in European law and in the legal frameworks of Barcelona, Bologna, Genoa and London.

7.4 Metadata

Three from four cities involved in the iCity project already have their own Open Data platform. Genoa is developing its Open Data platform. The common interest of the involved cities is the integration of their Open Data platforms under iCity Platform in order to be able to benefit from the cross-platform search. That will enable iCity users to easily find open datasets in the cities through a common search interface and will foster the use of the published Open Data. Each Open Data platform uses its own format to store the metadata, which made it necessary to develop custom transformers for each of them (see Figure 83). For Barcelona, Bologna and London metadata harvesters were implemented and metadata were harvested into the iCity Open Data portal. The datasets can be filtered by these three data sources (see Figure 84). A custom Java application queries the data from the platforms, converts it into the iCity metadata scheme and writes it via the CKAN API in the iCity Open Data Registry. CKAN uses the JSON as data exchange format.

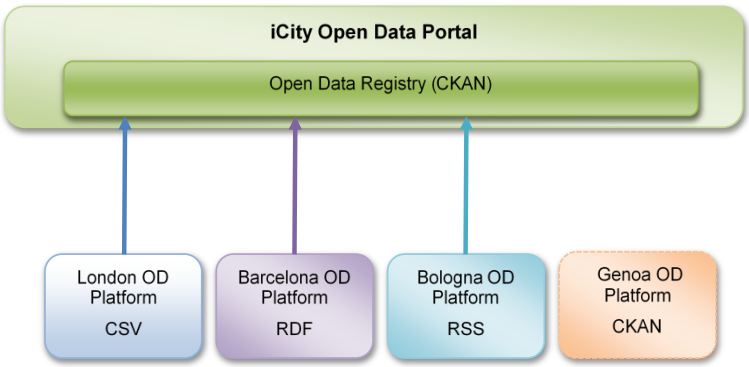

Figure 83 Harvesting Meta Data for the Portal

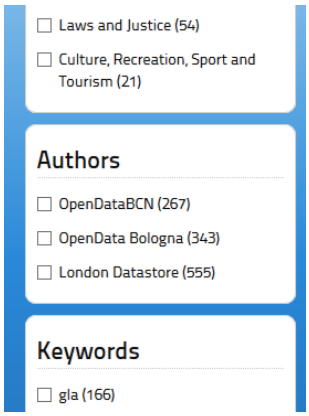

Figure 84 Filter for the Different Authors

7.4.1 London Datastore

The entire data catalogue of the London Datastore is accessible as a CSV file[10]. This file is updated daily from the MySQL database, which holds the actual data. It includes full metadata of every dataset. A special Java library for parsing CSV files (supercsv.sourceforge.net) was

[10] http://data.london.gov.uk/datafiles/datastore-catalogue.csv

used to read the data. The metadata structure is defined by the column titles of the CSV file. Each line represents one data set. The mapping of the basic fields is the following:

CSV File	**CKAN Schema**	**Comment**
TITLE	title	
DRUPAL_NODE	name	Trailed to unique string
DATASTORE_URL	url	
MAINTAINER	maintainer	
LONGDESC	notes	
TAGS	tags	
CATEGORIES	groups	The original categories are mapped to the iCity Open Data portal groups in terms of content
DOWNLOAD_URL, EXCEL_URL, CSV_URL, KML_URL, XML_URL, GOOGLEDOCS_URL	resources	

7.4.2 OpenData BCN

The Barcelona Open Data platform does not provide a sophisticated API for developers. Instead, it is possible to download the entire catalogue as an RDF file[11]. By parsing the received XML file an easy access to the entire data catalogue can be achieved. Every data field and links to the resources are included. The metadata information stored in the Open Data platform and accessible through the catalogue is only in Catalan. The Apache Jena Framework (jena.apache.org) was used to parse the file and convert into JSON. Because of the complexity of the RDF format and the implemented mapping we leave it out of this document.

7.4.3 OpenData Bologna

The Bologna Open Data platform offers a RSS feed[12] containing all metadata stored in the platform. The feed is updated on a daily basis. The JDOM library (www.jdom.org) was used to parse the XML of the RSS feed and transfer it into JSON. The tags of the RSS feed are mapped to the portals scheme according to the following table:

RSS Feed	CKAN Schema	Comment
title	title	
description	notes	
guid	name	Trailed to unique string
guid	url	

[11] http://opendata.bcn.cat/opendata/en/catalog/SECTOR_PUBLIC/data-catalog/0/RDF

[12] dati.comune.bologna.it/tuttidati.xml

enclosure	resources	
category	groups	The original categories are mapped to the iCity Open Data portal groups in terms of content

7.4.4 Genoa Open Data

The Open Data Platform for Genoa[13] has been published online in February 2015. It is a CKAN based Open Data portal using only the default basic CKAN data structure. The iCity Open Data portal has a more extended data structure, which completely covers the data structure of the Genoa's portal. Therefore transformation of metadata from this portal to the iCity Open Data portal has been implemented on the basis of direct mappings.

7.5 Extended Harvesting Application

The Java application, which was used to harvest, transform and write the metadata into the iCity Open Data portal is well suited for single metadata harvesting activities, but has to be further extended to update the harvested metadata on the regular basis. In 2015 Fraunhofer FOKUS will integrate this new functionality by the end of the iCity project. The core component is an interactive web application, allowing the user a much better administration of the harvesting and transformation process. The mapping of data fields from the source portals to the target portal, described in the preceding chapters, can be maintained within the web application. This allows a more flexible and fast adoption to changes. Furthermore the harvesting process can be monitored in a detailed way. The application creates detailed reports of every harvesting process, allowing the user to identifier irregularities within the source data.

[13] http://ckan.comune.genova.it/

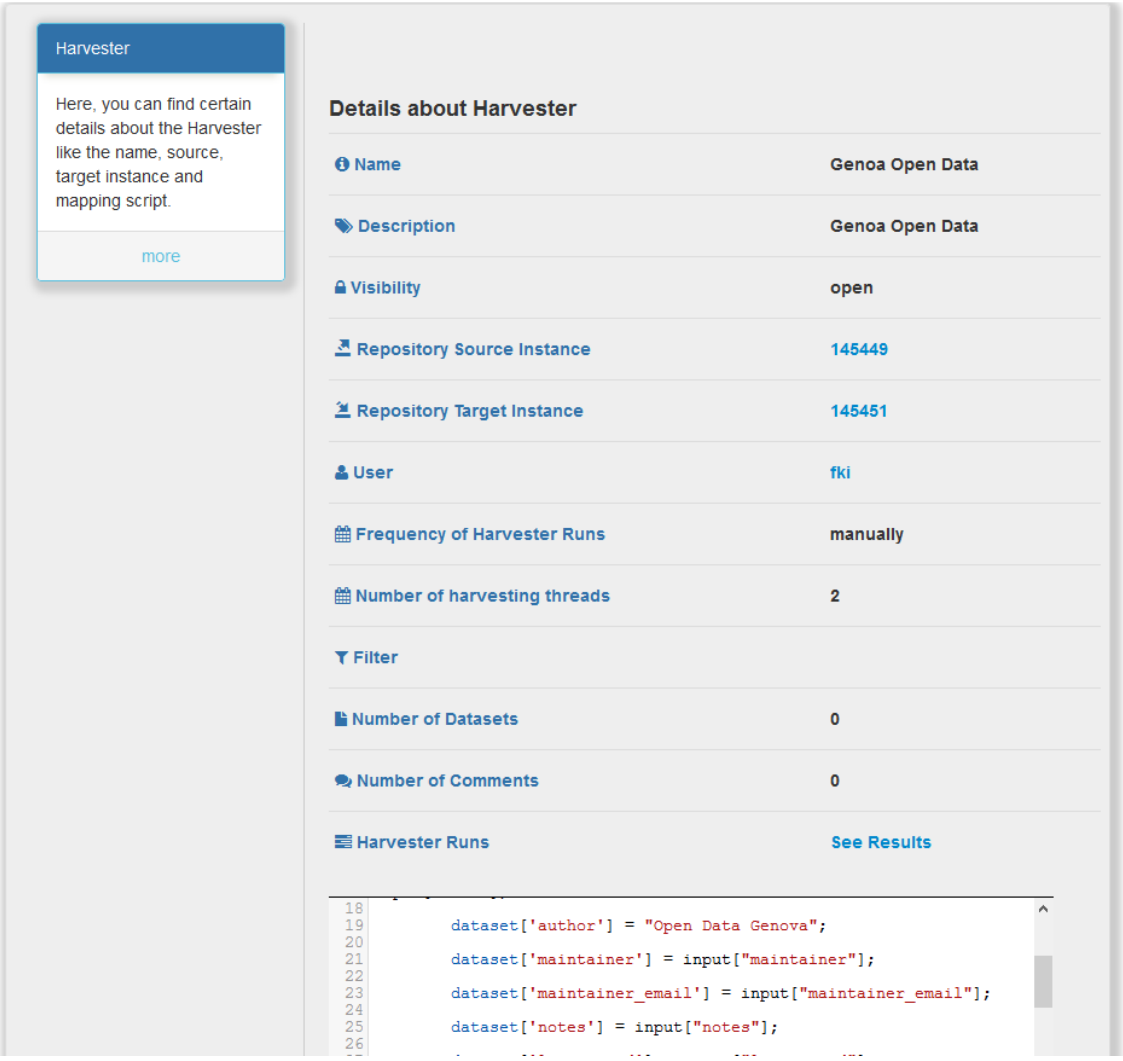

Figure 85 Extract of the Harvesting Application (Genoa Harvester)

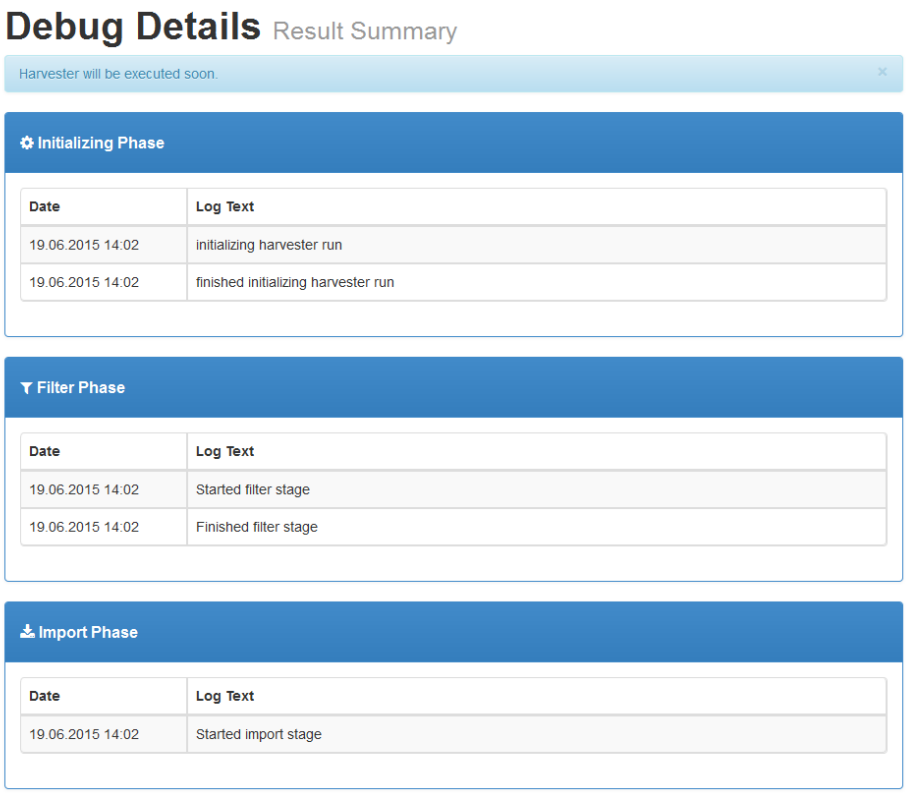

Figure 86 Extract of a Harvesting Report

7.5.1 Automatic and Scheduled Harvesting

In addition the extended application allows a scheduling and automatic execution of the harvesting process. The already created transformation scripts can be applied regularly, based on a user-defined schedule. That will allow keeping the iCity Open Data portal in sync with the respective source portals. It is planned to run the harvesters once a week.

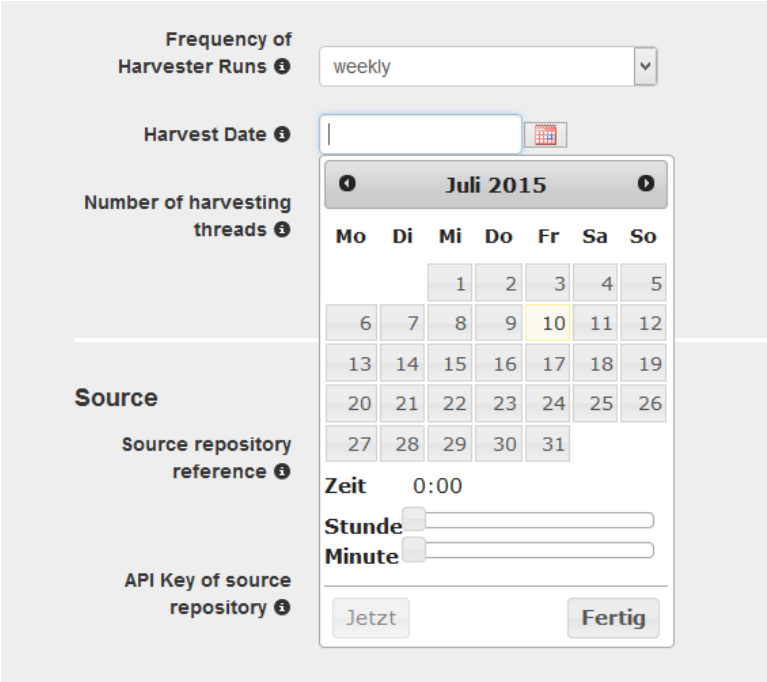

Figure 87 Scheduling a Harvester

7.6 iCity Open Data M42

The main achievement on Open Data in the reporting period is the integration of the new Genoa's Open Data portal in the iCity Open Data portal. Until the end of the project a dedicated service for regular harvesting of metadata from cities Open Data portals in the iCity Open Data portal will be deployed.

8. iCity SDK

The aim of this section is to describe two topics; the first one is about the architectural overview and technical considerations to create a SDK prototype for iCity and the second one is about a REST API for iCity which enables the access to iCity Data feed and Open Information Systems.

Is remarkable to say that the ultimate objective of the strategy behind these topics is to spread the using of these services around the 'techies' segment, with a special focus in those involved in the development of mobile technologies, this does not means forgetting the traditional ones (as desktop or server applications) but to offer the right conditions to propagate its using in an environment which is more and more delocalized.

The main objective during this year has been to provide developers new functionalities through a unique interface of communication with the different Information Systems that are integrated to iCity. This unique interface is the current iCity REST API.

Finally, it is important to mention that this section will be alive and modified during the whole life of the project in order to adapt its contents to the final iCity platform architecture.

8.1 Objectives

So the main objectives of iCity SDK/API REST are the following:

- To give the project an appropriate access to the platform.
- To give the project an appropriate access to Cities Information Systems in a transparent way.
- To give support an application abstraction layer to easily integrate with other applications.
- To keep in touch with stakeholders in order to improve the functionalities of the iCity SDK/ API REST.

- To provide different tools for developers to develop new apps.
- To develop an innovative method for user interactions, using different operating systems.

8.2 iCity mapping SDK

The following image shows the iCity platform prototype where SDK is the main interface between applications and iCity platform.

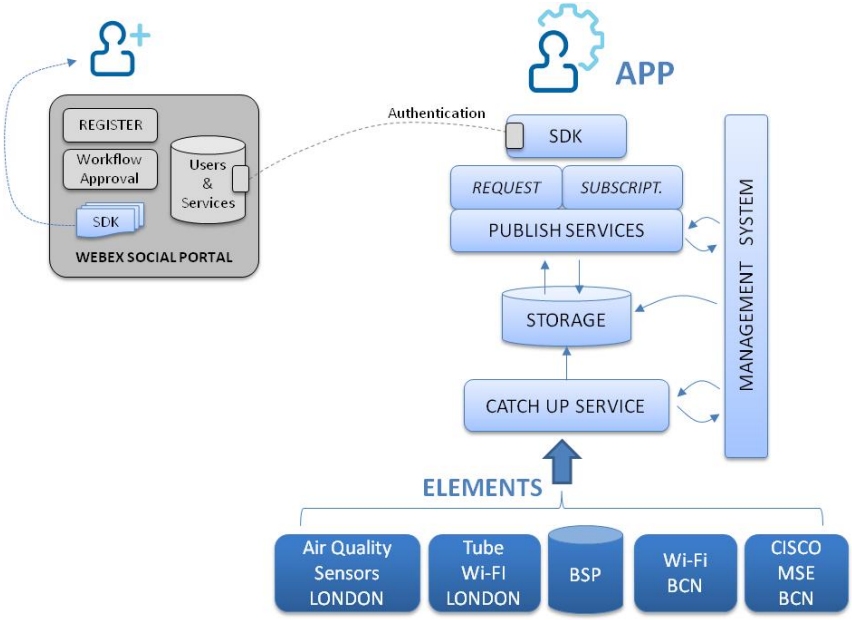

Figure 88 1st version of iCity prototype

The prototype follows the architecture designed and defined by WP3. Figure 89 shows the first prototype of SDK and their mapping over iCity architecture defined by WP3.

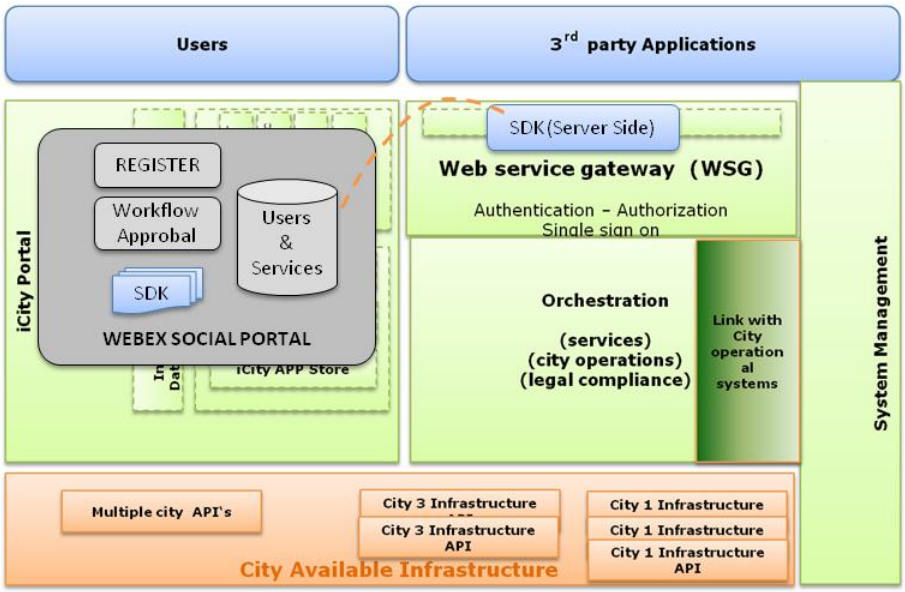

Figure 89 Mapping SDK over iCity architecture

For the correct operation of the SDK is needed the following modules in the iCity prototype:

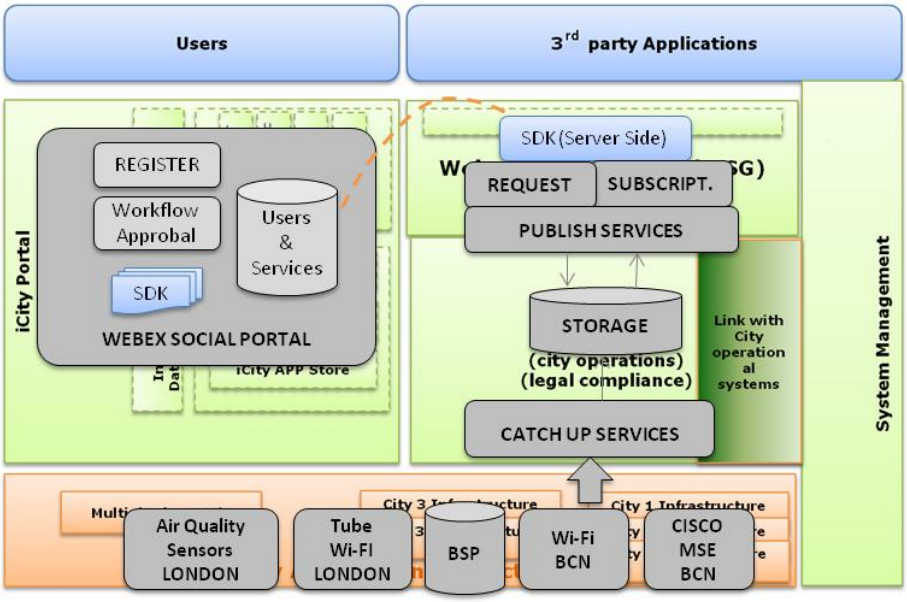

Figure 90 Mapping iCity Prototype over iCity architecture

The catch-up services module provides the connection of both, the open Information Systems and 3rd parties' platforms, to iCity platform. Is constituted of adapters where every adapter interacts with a specific open Information System or platform. Thus, when a new Information System or platform will be opened to iCity platform, just a new adapter must be deployed to integrate the new API of the Information System. This implementation allows easy growing and also technological interoperability at this layer of the iCity platform.

The storage module of the prototype has been developed using a relational database. The data structure is based in OGC.

The publish service module is based on OGC standard protocol and supports both types of publish services: request and subscription.

- Request services allow queries to obtain historical data and also actual data.
- Subscription services allow queries to subscribe to a type of data defined by a filter during the subscription process. Thus, queries will receive the data when obtained.

The iCity apps interact with the opened Information Systems through the iCity SDK, but not directly with the opened Information System. The publish service provide the data to SDK through Storage module. And even more important, the catch-up services module transforms the data collected from heterogeneous iCity Information Systems and stores in a homogeny way.

Finally, SDK allows the implementation of security access to open Information Systems.

8.3 iCity SDK Definition

In this chapter we are going to describe what a SDK means, which components does it contain and which is the relation between the SDK and the other components of the iCity platform.

iCity Software Development Kit (SDK) provides all the functionalities and features needed to build powerful and innovative open Information System applications. The iCity SDK consists of a series of modules, or building blocks, easily configurable and extendable. The modules address

different kind of functionalities and are grouped into Client, server and communication. They are flexible and could be adapted according to customer needs.

Consequently with the strategy, the proposed architecture, platform and design rules will be focused on guarantee the next characteristics (sometimes opposite by their natural trend) that we will define as the Design Principles:

- Ease of use,
- Compatibility,
- Extensibility,
- Portability and,
- Maintainability.

Ease of use: this means the SDK has to encapsulate the Domain Model and make the services transparent to the client. It also means helpful documentation and samples to assist in its using.

Compatibility: Nonetheless, despite the encapsulation the SDK has to guarantee to keep providing the raw xml information as it comes from the services and sensors, in order to allow new definitions (or not defined in the current SDK version) that client can handle by herself while the SDK does not encapsulate it.

Extensibility: The SDK should provide Interfaces and abstract entities that can be easily extended by the client, to create their own models (still keeping compatibility).

Portability: The SDK should use a standard that gives flexibility in order to allow its using between different platforms. The language chosen for the prototype and the recommendation for other SDKs development is C# in combination with .NET and Mono compilers that can make the SDK's assemblies reach maximum market quota

Maintainability: The SDK should be developed according to a modern architecture pattern, this includes following DDD, TDD and N-Layers. In the user's side, it should provide a web environment where to place the information, FAQ, bugs and issue tracking. Finally it should consider a marketing strategy that includes blog, events and community participation.

8.4 iCity SDK Design

8.4.1 Comparison between WS, API or SDK

Although it is out of the scope of this paper to give a fully description of each of these terms, because this definition is done in WP3, it is necessary to present them and the relation they keep together, so that clarify the reason of use a SDK instead of a Web Service or API.

According to the picture below we can see that in the current context they all complementary and subsystem for the system above, being the **Web Service** the communication channel, the **API** the client assembly that encapsulates the model and exposes/consumes the service, and the **SDK** as the previous two ones plus the set of tools, information, samples, tutorial and whatever other resource that is provided to the developers.

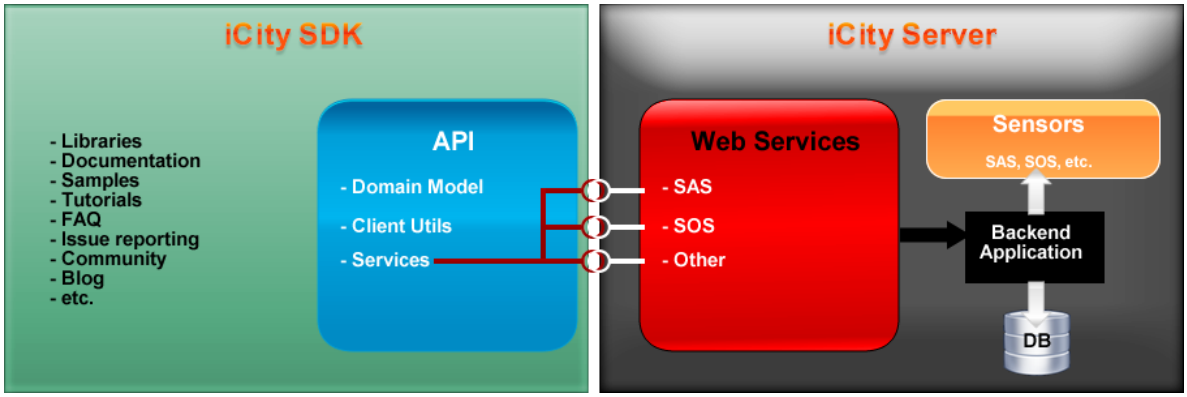

Figure 91 Workflow for application to access the iCity platform

In order to be as generic as possible, we have decided to use an SDK based on the OGC standard Sensor Web Enablement (SWE). The Open Geospatial Consortium (OGC) was established in 1994 and it's composed by both many public and private organizations. Its main purpose is to define open standards and interoperable within the GIS and the World Wide Web. They pursue agreements between different companies that enable the interoperation of geo-processing systems and facilitate the exchange of geographical information.

The Open Geospatial Consortium's Sensor Web Enablement (SWE) activities, which have been executed principally through the OGC Web Services (OWS) initiatives under the Interoperability Program, is establishing the interfaces and protocols that will enable "Sensor Webs" through which applications and services will be able to access sensors of all types over networks such

as the Internet and with the same standard technologies and protocols that enable the Web. These initiatives have defined, prototyped and tested several foundational components needed for a Sensor Web, namely:

1. **Observations & Measurements (O&M)** - The general models and XML encodings for sensor observations and measurements.
2. **Sensor Alert Service (SAS)** – A service by which a client can register for and receive sensor alert messages. The service supports both pre-defined and custom alerts and covers the process of alert publication, subscription, and notification.
3. **Sensor Model Language (SensorML)** – The general models and XML schema for describing sensors and processes associated with measurement.
4. **Sensor Planning Service (SPS)** – A service by which a client can determine collection feasibility for a desired set of collection requests for one or more sensors/platforms, or a client may submit collection requests directly to these sensors/platforms.
5. **Transducer Markup Language (TML)** – General characterizations of transducers (both receivers and transmitters), their data, how that data is generated, the phenomenon being measured by or produced by transducers, transporting the data, and any and all support data (metadata) necessary for later processing and understanding of the transducer data.
6. **Web Notification Service (WNS)** – A service by which a client may conduct synchronous dialogues (message interchanges) with one or more other services. This service is useful when many collaborating services are required to satisfy a client request, and/or when significant delays are involved is satisfying the request.

A **Sensor Observation Service** provides an API for managing deployed sensors and retrieving sensor data and specifically “observation” data. Whether from in-situ sensors (e.g., water monitoring) or dynamic sensors (e.g., satellite imaging), measurements made from sensor systems contribute most of the geospatial data by volume used in geospatial systems today.

The goal of SOS is to provide access to observations from sensors and sensor systems in a standard way that is consistent for all sensor systems including remote, in-situ, fixed and mobile sensors. This is a challenging task because the users of sensor data have historically been divided into those who primarily deal with in-situ sensors and those who primarily deal with remote sensors. The terminology, perspective, and expectations of these two broad groups are different. SOS leverages the Observation and Measurements (O&M) specification for modeling

sensor observations and the TransducerML and SensorML specifications for modeling sensors and sensor systems.

The approach that has been taken in the development of SOS, and the SWE specifications on which it depends, is to carefully model sensors, sensor systems, and observations in such a way that the model covers all varieties of sensors and supports the requirements of all users of sensor data. SOS leverages the standard properties of these two data types (sensors, observations) to provide specialized operation signatures for observation data.

SOS has three mandatory “core” operations: GetObservation, DescribeSensor, and GetCapabilities. The GetObservation operation provides access to sensor observations and measurement data via a spatio-temporal query that can be filtered by phenomena. The DescribeSensor operation retrieves detailed information about the sensors and processes generating those measurements. The GetCapabilities operation provides the means to access SOS service metadata. Several optional, non-mandatory operations have also been defined. There are two operations to support transactions, RegisterSensor and InsertObservation, and six enhanced operations, including GetResult, GetFeatureOfInterest, GetFeatureOfInterestTime, DescribeFeatureOfInterest, DescribeObservationType, and DescribeResultModel.

Used in conjunction with other OGC specifications, the SOS provides a broad range of interoperable capability for discovering, binding to and interrogating individual sensors, sensor platforms, or networked constellations of sensors in real-time, archived or simulated environments.

From now, in the iCity case, we use the “core” operations, in order to obtain all the information related to the open Information Systems involved in the iCity Project.

Get Capabilities:

This function allows obtaining the main information about the system which could be divided in two areas:

- System information
- System offerings

We mean by system information, the name and description of the service, version and people involved in the project and all the methods supported by the standard.

We understand by system offerings, all the devices of the system, its positioning, measures and the response format.

At the beginning we could use this function in order to know all the capabilities of the system.

DescribeSensor

This function allows obtaining a complete description of a single device of the system (sensor, hub, etc..).

It includes this information:

- Identifier: identifies the device with a unique identifier.
- Description: describes the element.
- Position: position of the element.
- Information type: manufacturer, version, etc…
- Dependencies: in case the element pertains to another one.
- Measures: information about the measures provided by an element

8.4.2 Architecture

According with the Design Principles we will split the architecture in three different scopes:

Common Library

It keeps the Domain Models described in the OGC encapsulated as well as the services contracts and other common utilities.

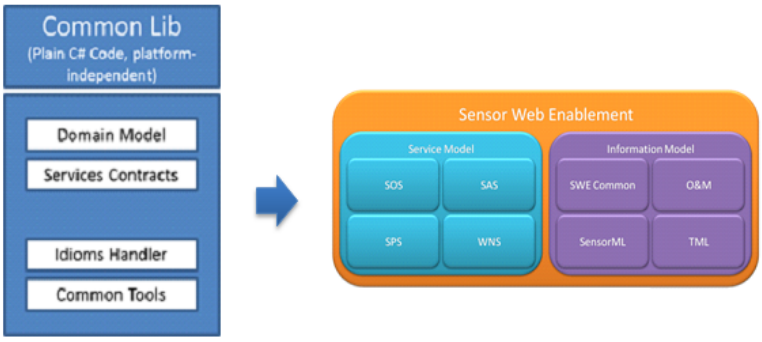

Figure 92 Common Library

Client API

This encapsulates the common lib and provides it as an assembly compiled for the supported platforms that will be consumed by a client application. The main functions of this API are to make the using of the services transparent to the user and to offer extra functionality which can be common for every platform (conversions, format, etc) or specific behaviour for each one (e.g. special classes or properties for iOS, Android, Windows, etc.).

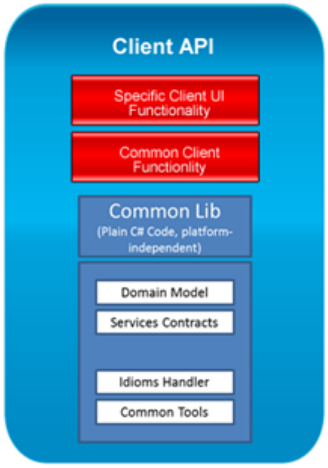

Figure 93 Client API

Server Services

This scope involves all the aspects related to adapt the server to provide the services according with the Services Contracts defined in the Common Library. If the server is created from the raw it should follow the Design Principles implementing not only the Common Library but following also DDD, TDD and N-Layers methodologies.

It is implementation decision to decide channels and communication technologies, in fact there are plenty of alternatives that are good candidates as it shown in the picture below, but it is mandatory that both the API and the Server implements the same Services Contracts with the same behaviour so the communication will be transparent and will affect exactly equal to every client no matter the platform.

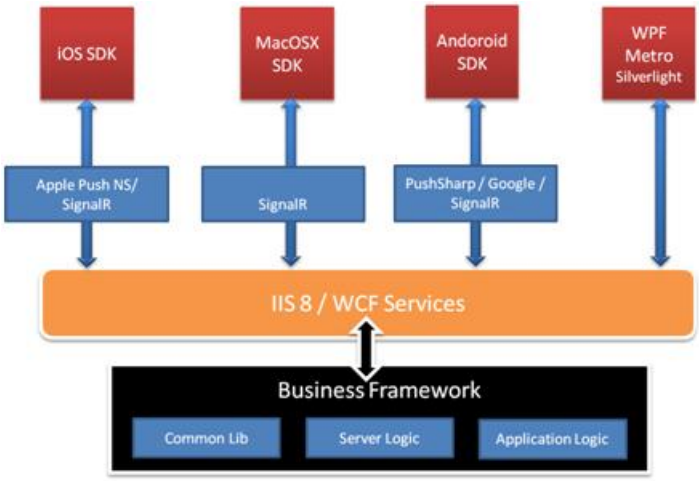

Figure 94 Server Services

8.4.3 Modules

We will classify the modules according with the scope define before:

<u>Common Library</u>

Domain Model: This domain model is subdivided also into two different parts:

- **Metadata and System models**: the properties and entities described in the O&M (e.g. Observation, Process, Feature Type, etc.) and that are used generically by all the services when providing actions (e.g. get capabilities, subscribe, etc.) and information coming from the sensors or alerts.

- **Sensor models**: unities and data types that are provided by supported sensors and alerts by a service. This also includes conversion entities according with the unity types.

- **Services Contracts**: The service contracts are the signature of the methods provided by each service exposed and define specific (and common between the server and the client API) input and output types. There are also two different parts:

    - **Operational Services Contracts**: those ones that correspond to the available services (e.g. SOS, SAS, etc.) There will be one Service Contract file for each one of the services.

    - **System Services Contracts**: those ones that are not included in the previous one, usually Permissions, Registration, Idioms, etc. Usually those related to administrative actions.

- **Idioms Handler**: In this module is placed everything related with Localization Resources (dictionaries, languages available, etc.) - but the Service Contracts - in order to provide culture adaption capabilities to the clients (in case they need it).

- **Common Tools**: Common libraries of programming tools that can be shared between Server and clients

Client API

- **Common Library**: described before but mentioned here as it is the main part of the API core.

- **Common UI functionality**: This component is optional and it depends on the functionality strategy. This would include View Controllers (Presenters, View Models, etc.) non-platform dependent with the purpose to provide predefined views to clients. They must be independent of platform and expose extensibility for platform specific actions.

- **Common functionality**: Those programming tools that can be share between the different platforms but not with the server (in other words, every client-side library tool that cannot be placed in the Common Library)
- **Platform functionality**: In case it is necessary, before compiling the API for each different supported platform, it will be added to an intermediate native project that will extend the API with platform specific actions.

Server Services:

- **Common Library**: described before but mentioned here as it is necessary to construct and to expose the services that will be consumed by the Client API.
- **Service handler Layer**: this is the intermediate layer between the Server backend application and the services exposed, in case they already existed before and not created based on the Common Library here it will be also a translation component between the existing server Domain Model and the Common Library one.
- **Service Layer**: Exposed through interfaces, here will be placed each one of the services defined in Common Library Contracts. It's up to the implementation to decide if the services will be exposed directly (for instance in case of it is still wanted that the services are able to be consumed as web services without the SDK) or through a proxy service that then redistribute the service internally to the service handler layer.

8.5 iCity Service Certification Process

Creating a service in the iCity Platform probably has some trust, technical and even legal consequences. So, a certification process must be mandatory in the iCity project.

Before a service is used, we need to verify it's compliant with the iCity platform rules, city strategy, legal and also technical aspects, this is a critical part of the iCity platform.

The purpose of this paper is to explain this certification process for services that use open Information Systems within the project iCity.

This process (it is mandatory and free of cost) has 3 main approval layers: city strategy, legal aspects and technical questions.

8.5.1 The petition

A third party organization makes an application proposal to create a new service using iCity platform. This petition contains meta-information about the service proposal, in fact, every iCity app is defined by a set of metadata. This information will be provided by the iCity Service developer when starting the iCity Service Certification Process and reviewed later if necessary. This is important because before a service is published, has to verify that it is compliant with the iCity platform rules in three main areas: city strategy, legal aspects and also technical questions.

The meta-information of each iCity app should be (some of this information will be available at the beginning of the certification process, other will be at the end of this process or when the service is published).

- Service name.
- Short description.
- Size (in Kb).
- Images (logo + screen captures).
- Owner/Developer.
- Last version available, Date.
- Language.
- User license.
- Which Information Systems are to be used?
- Cost, if any? If so, what does it cost?
- Final devices:
    - Mobile: which OS? Google Android, Apple iOS, Symbian, Microsoft Windows Phone, etc…
    - Web: Supported browsers
    - Tablet
- To which city is it aimed for?

- Mapping (entire city, a district, a particular neighbourhood, a street, a particular point, etc).
- Which is the topic?
- QR code or Link to directly download the result application.
- Score that users give*.
- Users' comments*.

* These two blocks of information are results of the natural use of the service and feed by users, so will be a part of the service's metadata when it is available in the iCity Platform.

8.5.2 City Strategy Approval

The city (where this new service will run) has to check if this new service "matches" the city strategy. If the result is "not valid" then the third party has to modify the petition.

This City Check has three steps:

1. First, the officials of the City must validate the petition; this is a "simple" check in order to ensure that the City has all data needed of the petition. As a result of this step, the City will know which Information System is involved in the petition.

2. The second step is the strategic validation of the petition. A high-level team of the City must decide if this petition is according of the City strategy.

3. When the high-level team validates the petition, then the Information System owner must analyse if the petition matches the technical limits of the Information System. In addition, the owner has to estimate the cost of the eventual changes to do in the Information System.

8.5.3 Legal Aspects Approval

Obviously, the petition has to be law-abiding. So, in parallel of the city strategy checkpoint, the service will be tested in legally aspects. Again, if the result is "not valid" then the third party has to modify the petition in order to make it fit to the legal framework.

8.5.4 Technical Aspects Approval

This step could have several “loops”. The first one is to ensure that the petition “matches” the technical aspects prior to development. If the result is “not valid”, then the third party has to modify the petition only the technical aspects. If the modification includes more than technical aspects, then the process goes back to start point. At this point, the third party can start the developing phase.

When the service past the last loop of the technical approval, then the petition is a real development, it has passed the technical approval. Automatically it will be added in the iCity Apps Store (according to meta-information of the petition).

This is the basic definition of the certification process for a new service, in the case of an upgrade of pre-existing service, then could be only a single loop of technical approval or all process, it depends on how deep the changes are.

iCity Service certification process schema:

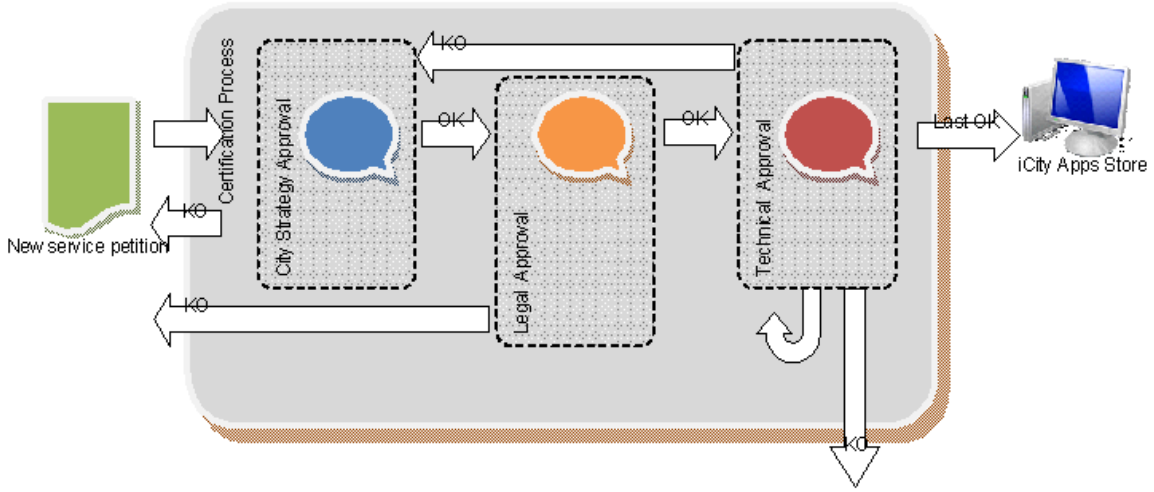

Figure 95 iCity Service certification process schema

8.6 iCity SDK Prototype

According with the previous points definitions, the structure that supports the iCity SDK is defined in this section. This index is classified according with a different scope that the one defined in the design guide as it is now understood that the common library is included within the Client API and Server Services ones. Therefore, the iCity SDK will have three different modules:

- Server side.
- Client side (web site based).
- Collaborative tool (web site based).

8.6.1 Server side

There are two different functions that have relation with the iCity SDK:

- One that provides the Services that are going to be consumed by the SDK
- One that provides hostage for the Client and the Community sides project parts.

The scope of the Server side topic is to describe the first one while the web hostage of the SDK will be defined in a different project.

8.6.1.1 *Platform Overview*

It is up to the implementation to decide if a new server application (full functional[14]) will be created, an existing one will modify or a new mirror server will be placed[15]. Whatever the server application strategy is decided it will expose Services Contracts defined in the Common Library

14 This means developing the whole Framework for the application to handle the domain, persistence, authentication and all the other server functionalities.

15 Consuming services from the first one and redirecting it to the SDK.

and will present the topology described in the picture below (exactly the same described during the Design):

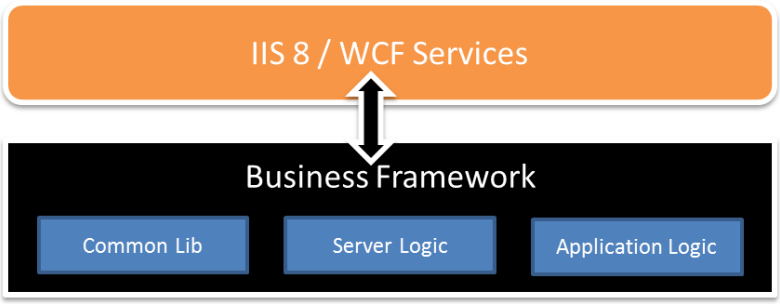

Figure 96 Platform Overview

8.6.1.2 ***Services Protocols***

- Operation Services Contracts (SOS,SAS)
- System Services Contracts

We will not extend in the Operational Services except for saying that current scope of iCity is exposing SOS and SAS for more details about them please go to the referenced documentation.

About the System Services Contracts they will include:

- User authentication
- Idioms Services
- Incidence tracking

For the prototype:

- Operation Services Contracts (SOS, SAS): all the services will be exposed but only a functional subset of the Domain Model will be provided (to be defined in the Prototype Specifications).

- System Services Contracts: simplified versions of the services mentioned above (to be defined in the Prototype Specifications).

8.6.2 Client side

This section corresponds to the set of tools that we provide to developers through collaboration tools. The index of contents it will present in the following charters.

8.6.2.1 *Download Section*

- Libraries (SOS API, SAS API, OGC Model for SAS & SOS)
  - Windows 7/8.
  - Windows Phone 7/8.
  - Android.
  - iOS.
  - Linux.
  - Other.
- Installation Guide.
- Requirements & Compatibility.

For the prototype:

- Libraries:
  - Windows 7/8.
- Included in the prototype a simplified installation guide.

- Included in the prototype a simplified requirement & compatibility document.

8.6.2.2 ***Documentation***

- Platform Overview
- Configuration & Connectivity
- Components
    - SOS API part (To do: Identifiy I/O parameters and extra methods)
    - SAS API part (To do: Identifiy I/O parameters and extra methods)
    - OGC Model (To do: Identify domain entities, create base entities)
- Extensibility Guide (eg Sensor ML, SPS, etc)

8.6.2.3 ***Samples***

There will be samples for each of the main platforms and they will cover different functionality provided by the different Services exposed and the UI extensions (common and platform-specific).

For the prototype is given a sample about SOS service.

8.6.2.4 ***Developer registration process***

The iCity Developer Portal (http://icity-devp.icityproject.com) provides a registration process for new users. Once a developer's account has been approved, it is possible to obtain more information and test the iCity API or simply download information.

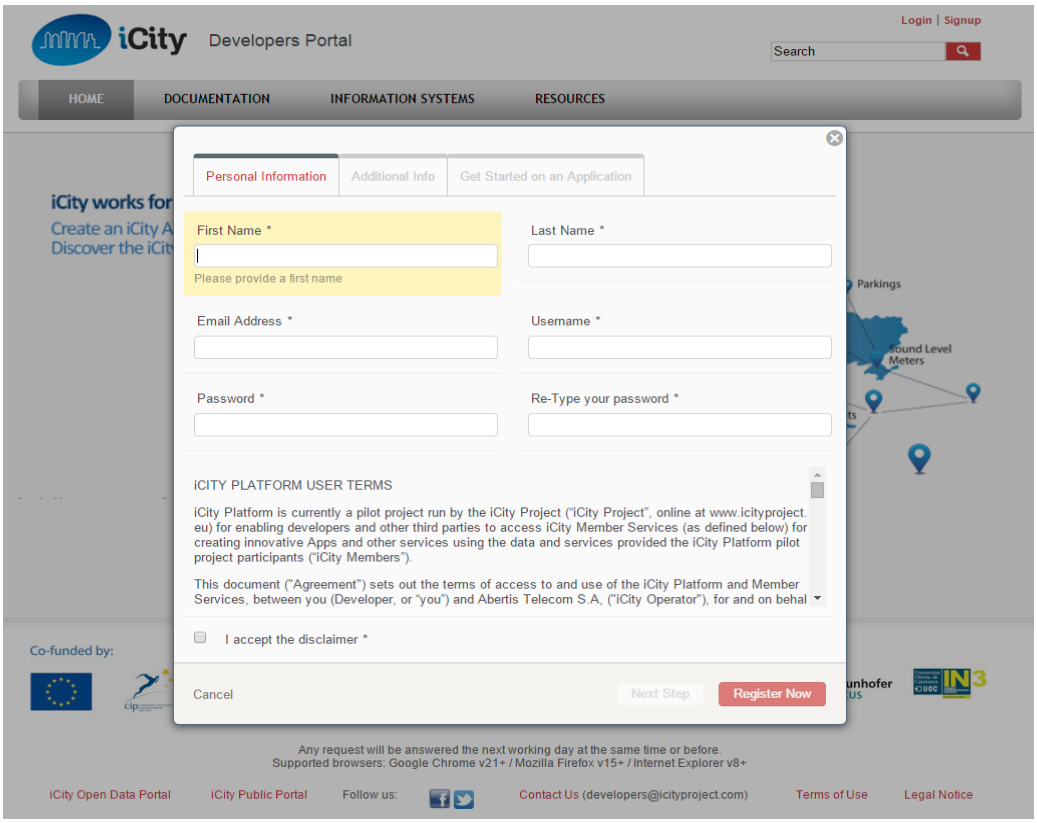

Figure 97 Developer registration Process

The registration process should be the following:

1. First of all, the user has to access to the iCity Platform through the main iCity portal or directly to http://icity-devp.icityproject.com. Click the "Sign up" button.

2. First fill the personal information in this tab.

3. The second tab to fill is about additional information.

4. The third and last tab allows the user to start creating an application.

5. After introducing all the information, a message will be sent to the email specified by the user in order to check the address.

6. Once the email has been verified and the platform's administrators have approved the new user, the access to the platform will be provided.

See more details http://icity-devp.icityproject.com/documentation/Registration%20Process.

8.6.3 Developer documentation

iCity Developers portal provides a space where users can find information about how to register, how to use the API, how to use the portal, etc…

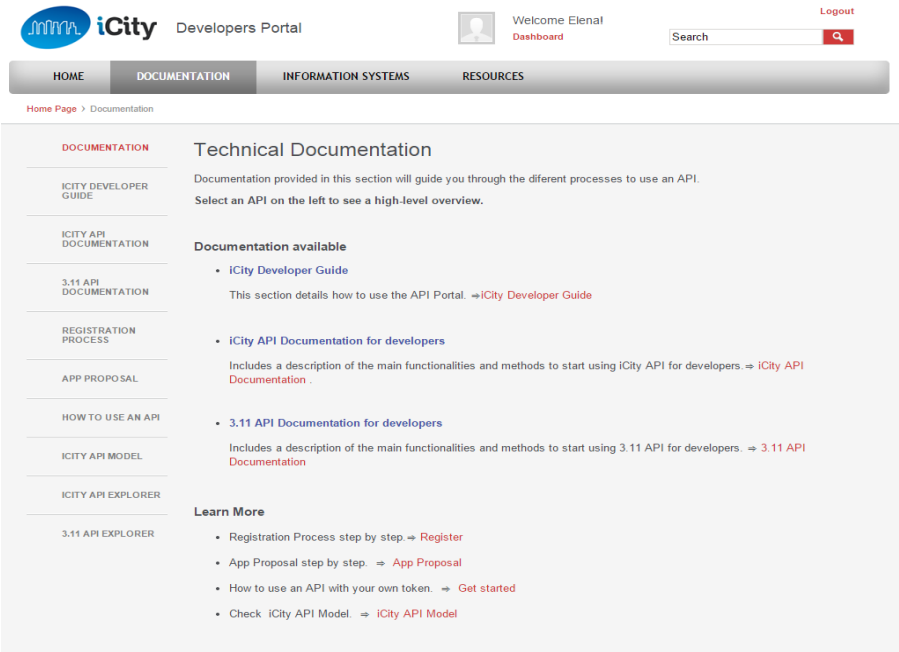

Figure 98 Developer documentation

8.7 iCITY API

8.7.1 Purpose

This section aims to collect the iCity API specifications to interact with the iCity platform which. It will be useful for developers of applications who want to participate and interact with the iCity platform.

8.7.2 Description

This iCity API expands the possibilities to obtain information, creation and control to the programmatic level of the system, in order to integrate the data to third-party applications, fitting different user models.

It includes the following functionalities to obtain data from the iCity network:

The public API only offers look-up methods (GET) to obtain information about a sensor or a group of them

- List all devices.
- Describe sensor.
- It is also available for platforms, cities or manufacturers.
- List all platforms/cities/manufacturers.
- Describe platform/city/manufacturer.
- List all devices from platform/city/manufacturer.
- Obtain observations from a sensor.
- Observation by samples.
- Observation by time interval.

In the next pages those functionalities are detailed.

8.7.3 Formats

This API supports several response formats, including XML, JSON, and PHP arrays. It checks the URL and look for the format, either as an extension or as a separate segment. By default, JSON is served.

This means the URLs can look like `http://ip/api/devices?format=xml`

Another possibility is specifying the format using an HTTP Accept header as `Accept: application/json.`

8.7.4 iCity API Architecture

Before moving to the API implementation stage, it should be clear what will functions contain and also the used methods and the input and output parameters in each case. Next, is defined the different functions for each object that will be implemented in the API.

The system is composed of four different objects: devices, Information Systems, cities, manufacturers. Being devices the basic element, the rest are groups of devices defined by the system.

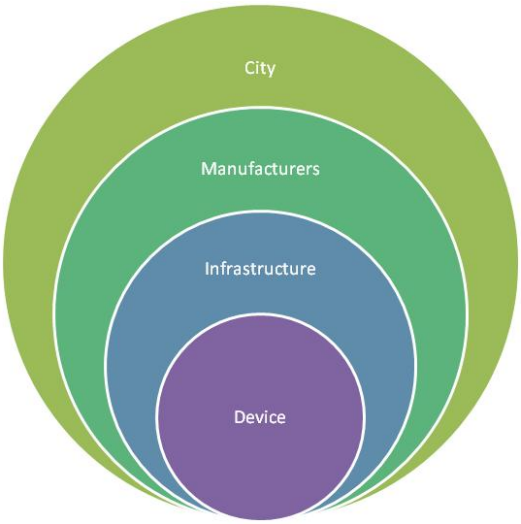

Figure 99 iCity API Architecture

Also it has been created the “collection” item that allows extracting information of a particular type of device, regardless of what platform or Information System it resides in order to obtain a global vision of the network.

8.7.4.1 *Devices*

The public API only offers look-up methods (GET) to obtain information about a device or a group of them. Devices are created, updated and deleted by the system or by the admin user.

List all devices

- Method: GET
- Resource: `/devices?deviceName={deviceName}&InformationSystemID={InformationSystemID&InformationSystemName={InformationSystemName}&manufacturerID={manufacturerID}&manufacturerName={manufacturerName}&cityID={cityID}&cityName={cityName}&propertyName={propertyName}`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
deviceName	*Device name*	*String*	*No*
Information SystemID	*Information System identification number*	*Integer*	*No*
Information SystemName	*Information System name*	*String*	*No*
manufacturerID	*Manufacturer identification number*	*Integer*	*No*
manufacturerName	*Manufacturer name*	*String*	*No*
cityID	*City identification number*	*Integer*	*No*
cityName	*City name*	*String*	*No*
propertyName	*Property name*	*String*	*No*

- Returns: A full list of all available devices objects in the specified format.

Function	**HTTP**	**Description**	**Input**	**Output**

	Method		**Parameters**	**Parameters**
List	GET	Lists all the devices registered on the system.	*Device name, Information SystemID, Information System name, ManufactureID, Manufacturer name, CityID, City name, Property*	Profile parameters of each sensor of the system.

Describe device

- Method: GET
- Resource: `/devices/{id}`

Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
id	Device identification number	Integer	Yes

- Returns: A full description of the sensor with ID `{id}`.

Function	**HTTP Method**	**Description**	**Input Parameters**	**Output Parameters**
Profile	GET	Device Information *sensorID*	*deviceID*	*deviceID* *platform* *latitude* *longitude* *offering*

				procedure *name* *city* *manufacturer* *properties*

8.7.4.2 ***Information Systems***

List all Information Systems

- Method: GET
- Resource:`/Information Systems?Information SystemName={Information SystemName}&manufacturerID={manufacturerID}&manufacturerName={manufacturerName}&cityID={cityID}&cityName={cityName}&propertyName={propertyName}`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
Information SystemName	*Information System name*	*String*	*No*
manufacturerID	*Manufacturer identification number*	*Integer*	*No*
manufacturerName	*Manufacturer name*	*String*	*No*
cityID	*City identification number*	*Integer*	*No*
cityName	*City name*	*String*	*No*
propertyName	*Property name*	*String*	*No*

- Returns: A full list of all available platform objects in the specified format.

Function	HTTP Method	Description	Input Parameters	Output Parameters
List	GET	Lists all the Information Systems registered on the system.	*Information System name, ManufacturerID, Manufacturer name, CityID, City name, Property*	Profile parameters of each system Information System.

Describe Information System

- Method: GET
- Resource: `/Information System/{id}`
- Request parameters:

Parameter	*Description*	*Type*	*Compulsory*
id	Information System identification number	Integer	Yes

- Returns: A full description of the platform with ID `{id}`.

Function	HTTP Method	Description	Input Parameters	Output Parameters
Profile	GET	Information System Information Information System *ID*	Information System *ID*	Information System *ID* *sos_nickname* *latitude* *longitude* *city* *manufacturer*

List all devices from Information System

- Method: GET
- Resource: `/ Information System /{id}/devices`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
id	Information System identification number	Integer	Yes

- Returns: A full list of all available devices from a certain platform with ID {id} in the specified format.

Function	**HTTP Method**	**Description**	**Input Parameters**	**Output Parameters**
List devices	GET	Lists all the devices that contain the Information System. *platformID*	Information System *ID*	Profile parameters of each sensor of Information System.

8.7.4.3 ***Cities***

List all cities

- Method: GET
- Resource:`/cities?cityName={cityName}&Information SystemID={Information SystemID}&Information SystemName={Information SystemName}&manufacturerID={manufacturerID}&manufacturerName={manufacturerName}&propertyName={propertyName}`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***

Information SystemID	*Information System identification number*	*Integer*	*No*
Information SystemName	*Information System name*	*String*	*No*
manufacturerID	*Manufacturer identification number*	*Integer*	*No*
manufacturerName	*Manufacturer name*	*String*	*No*
cityName	*City name*	*String*	*No*
propertyName	*Property name*	*String*	*No*

- Returns: A full list of all available city objects in the specified format.

Function	**HTTP Method**	**Description**	**Input Parameters**	**Output Parameters**
List	GET	Lists all the cities registered on the system.	*Information System ID, Information System name, Manufacturer ID, Manufacturer name, City name, Property*	Profile parameters of each city of the system.

Describe city

- Method: GET
- Resource: `/cities/{id}`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
id	City identification number	Integer	Yes

- Returns: A full description of the city with ID `{id}`.

Function	HTTP Method	Description	Input Parameters	Output Parameters
Profile	GET	City Information *cityID*	*cityID*	*cityID* *sos_nickname* *name* *longitude* *latitude* *description*

List all devices from city

- Method: GET
- Resource: `/cities/{id}/devices`
- Request parameters:
- Returns: A full list of all available devices from a certain city with ID `{id}` in the specified format.

Function	HTTP Method	Description	Input Parameters	Output Parameters
List devices	GET	Lists all the devices that contains the city *cityID*	*cityID*	Profile parameters of each sensor of the city.

8.7.4.4 ***Manufacturers***

List all manufacturers

- Method: GET
- Resource:`/manufacturers?manufacturerName={manufacturerName}&InformationSystemID={InformationSystemID}&InformationSystemName={InformationSystemName}&cityID={cityID}&citName={cityName}&propertyName={propertyName}`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
Information SystemID	*Information System identification number*	*Integer*	*No*
Information SystemName	*Information System name*	*String*	*No*
manufacturerID	*Manufacturer identification number*	*Integer*	*No*
CityID	*City identification number*	*Integer*	*No*
cityName	*City name*	*String*	*No*
propertyName	*Property name*	*String*	*No*

- Returns: A full list of all available manufacturers objects in the specified format.

Function	**HTTP Method**	**Description**	**Input Parameters**	**Output Parameters**
List	GET	Lists all the manufacturers registered on the system.	*Information System ID, Information System name, Manufacturer ID , CityID, City name, Property*	Profile parameters of each system's manufacturer.

Describe manufacturer

- Method: GET
- Resource: `/manufacturers/{id}`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
id	Manufacturer identification number	Integer	Yes

- Returns: A full description of the manufacturer with ID `{id}`.

Function	**HTTP Method**	**Description**	**Input Parameters**	**Output Parameters**
Profile	GET	Manufacturer information *manufacturerID*	*manufacturerID*	*manufacturerID* *sos_nickname* *name* *location* *website*

List all devices from manufacturer

- Method: GET
- Resource: `/manufacturers/{id}/devices`
- Request parameters:

Parameter	*Description*	*Type*	*Compulsory*
id	Manufacturer identification number	Integer	Yes

- Returns: A full list of all available devices from a certain manufacturer with ID `{id}` in the specified format.

Function	HTTP Method	Description	Input Parameters	Output Parameters
List devices	GET	Lists all the devices that contain the manufacturer. *manufacturerID*	*manufacturerID*	Profile parameters of each manufacturer's sensor.

8.7.4.5 *Collections*

List all collections

- Method: GET
- Resource: `/collections`
- Request parameters: none
- Returns: A full list of all available collections objects in the specified format.

Function	HTTP Method	Description	Input Parameters	Output Parameters
List	GET	Lists all the collections registered on the system.	*None*	Profile parameters of each system's collection.

Describe collection

- Method: GET
- Resource: `/collections/{id}`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
id	Collection identification number	Integer	Yes

- Returns: A full description of the manufacturer with ID `{id}`.

Function	**HTTP Method**	**Description**	**Input Parameters**	**Output Parameters**
Profile	GET	Collection information *collectionID*	*collectionID*	*collectionID* *name* *description* *creation_date* *userID*

List all devices from collection

- Method: GET
- Resource: `/collections/{id}/devices`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
id	Collection identification number	Integer	Yes

- Returns: A full list of all available devices from a certain manufacturer with ID `{id}` in the specified format.

Function	HTTP Method	Description	Input Parameters	Output Parameters
List devices	GET	Lists all the devices that contain the collection. *collectionID*	*collectionID*	Profile parameters of each collection's sensor.

8.7.5 Observations

List observations by samples

- Method: GET
- Resource:`/observations/last?id={id}&Information Systemid={Information Systemid}&property={property}&n={n}&filter={filter}`
- Request parameters:

Parameter	*Description*	*Type*	*Compulsory*
id	Device identification number	Integer	This is required if no **Information Systemid** is provided
Information Systemid	Information System identification number	Integer	This is required if no **id** is provided
property	URI of the requested observation	String	Yes

Filter	The structure of the filter string depends on the property of the observation request: urn:bus:expected_arrival: {stop};{line};{time} urn:bus:expected_arrival:ivr: {stop};{line};{time} urn:bus:sales_point: {stop;{salespoint} urn:infocity:topic: {NAME};{TAG} urn:infocity:office: {AREA};{NAME} urn:infocity:topic:data: {TOPIC} urn:infocity:office:data: {OFFICE} urn:infocity:trip: {name1=value1%26name2=value2...} urn:iris:state{token}	String	No
n	Number of requested samples	Integer	Yes

- Returns: Description of the observation samples.

Function	HTTP Method	Description	Input Parameters	Output Parameters
Samples	GET	Lists the N observations of the sensor *deviceID* and the magnitude *property*.	*deviceID* *property* *N* *Filter* *Information SystemID*	*time* *value* *units*

List observations by time interval

- Method: GET

- Resource:`/observations/interval?id={id}&Information Systemid={Information Systemid}&property={property}&from={from}&to={to}&filter={filter}`

- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
id	Device identification number	Integer	This is required if **no Information System**id is provided
Information Systemid	Information System identification number	Integer	This is required if no **id** is provided
property	URI of the requested observation	String	Yes
from	Start of the interval	ISO-8601 date (yyyy-mm-ddThh:mm:ss)	Yes
to	End of the interval	ISO-8601 date (yyyy-mm-ddThh:mm:ss)	Yes
filter	The structure of the filter string depends on the property of the observation request: urn:bus:expected_arrival: {stop};{line};{time} urn:bus:expected_arrival:ivr: {stop};{line};{time} urn:bus:sales_point: {stop;{salespoint} urn:infocity:topic: {NAME};{TAG} urn:infocity:office: {AREA};{NAME} urn:infocity:topic:data: {TOPIC} urn:infocity:office:data: {OFFICE} urn:infocity:trip: {name1=value1%26name2=value2...} urn:iris:state{token}	String	No

- Returns: Description of the observation samples.

Function	HTTP Method	Description	Input Parameters	Output Parameters
Interval	GET	Lists the observations of the device device*ID* and the magnitude *property*, since the time *from* to the time *to*.	*deviceID* *property* *from* *to* *Information Systemid* *filter*	*time* *value* *units*

8.7.5.1 ***Other tools***

List devices by geographical proximity

- Method: GET
- Resource: `/radius/{latitude}/{longitude}/{distance}`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
Latitude	Latitude coordinate	Float (10,6)	Yes
Longitude	Longitude coordinate	Float (10,6)	Yes
Distance	Radius if the area in kilometres	Integer	Yes

- Returns: Description of the observation samples.

Function	HTTP Method	Description	Input Parameters	Output Parameters
Radius	GET	Lists the devices by geographical proximity	*Longitude* *Latitude* *Distance*	*DeviceID*

8.8 3.11 API

8.8.1 Purpose

This section aims to collect the 3.11 API specifications to interact with the iCity platform. It will be useful for developers who want to participate and interact with iCity.

8.8.2 Description

This 3.11 API expands the possibilities to obtain information, creation and control of the system, in order to integrate the data to third-party applications, fitting different user models.

The public API offers look-up methods (GET) & (POST) to obtain and put information about suggestions and complaints.

8.8.3 Formats

This API supports several response formats, including XML, JSON, and PHP arrays. By default, JSON is served.

8.8.4 Functionalities

The main functionalities of 3.11 API are related with two different objects detailed in this section: services and requests.

8.8.4.1 *Services*

The public API only offers look-up methods (GET) to obtain information about a `service` or a group of them.

List all Services

- Method: GET
- Resource: `/services`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
Jurisdiction_id	*Jurisdiction identification number*	*Integer*	*Yes*

- Returns:

Function	**HTTP Method**	**Description**	**Input Parameters**	**Output Parameters**
List	GET	Lists of all services registered in to the jurisdiction.	*Jurisdiction ID*	Profile parameters of each service.

Describe a Service

- Method: GET

- Resource: `/services/{service_code}`
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
Jurisdiction_id	*Jurisdiction identification number*	*Integer*	*Yes*
Service_code	*Service identification code*	*String*	*Yes*

- Returns:

Function	**HTTP Method**	**Description**	**Input Parameters**	**Output Parameters**
Profile	GET	Service Information	*Jurisdiction ID*	Profile parameters of service with the service_code added.

8.8.4.1 ***Requests***

The public API offers look-up methods to post a request and track the status of that request and other requests.

Post a Request

- Method: POST
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
Jurisdiction_id	*Jurisdiction identification number*	*Integer*	*Yes*
service_code	*Service identification code*	*String*	*Yes*
location	*A full location*	*String*	*One*

parameter	*parameter must be submitted.*		*of lat & long or address_string or address_id*
attribute	*An array of key/value responses based on Service Definitions.*	*String*	*Yes*
lat	*Latitude using the (WGS84) projection.*	*String*	*No*
long	*Longitude using the (WGS84) projection*	*String*	*No*
address_string	*Human entered address or description of location.*	*String*	*No*
address_id	*The internal address ID used by a jurisdiction's master address repository or other addressing system.*	*Integer*	*No*
email	*The email address of the person submitting the request*	*String*	*No*
device_id	*The unique device ID of the device submitting the request. This is usually only used for mobile devices.*	*Integer*	*No*
account_id	*The unique ID for the user account of the person submitting the request*	*Integer*	*No*
first_name	*The given name of the person submitting the request*	*String*	*No*
last_name	*The family name of the person submitting the request*	*String*	*No*

phone	*The phone number of the person submitting the request*	*String*	*No*
description	*A full description of the request or report being submitted.*	*String*	*No*
media_url	*A URL to media associated with the request, eg an image.*	*String*	*No*

- Returns:

Function	**HTTP Method**	**Description**	**Input Parameters**	**Output Parameters**
Add	POST	Make a request to upload a new suggestion or complaint	*Jurisdiction ID, service_code, location parameter, email, lat, long, attribute, address string, address id, email, account id, first name, last name, phone, description, media url.*	Service_request_id, service_notice, account_id

List all Requests

- Method: GET

- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
Jurisdiction_id	*Jurisdiction identification number*	*Integer*	*Yes*
service_code	*Service identification code*	*String*	*Yes*
Service_request_id	*To call multiple Service*	*Integer*	*No*

	Requests at once, multiple service_request_id can be declared; comma delimited.		
Start_date	*Earliest datetime to include in search. When provided with end_date allows one to search for requests which have a requested_datetime that matches a given range, but may not span more than 90 days.* *Format: 2010-01-01T00:00:00Z.*	*String*	*No*
End_date	*Latest datetime to include in search. When provided with start_date, allows one to search for requests which have a requested_datetime that marches a given range, but may not span more than 90 days.* *Format: 2010-01-01T00:00:00Z.*	*String*	*No*
status	*Allows one to search for requests which have a specific status. This defaults to all statuses; can be declared multiple times, comma, delimited;* *Options: open, closed*		*No*

- Returns:

Function	HTTP Method	Description	Input Parameters	Output Parameters
List	GET	Lists of all requests registered in to the jurisdiction.	*Jurisdiction ID, service_code, service_request_id, start_date, end_date, status*	Profile parameters of all requests.

Describe Service Request

- Method: GET
- Request parameters:

Parameter	***Description***	***Type***	***Compulsory***
Jurisdiction_id	*Jurisdiction identification number*	*Integer*	*Yes*
Service_request_id	*To call multiple Service Requests at once, multiple service_request_id can be declared; comma delimited.*	*Integer*	*No*

- Returns:

Function	HTTP Method	Description	Input Parameters	Output Parameters
Profile	GET	Request Information *service_request_id*	*Jurisdiction ID, service_request_id,*	Service_request_id Status Status_notes Service_name Service_code Description

				Agency_responsible Requested_datetime Update_datetime Expected_datetime Address Address_id Lat Long media_url

8.9 Conclusions

During the first year of iCity project, WP4 has developed a set of tools that allows standard access to iCity Platform for any developer who wants to develop an iCity application. Like for example SOS Web service interface or SOS base SDK.

Using the SDK provides to developers the following key benefits:

- Reduced time-to-marked-reuse the complete Library for creating your applications.
- One (or as few as possible) code base(s) for all platforms and devices.
- End-to-end solution: server framework, communication layer and client framework.
- Network, content, device, and OS agnostic.
- Secure communication protocol.
- Easy operation and distribution of your application.

During the second year of the project, thanks to the inputs received from the pilots done in WP5, where a large number of developers asked for a simply and lighter interface, WP4 decided to create a REST API.

Using the iCity REST API for accessing to iCity Data, provides to developers the following key benefits:

- Web API easy to understand with a clear language for actions like GET, POST, PUT and DELETE.
- Network, content, device and OS agnostic.
- Lighter than other protocols. The answer can be represented in different formats (JSON, XML, etc.).
- One (or as few as possible) code base(s) for all platforms and devices.

At the end of the second year of the project, the new version of the iCity REST API included a list of new functionalities (iCity API v2) in order to provide access to the new open Information Systems which have new requirements for requesting the data.

This version permits to add different information in different formats with the aim of help to developers and users to request for a determined information like, for example, the expected arrival time of the bus for a concrete stop and line.

During the third year of the project, WP4 has continued making improvements in the iCity REST API, providing it the following main functionalities:

- **FILTER**: allows developers to reduce the calls number to obtain certain data. WP4 has done a cross-over data adding new parameters in the existing calls.
- **RADIUS:** allows developers to request for all the devices, services or events in an specific place indicating latitude, longitude and distance (in radius mode).
- **OPEN 311:** allows the access to Suggestions & Complains cities information systems, like Barcelona (IRIS), Lamia and Zaragoza.

The REST API is an alive API which includes new functionalities in order to adapt it to the new integrated Information Systems, helping the engagement community process, as it is required by WP2 and WP5.

During the fourth year, from M36 to M42, WP4 have been focused on the integration of new information systems with the iCity platform and the adaptation of the iCity API and its documentation to their requirements. In addition, WP4 has been working in the iCity Platform improvements and its maintenance.

Moreover WP4 is now involved in the deployment and publication process for the iCity APPShowCase.

9. APPENDIX I: iCity API Developers

"Linked Open Apps Ecosystem to open up innovation in smart cities"

Project Number: 297363

iCity API Documentation for Developers

Version: **1.2**

DOCUMENT HISTORY

Version	Date of issue	Status	Content and changes	Modified by
1.2	10/12/2014		Update with new contents	iCity

1. General Notes

This API expands the possibilities to obtain information, creation and control to the programmatic level of the system, in order to integrate the data to third-party applications, fitting different user models.

1.1 Base URL

The API follows the REST principles and is accessible over HTTP at http://icity-gw.icityproject.com:8080/developer/api/.

1.2 Authentication

All interactions with the API require an API Key. In order to pass it in a request, you must format the URL as http://icity-gw.icityproject.com:8080/developer/api/resource?apikey=random_string.

2. Resources

The system is composed of five different objects: devices, Information Systems, cities, manufacturers and collections. Being the devices the basic element, the rest are groups of devices. Information Systems, cities and manufacturers are defined by the system operators, while collections defined are by the user.

2.1. Devices

The public API only offers look-up methods (GET) to obtain information about a device or a group of them.

Devices are created, updated and deleted by the system or by the administrator.

2.1.1. List all devices

- Method: GET
- Resource:/devices?deviceName={deviceName}&Information SystemID={Information SystemID}&Information SystemName={Information SystemName}&manufacturerID={manufacturerID}&manufacturerName={manufacturerName}&cityID={cityID}&cityName={cityName}&propertyName={propertyName}
- Request parameters:

Parameter	Description	Type	Compulsory
deviceName	Device name	String	No
Information SystemID	Information System identification number	Integer	No
Information SystemName	Information System name	String	No
manufacturerID	Manufacturer identification number	Integer	No
manufacturerName	Manufacturer name	String	No
cityID	City identification number	Integer	No
cityName	City name	String	No
propertyName	Property name	String	No

- Returns: A full list of all available devices objects in the specified format

Sample JSON response


```
[
	{
		"deviceID":"1",
		"name":"SNSBG1",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.177891",
		"latitude":"51.563751",
		"properties":
		[
		"urn:air_qualityNO2",
		"urn:air_qualitySO2"
		]
	},
	{
		"deviceID":"2",
		"name":"SNSBG2",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.132857",
		"latitude":"51.529388",
		"properties":
		[
		"urn:air_qualityNO2",
		"urn:air_qualityPM10"
		]
	}
]
```


2.1.2 Describe device

- Method: GET
- Resource: /devices/{id}
- Request parameters:

Parameter	Description	Type	Compulsory
Id	Device identification number	Integer	Yes

- Returns: A full description of the device with IF {id}

Sample JSON response


```
{
       "deviceID":"1",
       "name":"SNSBG1",
       "cityID":"3",
       "Information SystemID":"1",
       "manufacturerID":"7",
       "longitude":"0.177891",
       "latitude":"51.563751",
       "properties":
       [
       "urn:air_qualityNO2",
       "urn:air_qualitySO2"
       ]
}
```


2.2 Information Systems

2.2.1 List all Information Systems

- Method: GET
- Resource:/Information Systems?Information SystemName={Information SystemName}&manufacturerID={manufacturerID}&manufacturerName={manufacturerName}&cityID={cityID}&cityName={cityName}&propertyName={propertyName}
- Request parameters:

Parameter	Description	Type	Compulsory
Information SystemName	Information System name	String	No
manufacturerID	Manufacturer identification number	Integer	No
manufacturerName	Manufacturer name	String	No
cityID	City identification number	Integer	No
cityName	City name	String	No
propertyName	Property name	String	No

- Returns: A full list of all available platform objects in the specified format.

Sample JSON response


```
[
	{
	"Information SystemID":"1",
	"name":"PLTLDN001",
	"cityID":"3",
	"manufacturerID":"7",
	"latitude":"51.617329",
	"longitude":"-0.460826"
	},
	{
	"Information SystemID":"2",
	"name":"PLTBOL0001",
	"cityID":"9",
	"manufacturerID":"11",
	"latitude":"44.564266",
	"longitude":"11.220111"
	},
	{
	"Information SystemID":"3",
	"name":"PLTBOL0002",
	"cityID":"9",
	"manufacturerID":"11",
	"latitude":"44.594299",
	"longitude":"11.220111"
	}
]
```


2.2.2 Describe Information System

- Method: GET
- Resource: /Information Systems/{id}
- Request parameters:

Parameter	Description	Type	Compulsory
Id	Information System identification number	Integer	Yes

- Returns: A full description of the Information System with ID {id}.

Sample JSON response

```
{
	"Information SystemID":"1",
	"name":"PLTLDN001",
	"cityID":"3",
	"manufacturerID":"7",
	"latitude":"51.617329",
	"longitude":"-0.460826"
}
```


2.2.3 List all devices from Information System

- Method: GET
- Resource: /Information Systems/{id}/devices
- Request parameters:

Parameter	Description	Type	Compulsory
Id	Information System identification number	Integer	Yes

- Returns: A full list of available devices from a certain Information System with ID {id} in the specified format.

Sample JSON response


```
[
	{
		"deviceID":"1",
		"name":"SNSBG1",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.177891",
		"latitude":"51.563751",
		"properties":
		[
		"urn:air_qualityNO2",
		"urn:air_qualitySO2"
		]
	},
	{
		"deviceID":"2",
		"name":"SNSBG2",
		"cityID":"3",
		"Information SystemID":"1",
```

```
                "manufacturerID":"7",
                "longitude":"0.132857",
                "latitude":"51.529388",
                "properties":
                [
                "urn:air_qualityNO2",
                "urn:air_qualityPM10"
                ]
        }
]
```


2.3 Cities

2.3.1 List all cities

- Method: GET
- Resource:/cities?cityName={cityName}&Information SystemID={Information SystemID}&Information SystemName={Information SystemName}&manufacturerID={manufacturerID}&manufacturerName={manufacturerName}&propertyName={propertyName}
- Request parameters:

Parameter	Description	Type	Compulsory
Information SystemID	Information System identification number	Integer	No
Information SystemName	Information System name	String	No
manufacturerID	Manufacturer identification number	Integer	No
manufacturerName	Manufacturer name	String	No
cityName	City name	String	No
propertyName	Property name	String	No

- Returns: A full list of city objects in specified format.

Sample JSON response


```
[
        {
                "cityID":"3",
                "name":"London",
```

```
                "longitude":"-0.119800",
                "latitude":"51.511189"
        },
        {
                "cityID":"7",
                "name":"Barcelona",
                "longitude":"2.170030",
                "latitude":"41.387058"
        },
        {
                "cityID":"8",
                "name":"Genova",
                "longitude":"8.949722",
                "latitude":"41.416672"
        },
        {
                "cityID":"9",
                "name":"Bologna",
                "longitude":"11.351389",
                "latitude":"44.507500"
        }
]
```


2.3.2 Describe a city

- Method: GET
- Resource: /cities/{id}
- Request parameters:

Parameter	Description	Type	Compulsory
Id	City identification number	Integer	Yes

- Returns: A full description of the city with ID {id}.

Sample JSON response


```
{
        "cityID":"9",
        "name":"Bologna",
        "longitude":"11.351389",
        "latitude":"44.507500"
        "description":""
}
```

2.3.3 List all devices from city

- Method: GET
- Resource: /cities/{id}/devices
- Request parameters:

Parameter	Description	Type	Compulsory
Id	city identification number	Integer	Yes

- Returns: A full list of available devices from a certain city with ID {id} in the specified format.

Sample JSON response


```
[
	{
		"deviceID":"1",
		"name":"SNSBG1",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.177891",
		"latitude":"51.563751",
		"properties":
		[
		"urn:air_qualityNO2",
		"urn:air_qualitySO2"
		]
	},
	{
		"deviceID":"2",
		"name":"SNSBG2",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.132857",
		"latitude":"51.529388",
		"properties":
		[
		"urn:air_qualityNO2",
		"urn:air_qualityPM10"
		]
	}
]
```

2.4 Manufacturers

2.4.1 List all manufacturers

- Method: GET
- Resource:/manufacturers?manufacturerName={manufacturerName}&Information SystemID={Information SystemID}&Information SystemName={Information SystemName}&cityID={cityID}&cityName={cityName}&propertyName={propertyName}
- Request parameters:

Parameter	Description	Type	Compulsory
Information SystemID	Information System identification number	Integer	No
Information SystemName	Information System name	String	No
manufacturerID	Manufacturer identification number	Integer	No
CityID	City identification number	Integer	No
cityName	City name	String	No
propertyName	Property name	String	No

- Returns: A full list of available manufacturer objects in specified format.

Sample JSON response


```
[
	{
		"manufacturerID":"7",
		"name":"London",
		"location":"London"
	},
	{
		"manufacturerID":"10",
		"name":"PSB",
		"location":"Barcelona"
	},
	{
		"manufacturerID":"11",
		"name":"Bologna",
		"location":"Bologna"
	}
]
```

2.4.2 Describe manufacturer

- Method: GET
- Resource: /manufacturers/{id}
- Request parameters:

Parameter	Description	Type	Compulsory
Id	Manufacturer identification number	Integer	Yes

- Returns: A full list description of the manufacturer with ID {id}.

Sample JSON response

```
{
	"manufacturerID":"10",
	"name":"PSB",
	"location":"Barcelona",
	"website":"",
	"description":"PSB"
}
```

2.4.3 List all devices from manufacturer

- Method: GET
- Resource: /manufacturers/{id}/devices
- Request parameters:

Parameter	Description	Type	Compulsory
Id	Manufacturer identification number	Integer	Yes

- Returns: A full list of available devices from a certain manufacturer with ID {id} in the specified format.

Sample JSON response

```
[
	{
		"deviceID":"1",
		"name":"SNSBG1",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.177891",
		"latitude":"51.563751",
		"properties":
		[
			"urn:air_qualityNO2",
			"urn:air_qualitySO2"
		]
	},
	{
		"deviceID":"2",
		"name":"SNSBG2",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.132857",
		"latitude":"51.529388",
		"properties":
		[
			"urn:air_qualityNO2",
			"urn:air_qualityPM10"
		]
	}
]
```


2.5 Collections

2.5.1 List all collections

- Method: GET
- Resource: /collections
- Request parameters: none
- Returns: A full list of available collections objects in specified format.

Sample JSON response


```
[
	{
		collectionID":"1",
		name":"My favourite sensors",
		description":"A collection of my favourite sensors",
		creation_date":"2013-01-20 16:55:30",
		userID":"1"
```

```
        },
        {
                collectionID":"2",
                name":"Parking sensors",
                description":"Street parking sensors",
                creation_date":"2013-01-20 16:58:30",
                userID":"1"
        },
        ...
]
```

2.5.2 Describe collections

- Method: GET
- Resource: /collections/{id}
- Request parameters:

Parameter	Description	Type	Compulsory
Id	collection identification number	Integer	Yes

- Returns: A full list description of the collection with ID {id}.

Sample JSON response

```
{
        collectionID":"1",
        name":"My favourite sensors",
        description":"A collection of my favourite sensors",
        creation_date":"2013-01-20 16:55:30"
}
```

2.5.3 List all devices from a collection

- Method: GET
- Resource: /collections/{id}/devices
- Request parameters:

Parameter	Description	Type	Compulsory
Id	Collection identification number	Integer	Yes

- Returns: A full list of available devices from a certain collection with ID {id} in the specified format.

Sample JSON response


```
[
	{
		"deviceID":"1",
		"name":"SNSBG1",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.177891",
		"latitude":"51.563751",
		"properties":
		[
		"urn:air_qualityNO2",
		"urn:air_qualitySO2"
		]
	},
	{
		"deviceID":"2",
		"name":"SNSBG2",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.132857",
		"latitude":"51.529388",
		"properties":
		[
		"urn:air_qualityNO2",
		"urn:air_qualityPM10"
		]
	}
] [
	{
		"deviceID":"1",
		"name":"SNSBG1",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.177891",
		"latitude":"51.563751",
		"properties":
		[
		"urn:air_qualityNO2",
		"urn:air_qualitySO2"
		]
	},
	{
		"deviceID":"2",
		"name":"SNSBG2",
		"cityID":"3",
		"Information SystemID":"1",
		"manufacturerID":"7",
		"longitude":"0.132857",
```

```
            "latitude":"51.529388",
            "properties":
            [
            "urn:air_qualityNO2",
            "urn:air_qualityPM10"
            ]
      }
]
```


2.6 Observations

2.6.1 List observations by samples

- Method: GET
- Resource:/observations/last?id={id}&Information Systemid={Information Systemid}&property={property}&n={n}&filter={filter}
- Request parameters:

Parameter	Description	Type	Compulsory
id	Device identification number	Integer	This is required if no Information Systemid is provided
Information Systemid	Information System identification number	Integer	This is required if no id is provided
property	URI of the requested observation	String	Yes
Filter	The structure of the filter string depends on the property of the observation request: urn:bus:expected_arrival: {stop};{line};{time} urn:bus:expected_arrival:ivr: {stop};{line};{time} urn:bus:sales_point: {stop;{salespoint} urn:infocity:topic: {NAME};{TAG} urn:infocity:office: {AREA};{NAME} urn:infocity:topic:data: {TOPIC} urn:infocity:office:data: {OFFICE} urn:infocity:trip: {name1=value1%26name2=value2...} urn:iris:state{token}	String	No
n	Number of requested samples	Integer	Yes

- Returns: Description of the observation samples

Sample JSON response

```
[
	{
		"time":"2013-01-19T17:49:35",
		"value":14.306,
		"units":"C"
	},
	{
		"time":"2013-01-19T18:09:41",
		"value":14.187,
		"units":"C"
	},
	{
		"time":"2013-01-19T18:29:52",
		"value":14.068,
		"units":"C"
	},
...
]
```


2.6.2 List observations by samples with filters

- Method: GET
- Resource:/observations/interval?id={id}&Information Systemid={Information Systemid}&property={property}&from={from}&to={to}&filter={filter}
- Request parameters:
- Returns: Description of the observation samples

Sample JSON response


```
[
	{
	 "time":"2014-01-22T11:56:41",
	 "value":"14.306",
	 "units":"C"
	}
]
```


2.6.3 List observations by time interval

- Method: GET

- Resource: /observations/interval?id={id}&property={property}&from={from}&to={to}
- Request parameters:

Parameter	Description	Type	Compulsory
id	Device identification number	Integer	This is required if no Information Systemid is provided
Information Systemid	Information System identification number	Integer	This is required if no id is provided
property	URI of the requested observation	String	Yes
from	Start of the interval	ISO-8601 date (yyyy-mm-ddThh:mm:ss)	Yes
to	End of the interval	ISO-8601 date (yyyy-mm-ddThh:mm:ss)	Yes
filter	The structure of the filter string depends on the property of the observation request: urn:bus:expected_arrival: {stop};{line};{time} urn:bus:expected_arrival:ivr: {stop};{line};{time} urn:bus:sales_point: {stop;{salespoint} urn:infocity:topic: {NAME};{TAG} urn:infocity:office: {AREA};{NAME} urn:infocity:topic:data: {TOPIC} urn:infocity:office:data: {OFFICE} urn:infocity:trip: {name1=value1%26name2=value2...} urn:iris:state{token}	String	No

- Returns: Description of the observation samples
-

Sample JSON response

```
[
	{
		"time":"2013-01-19T17:49:35",
		"value":14.306,
		"units":"C"
	},
	{
		"time":"2013-01-19T18:09:41",
		"value":14.187,
		"units":"C"
```

```
        },
        {
                "time":"2013-01-19T18:29:52",
                "value":14.068,
                "units":"C"
        },
...
]
```


2.7 Other tools

2.7.1 List devices by geographical proximity

- Method: GET
- Resource: /radius/{latitude}/{longitude}/{distance}
- Request parameters:

Parameter	Description	Type	Compulsory
Latitude	Latitude coordinate	Float (10,6)	Yes
Longitude	Longitude coordinate	Float (10,6)	Yes
distance	Radius if the area in kilometres	Integer	Yes

- Returns: Description of the observation samples

Sample JSON response


```
[
        {
                "deviceID":"1",
                "name":"SNSBG1",
                "cityID":"3",
                "Information SystemID":"1",
                "manufacturerID":"7",
                "longitude":"0.177891",
                "latitude":"51.563751",
                "properties":
                [
                        "urn:air_qualityNO2",
                        "urn:air_qualitySO2"
                ]
        },
        {
                "deviceID":"2",
                "name":"SNSBG2",
                "cityID":"3",
                "Information SystemID":"1",
```

```
                "manufacturerID":"7",
                "longitude":"0.132857",
                "latitude":"51.529388",
                "properties":
                [
                        "urn:air_qualityNO2",
                        "urn:air_qualityPM10"
                ]
        }
]
```

10. APPENDIX II: 3.11 API Developers

"Linked Open Apps Ecosystem to open up innovation in smart cities"

Project Number: 297363

3.11 API Documentation for Developers

Version: **0.1**

DOCUMENT HISTORY

Version	Date of issue	Status	Content and changes	Modified by
0.1	10/12/2014		Creation	iCity

1. General Notes

This API expands the possibilities to obtain information, creation and control to the programmatic level of the system, in order to integrate the data to third-party applications, fitting different user models.

1.1 Jurisdiction_id

As a means to allow this API to distinguish multiple jurisdictions within one API endpoint we've included a "jurisdiction_id" which will be the unique identifier for cities:

- **Jurisdiction_id=1**: IRIS Barcelona
- **Jurisdiction_id=2:** Issue Reporting Lamia
- **Jurisdiction_id=3:** Suggestions & Complaints Zaragoza
- **Jurisdiction_id=4:** IRIS OT

2. Resources

The system is composed of two different objects: services and requests.

2.1. Services

The public API only offers look-up methods (GET) to obtain information about a service or a group of them.

2.1.1. List all Services

- Method: GET
- Resource:/services
- Request parameters:

Parameter	Description	Type	Compulsory
Jurisdiction_id	Jurisdiction identification number	Integer	Yes

- Returns: List of all services registered into the jurisdiction.

Sample JSON response


```
[

      {

              "service_code":"001",
              "service_name":"Service 001",
              "description":"Service 001 description” ,
              "metadata":true ,
              "group":"group 1” ,
              "keywords":"keyword1, keyword2” ,
              "type":"realtime”},
     {

              "service_code":"002",
              "service_name":"Service 002",
              "description":"Service 002 description” ,
              "metadata":true ,
              "group":"group 1” ,
```

```
            "keywords":"keyword3, keyword4” ,
            "type":"realtime”},

]
```


2.1.2 Get Service Definition

- Method: GET
- Resource: /services/{service_code}
- Request parameters:

Parameter	Description	Type	Compulsory
Jurisdiction_id	Jurisdiction identification number	Integer	Yes
Service_code	Serivce identification code	String	Yes

- Returns: List of attributes associated with a service_code.

Sample JSON response


```
{
      "service_code":"001",
      "attibutes":[
          {
                "variable":true,
                "code":"WHISHETN",
                "datatype":"singlevaluelist",
                "required":true,
                "datatype_description":" single value list ",
                "order":"",
                "description":"Tickets",
                "values":[
```

```
                        {
                                "key":1
                                "name":"value1"
                        },
        {
                                "key":2
                                "name":"value2"
                        }
                ]
            }
        ]
 }
```

2.1. Requests

The public API offers look-up methods to post a request and track the status of that request and other requests

2.1.1. Post a request

- Method: POST
- Request parameters:

Parameter	Description	Type	Compulsory
Jurisdiction_id	Jurisdiction identification number	Integer	Yes
Service_code	Serivce identification code	String	Yes
Location parameter	A full location parameter must be submitted.	String	One of lat & ling or address_string/id
Attribute	An array of key/value responses based on Service Definitions.	String	Yes
Lat	Latitude using de (WGS84) projection	String	No
Long	Longitude using de (WGS84) projection	String	No
Address_string	Human entered address or description of location	String	No

Adress_id	The internal address ID used by a jurisdiction's master address repository or other addressing system.	Integer	No
Email	The email address of the person submitting the request	String	No
Device_id	The unique device ID of the device submitting the request. This is usually only used for mobile devices.	Integer	No
Account_id	The unique ID for the user account of the person submitting the request	Integer	No
First_name	The given name of the person submitting the request	String	No
Last_name	The family name of the person submitting the request	String	No
phone	The phone number of the person submitting the request	String	No
description	A full description of the request or report being submitted.	String	No
Media_url	A URL to media associated with the request, eg an image.	String	No

Sample JSON response

```
[
        {
                "service_request_id":26585,
                "service_notice":"Service notice",
                "account_id":null
        }
]
```

2.1.1. Get Service Requests

- Method: GET
- Resource: /requests
- Request parameters:

Parameter	Description	Type	Compulsory
Jurisdiction_id	Jurisdiction identification number	Integer	Yes
Service_code	Serivce identification code	String	Yes
Service_request_id	To call multiple Service Requests at once, multiple	Integer	No

	service_request_id can be declared; comma delimited		
Start_date	Earliest datetime to include in search. When provided with end_date, allows one to search for requests which have a requested_datetime that matches a given range, but may not span more than 90 days. Format: 2010-01-01T00:00:00Z.	String	No
End_date	Latest datetime to include in search. When provided with start_date, allows one to search for requests which have a requested_datetime that matches a given range, but may not span more than 90 days. Format: 2010-01-01T00:00:00Z.	String	No
Status	Allows one to search for requests which have a specific status. This defaults to all statuses; can be declared multiple times, comma delimited; Options: open, closed	String	No

- Returns: List of all services requests registered into the jurisdiction.

Sample JSON response


```
[

	{

		"service_request_id":1,
		"status":"closed",
		"status_notes":"Duplicate request”,
		"service_name":"service name”,
		"service_code":"001”,
		"description":null,
		"agency_responsible":null,
		"requested_datetime":"2014-02-10 12:10:13",
		"updated_datetime":"2014-02-10 12:10:13",
		"expected_datetime":"2014-02-10 12:10:13",
		"address":null,
		"address_id":545483,
		"zipcode":12345 ,
		"lat": 37.762221815,
		"long": -122.4651145 ,
		"media_url":null

	},

	{

		"service_request_id":2,
		"status":"open",
```

```
            "status_notes":null,
            "service_name":"service name”,
            "service_code":"001”,
            "description":null ,
            "agency_responsible": null ,
            "requested_datetime":"2014-02-10 12:10:13",
            "updated_datetime":"2014-02-10 12:10:13",
            "expected_datetime":"2014-02-10 12:10:13",
            "address":null,
            "address_id":545483,
            "zipcode":12358,
            "lat": 37.762221815,
            "long": -122.4651145,
            "media_url":null

      },

]
```


2.1.1. Get Service Request

- Method: GET
- Resource: /requests
- Request parameters:

Parameter	Description	Type	Compulsory
Jurisdiction_id	Jurisdiction identification number	Integer	Yes
Service_request_id	To call multiple Service Requests at once, multiple service_request_id can be declared; comma delimited	Integer	No

- Returns: The current status of an individual request.

Sample JSON response


```
[

      {

            "service_request_id":1,
            "status":"closed",
            "status_notes":"Duplicate request”,
            "service_name":"service name”,
            "service_code":"001”,
            "description":null,
            "agency_responsible":null,
```

```
            "requested_datetime":"2014-02-10 12:10:13",
            "updated_datetime":"2014-02-10 12:10:13",
            "expected_datetime":"2014-02-10 12:10:13",
            "address":null,
            "address_id":545483,
            "zipcode":12345 ,
            "lat": 37.762221815,
            "long": -122.4651145 ,
            "media_url":null

      }

]
```

11. APPENDIX III: iCity Portal User's Guide

"Linked Open Apps Ecosystem to open up innovation in smart cities"

Project Number: 297363

iCity Developer Guide	
Version:	**1.1**

DOCUMENT HISTORY

Version	Date of issue	Status	Content and changes	Modified by
1.1	10/12/2014		Final	iCity

Abbreviations and Acronyms

Acronym	Description
API	Application Programming Interface
HTTP	Hyper Text Transfer Protocol
OOB	Out Of Band
RBAC	Role Based Access Control
URL	Uniform Resource Locator
WADL	Web Application Description Language

1. Getting started with the API portal

1.1 About the API portal

1.1.1 Logging in

To log in to the API Portal:

Open your web browser and navigate to the following address:

http://icity-devp.icityproject.com

Click Login at the top of the browser window and then enter your Username and Password (if you do not have a login, please signup). To find more info on the Registration Process Scenario, click the link on our documentation page).

The Dashboard is displayed:

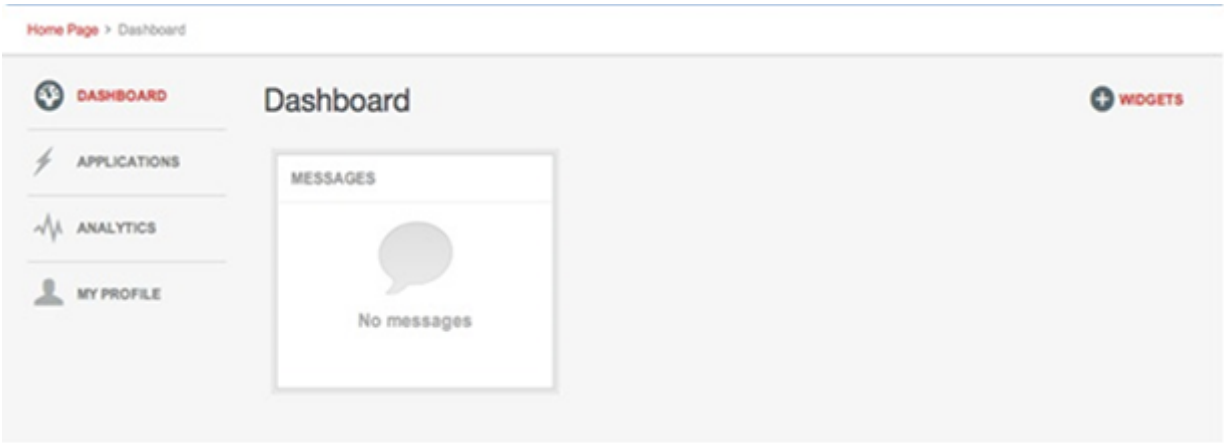

Figure 100 The iCity Dashboard

1.2 Working with the Dashboard

- The Dashboard is the interface for developers.
- Each individual user can personalize the Dashboard.

- Once you log in, the Dashboard page is displayed by default. You can quickly return to the Dashboard at any time by clicking the Dashboard link below the Welcome message at the top of the browser.

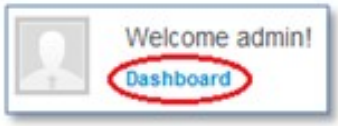

Figure 101 Dashboard link in browser

1.2.1 Customize your Dashboard widgets

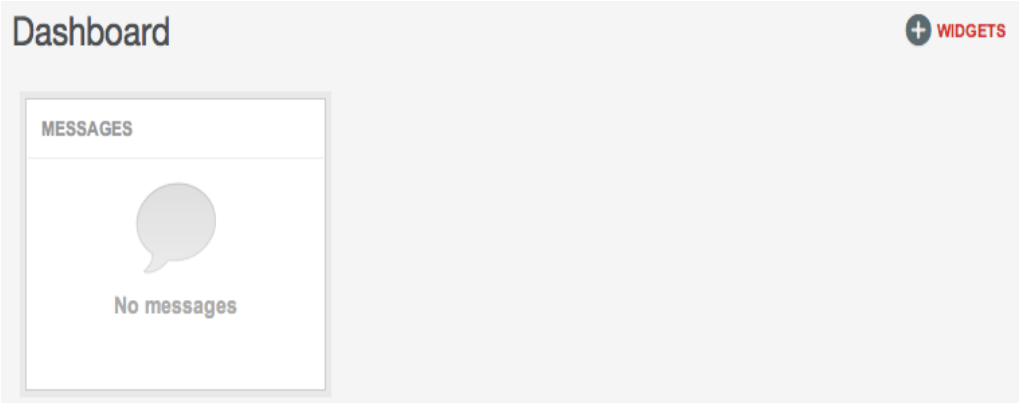

Figure 102 Dashboard Widgets

From the Dashboard, click [**+ WIDGETS**] to open the widget area.

- To add a widget, drag it into the Dashboard area.
- To remove a widget, click [**X**] on the top left corner of the widget.
- To configure a widget, point to the widget title and then click the gear icon on the right side of the title bar. Every widget has its own configuration settings. Click [**Done**] to save the configuration changes.

When you have finished customizing your Dashboard, click [**- WIDGETS**] to close the widget area.

Administrators can set which dashboard widgets are active. The widgets available in the Dashboard section are detailed in the following lines:

- **Messages:** This section shows the messages that a developer received from an Administrator or Manager User.
- **My APPS:** List of applications created by the developer with their availability to access to APIs.
- **Latest API Docs**: List of latest documents uploaded to the portal.

1.3 Updating Account Information

- To change the icon displayed beside the Dashboard link:
  - From within My Profile, click [**Choose File**]. A file upload dialogue box displays.
  - Locate a jpeg/jpg file on your local system and click [**Open**]. The file is uploaded to the API Portal.
  - Click [**Save**]. The icon beside the Dashboard link is updated with your file.

1.3.1 Developers can change their own passwords via the Manage My Profile page

Your personal account information can be changed at any time - for example, to reset your login and password information.

- To update your account information:
  - From the Dashboard, select My Profile from the navigation sidebar.
  - Update the fields as required. Note that if you want to change the password, you must also enter your current password.
  - Click [**Save**] when done.

Note: You will need to click on a link before the uploaded icon is displayed.

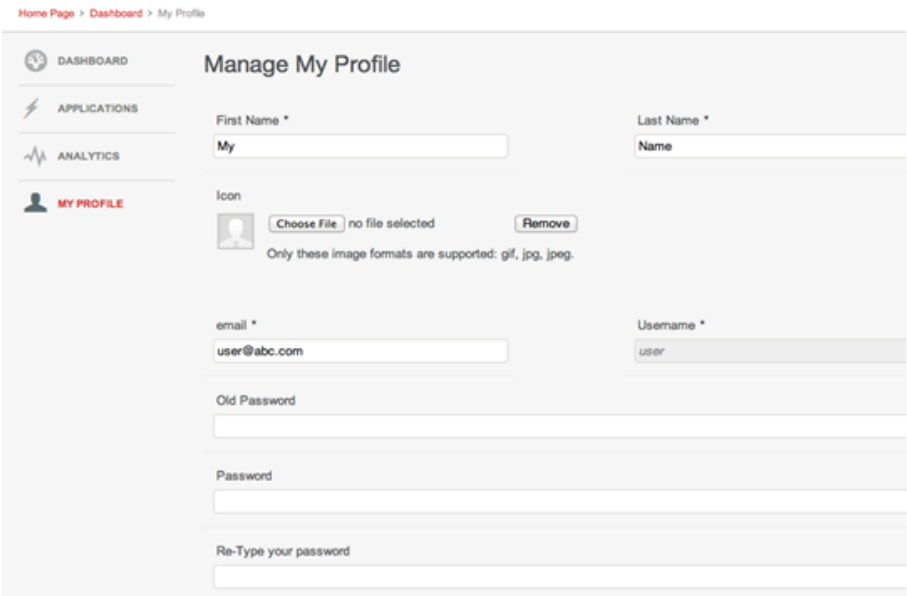

Figure 103 The iCity Dashboard

- Log in to the iCity API Portal.
- Click **My Profile** in the navigation sidebar.
- Enter your current password in the **Current Password** field.
- Enter the new password in both the **New Password** and **Re-Enter your password** fields.
- Click [**Save**] when you are done.

2. Reporting

2.1 Developer Reports

Developers have access to both application reports and API reports. In addition, each report offers two views located on two separate tabs: Usage and Latency.

The Application reports allow developers to:

- View usage for an application: Select an application from Application drop- down. The graph shows total API queries/requests (a.k.a. “hits”) for that application.
- Top graph: Shows all hits against all the APIs the application uses.
- Middle graph: Shows, of the total hits, how many successfully received a reply (i.e., resulted in a successful transaction).
- Bottom graph: Shows, of the total hits, how many did not receive a reply (i.e., resulted in an error).
- View latency for an application: Select an application from Application drop- down. The graph shows average latency for that application.
- Top graph: Shows the time it takes for an application request to enter the API Proxy, get passed to the back-end API(s), and then leave the API Proxy.
- Bottom graph: Shows the time it takes for a request to be processed by the API Proxy.

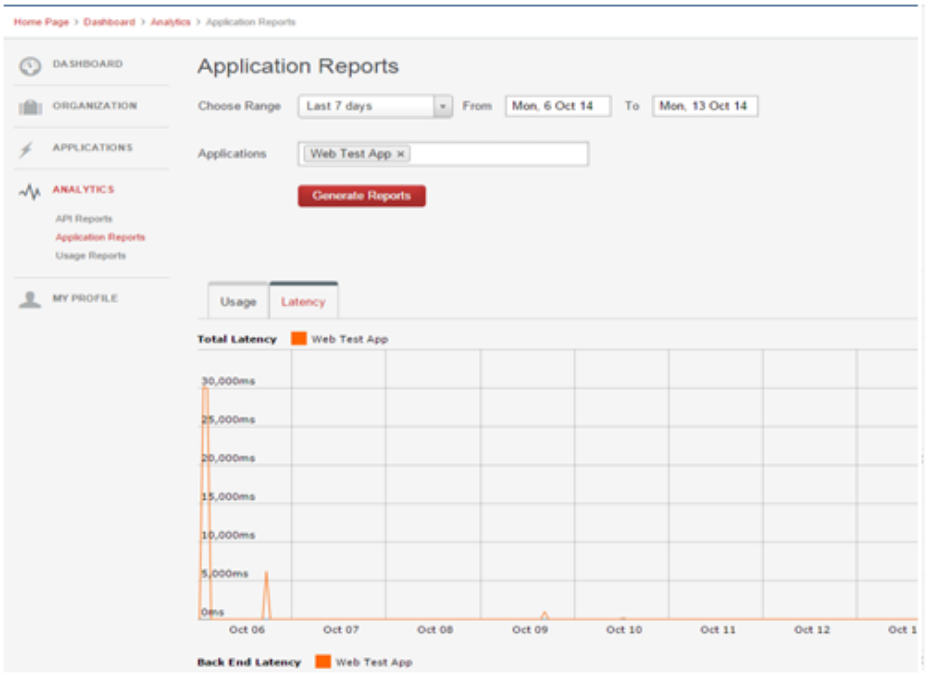

Figure 104 API reports

<u>The API reports allow developers to:</u>

- **View usage for an API**: Select an API from the API drop-down (note that the API drop-down will be populated only with APIs to which the developer has access). The graph shows hits against that API.
- **Top graph**: Shows total hits against the API from all the developer's applications.
- **Middle graph**: Shows, of the total hits, how many successfully received a reply (i.e., resulted in a successful transaction).
- **Bottom graph**: Shows, of the total hits, how many did not receive a reply (i.e., resulted in an error).
- **View latency for an API**: Select an API from API drop-down. The graph shows average latency of the API for all the developer's applications.
- **Top graph**: Shows the total, round trip time for an application request to enter the API Proxy, go to the back-end API(s), and then pass back through the API Proxy.

- **Bottom graph**: Shows the time it takes for a request to be processed by the API Proxy.

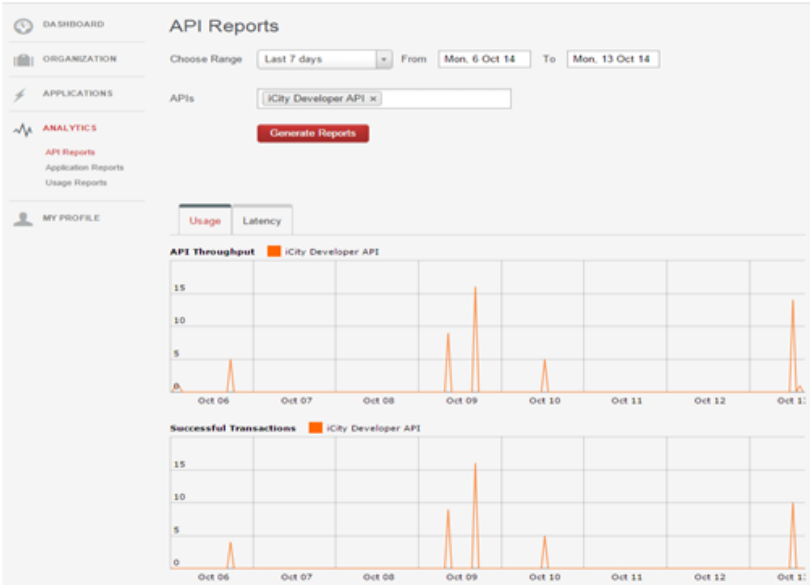

Figure 105 Application Reports

2.2 Usage Reports

This page provides a high level view of Account Plan usage by organization.

- To view usage reports:
    - From the Dashboard, select **Analytics** > **Usage Reports** from the navigation sidebar. The Usage Reports page displays.

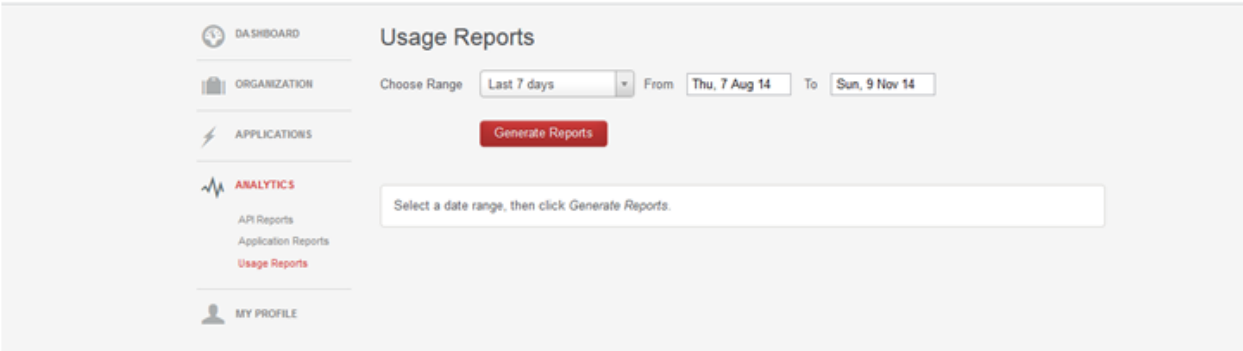

Figure 106 Usage Reports Dashboard

In the **Choose Range** drop-down, select the date range, from **Last 24 hours, Last 7 days, Last 30 days**, and **Last 365 days**. Alternatively, you may select specific dates in the **From** and **To** fields.

If you belong to more than one organization, choose the organization to report on from the **Organization** drop-down.

Click [**Generate Reports**] to view the report.

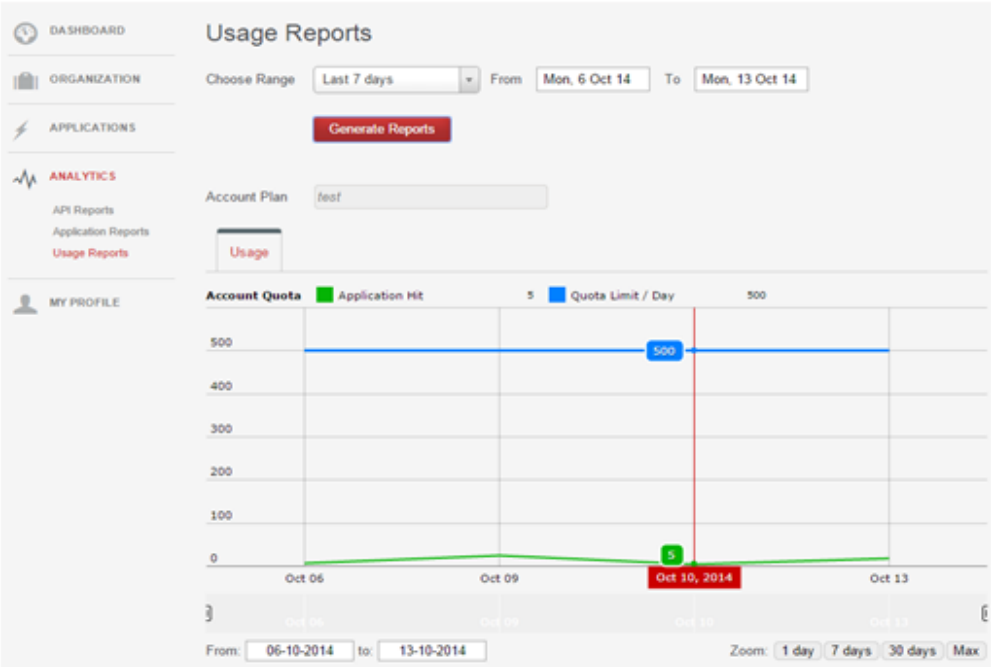

Figure 107 Usage Reports Generator

3. Developer Functionality

This section describes how developers will use the API Portal.

The first developer to register for your organization will also have the role of Organization Admin.

3.1 Developer Roles

The following “Developer” user roles are preconfigured on the iCity API Portal:

- **OrgAdmin**: The owner of an organization. This is typically a third-party user that signs up for an account on the iCity API Portal using the Registration Form. This person is responsible for managing his or her own organization and is usually the only developer or the first one to register for the organization.
- **Developer**: A user that has been invited to join the iCity API Portal by an organization owner (OrgAdmin). These users are enrolled under the OrgAdmin's account. Developers are responsible for creating and managing new applications.

3.2 Register for an Account

The first thing developers need to do is register for an account. They do this by signing up on the iCity API Portal and completing a registration form.

Note: The iCity API Portal does not permit registration of duplicate organization names. As a result, once the first developer from an organization has registered an account with the portal, subsequent developers from that same organization require an invitation to be registered.

3.2.1 To register an account

On the API Portal home page, click **Signup** in the upper right corner. Complete the registration dialogue as follows. Tip: Advance to the next tab either by clicking [**Next Step**] or by clicking the tab title.

Note: Administrators can configure which tabs are available on the registration dialogue so not see all of the tabs described below may be visible.

Tab	Description
Personal Information	Complete the personal information fields required. Keep in mind the following: • Ensure that the email address is entered correctly; as a notification will be send to the address for the developer to activate the approved account.
Tab	**Description**
Personal Information	The username must be unique The disclaimer checkbox must be selected All Fields with an asterisk (*) must be completed in order to access the other tabs. Alternatively, you may proceed to the step below.
Additional Info	This tab records information about the developer's organization as well as any custom information requested by the Portal for registration. **Organization Name:** Optionally enter the name of the developer's organization. This will appear as the Organization Name in the API Portal interface. If left blank, the username with "_org" will be used as the Organization Name. **Organization Description:** Optionally enter a description about the organization. This will appear as the Organization Description in the API Portal interface. If left blank, the username will be used in the Organization Description.
Get Started on an Application	Completing this tab is optional as are all fields. It records additional information about the developer's application. **Application Name:** Enter the name of the application that will be developed against the API(s). If left blank, no application name is shown. **Platform:** Choose a platform for the application from the drop-down list. **Description:** Enter a description of the application that will be developed against the API(s). If left blank, no description is shown. **Add APIs:** Choose the API(s) to use with the application from the drop-down list. For more information on an API, hover your mouse over the information icon beside the API name. Once you have

	chosen an API, click [**I Accept the Terms and Conditions**]. The selected API displays under Current APIs: • Click the information icon for more information on the selected API. • To remove the API, click the trash can icon.

Click [**Register Now**]. If registrations are subject to approval, the developer will receive an email stating that the account is under review. Otherwise, the developer will be emailed a link to click on in order to activate the account.

Tip: If the email does not appear in the developer's inbox, advise them to check their junk mail folder.

Each new developer has assigned a default Account Plan when they finish the registration process. An **Account Plan** determines the number of queries (hits) that each user can make to any existing applications.

For example, the default one (**Bronze Account Plan**) allows doing 10000 hits per day.

Each user can request an upgrade of Account Plan depending on their needs, increasing the value of the threshold's quota.

3.3 Add New Applications

Developers can add applications of their own through the API Portal.

3.3.1 To add a new application

- Log in to the iCity API Portal.
- From the Dashboard, select **Applications** in the navigation sidebar. The list of your applications displays.
- Click [**Add Application**]. This displays an application wizard with tabs.

- Complete the application information as follows. Click [**Next Step**] when each tab is completed, or click the tab name to move between tabs:

Tab	Description
Application Information	**Name of Application:** Enter a name for the application. **Platform:** Choose a platform from the drop-down list provided. **Description:** Enter a description of the application—for example, details about the platform, whether it's a mobile application, etc. You may use the formatting toolbar to apply basic formatting to the text if desired. Note: The Name of Application and Platform fields must be completed in order to proceed to the next tab.
Additional Info	This tab displays if additional information is required by your organization. Field that display here are customized by administrators.
Tab	**Description**
API Management	This tab lists the APIs currently used by the application. To add a new API: Choose the API from the drop-down list. For more information on an API, hover your mouse over the information icon beside the API name. Read the EULA and signal acceptance by clicking [**I Accept the Terms and Conditions**]. The selected API displays under Current APIs: For more information on the selected API, click the information icon . To remove the API, click the trash can icon . Add additional APIs as necessary.

Auth	If your application is using 0Auth 1.0 or 2.0, complete the following fields as appropriate: **Callback URL:** Supply a call back URL in this field. You can enter multiple URLs separated by commas. **Scope:** Enter the scope or list of access permissions for this client. Scope can be designated in many ways: as a list of resources; URLs or URIs of service endpoints; etc. Scope is a required field for OAuth clients. *Note: By default, the iCity OAuth Toolkit expects Scope to be set to OOB (Out of Band).* **Type:** Select a client type from the drop-down list. Choose **Public** for client-side OAuth clients (such as browser-based JavaScript clients) or **Confidential** for server-side clients. Confidential is also the required Type for the OAuth 2.0 Grant Type of Implicit.

3.4 Manage Applications

Developers can add, edit, enable, disable, or delete their applications via the Manage Applications page.

The Manage Applications page also allows you to view your organization's Account Plan quotas.

Each new Application has assigned a default **API Plan** when the application has been enabled. An API Plan determines the number of queries (hits) that each application can make to their assigned APIs.

For example, the default one (**Sandbox API Plan**) provides an appropriate API service level for developers who are starting to build an application.

Developers can request a different API Plan (for example), when they move their application into test or production.

3.4.1 To enable, disable, or delete your applications

- Log in to the iCity API Portal as a developer or OrgAdmin.
- From the Dashboard, select **Applications** in the navigation sidebar. The list of your applications displays.
- Hover over the appropriate application.
- Select the appropriate action **Delete**, **Disable**, or **Enable** from the drop-down menu of the gear icon.
- Click [**OK**] to confirm.

Note: If an application is Pending Approval, you can view more information about it by selecting View from the drop-down menu of the gear icon.

IMPORTANT: Deleting an application makes all report history for that application irretrievable. If Account Plan Quotas are in effect, API hits made via an application prior to deletion will continue to count toward the Account Plan Quota, even though these hits won't be shown in the report. iCity recommends disabling applications instead of deleting them to maintain report accuracy.

3.5 Edit Applications

Developers can edit their applications via the Manage Applications page.

They can also add, edit, enable, disable, or delete them.

If a new application has been rejected, its status will appear as Rejected in the list of applications and the details of its rejection will appear in an email. When you edit the application and save your changes, the application will be resubmitted and will display a status of Revised.

3.5.1 To edit your applications

- Log in to the iCity API Portal as a developer or OrgAdmin.
- From the Dashboard, select **Applications** in the navigation sidebar. The list of your applications displays.
- Choose a task from the following table:

To Do this	
Change the name or description of an application	Point to the application to edit. Select **Edit** from the drop-down menu of the gear icon. In the [**Application Information**] tab, modify the **Name of Application** or **Description** as appropriate.
Add or delete APIs associated with an application	Hover over the gear icon beside the application to edit. Select **Edit** from the drop-down menu. In the [**API Management**] tab: To remove access to an existing API, click [**Remove API**] next to the API. The application no longer has access to this API. To allow the application to access a new API, choose a new API from the drop-down list. Read and accept the EULA. Click [**Save**] to save the changes. You will receive an email confirmation.
Change API Plan for an API	Hover over the gear icon beside the application to change. Select **Edit** from the drop-down menu. In the [**API Management**] tab, choose a new plan from the **Request Change to** drop-down list. You will receive emails confirming the change request and notifying you whether the request has been accepted or rejected.
Change the Key Secret for an API	Point to the application to change. Select **Edit** from the drop-down menu of the gear icon. In the [**Auth**] tab, click [**Secret**] to access the Key Secret field. Click [**Request a New Shared Secret**].

	Click [**OK**] to confirm. Click [**Save**] when done. Notes: (1) Access to the [**Request a New Shared Secret**] button is limited to Organization Administrators, Business Managers, API Owners, and Account Managers. (2) Issuing a new key secret will immediately invalidate all calls from existing applications using the previous key secret.
Enter a callback URL	If your application will make use of OAuth, enter a callback URL in this field. You may also enter multiple callback URLs separated by commas.

4. APIs Explorer

The API Explorer lets developers interactively discover APIs.

By making choices from among your API's valid resources and methods, and then submitting queries and viewing responses, developers can gain a better understanding of not only how your APIs work, but also the authentication methods required to access them.

4.1 Using an API Explorer

Use the API Explorer to test or change an API resource by sending a request. You can also view the queries sent that generated the response as well as code samples. Any published API with a WADL attached to it is automatically pre-populated into the API Explorer.

Because authentication methods are used to control access to each API resource on the server side, valid credentials are required in order to test the API.

4.1.1 To test applications via the iCity API Explorer

Log in to the API Portal. The API Portal for your organization is displayed.

On the menu bar, click **Documentation** to access the Documentation page. Click iCity API Explorer or 3.11 API Explorer on the navigation sidebar.

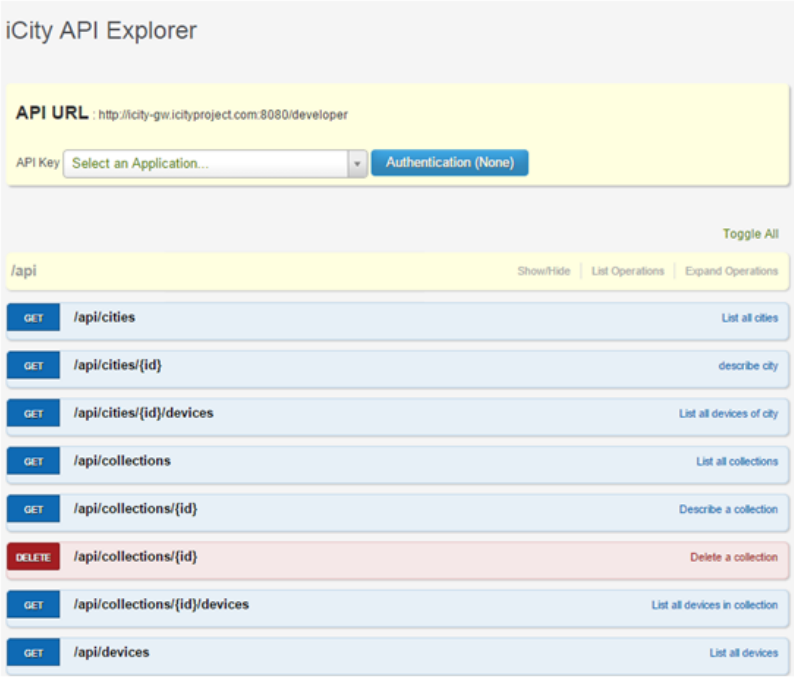

Figure 108 The iCity API Explorer

The API Key drop-down displays:

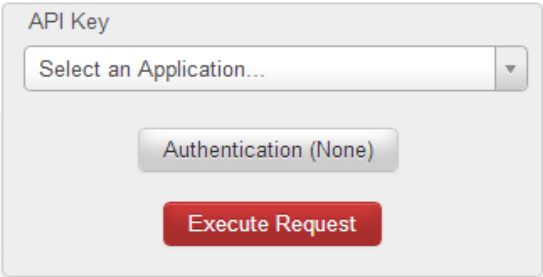

Figure 109 API Key Drop-Down List

To test an application with an API key, choose the application from the **API Key** drop-down list. This pre-populates the API Key value and API Key secret of the chosen application in the **Service Authentication** dialogue box.

Note: If there is no API key for the application, a message displays stating "No API key is available." In order to test an API key, the API key must be generated on the iCity API Portal.

Setting	Description
Resource	Choose the resource for the selected API from the drop-down list.
Method	Choose the method to use for the selected resource. Note: This list may or may not contain an entry as this field is optional. If no methods are displayed, the API Explorer defaults to using the GET method.
[Request] tab	This tab is used to set the resource and method input parameters (if available).
[Add Parameter]	This control is under the [**Request**] tab. It displays the **Add Parameter** dialogue that is used to add additional parameters that are not otherwise specified in the WADL file. Complete the Add Parameter dialogue as follows: **Name:** Enter the name of the parameter to add. **Value:** Enter the value of the parameter to add. **Parameter Type:** Choose a parameter type from the following: **Query:** The input is part of the query parameter. **Header:** The input is part of the request header. Click [**Add**] to validate the input (for existence of value) and add the input to the request.
[Authentication]	Displays the Service Authentication dialogue where you attach an authentication to the selected API. Select API key

Click [**Execute Request**]. The results are displayed in the Response tab.

After viewing the response in the Response tab, you can choose to do the following:

- To view the request sent to the server, click [**Request**].
- To view the query sent to the server, click [**Query**].

The Query tab displays the following:

- Raw request that contains the HTTP request method

- Full request URL, including the query parameters
- Request headers
- Request body (if available)
- Code samples (see below)

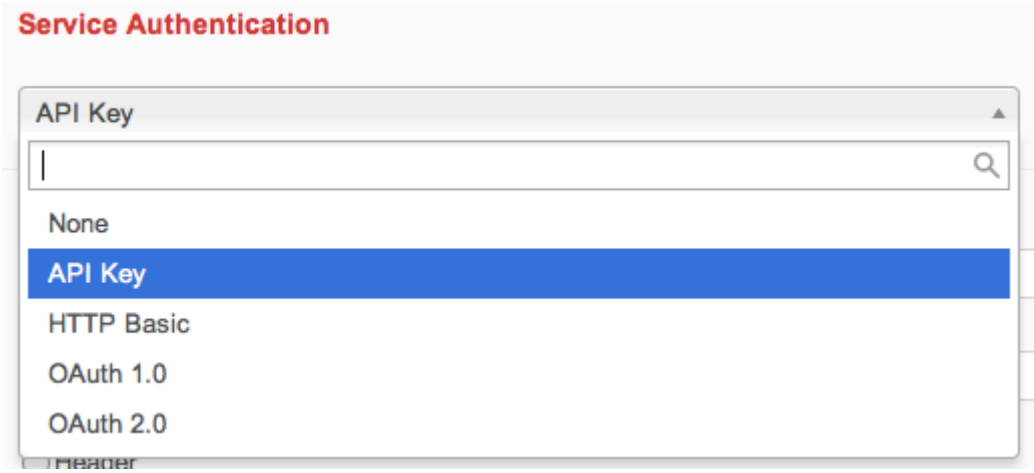

Figure 110 Service Authentication Menu

4.1.2 Example response

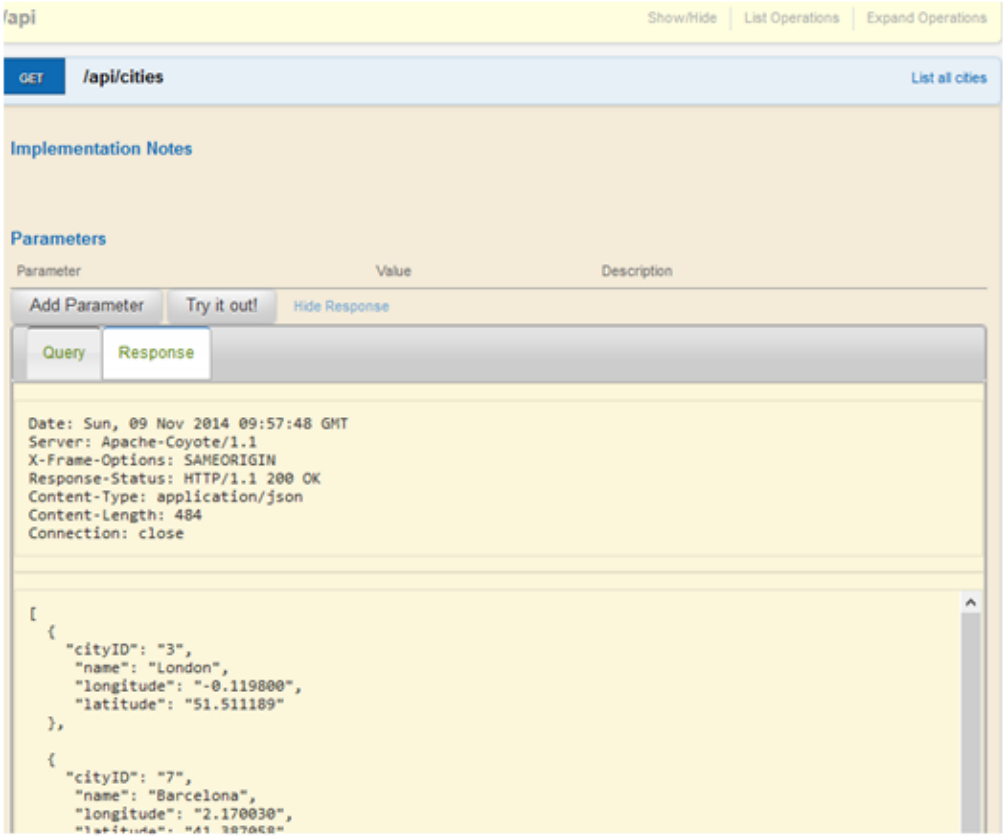

Figure 111 Example Response

4.1.3 Example Query

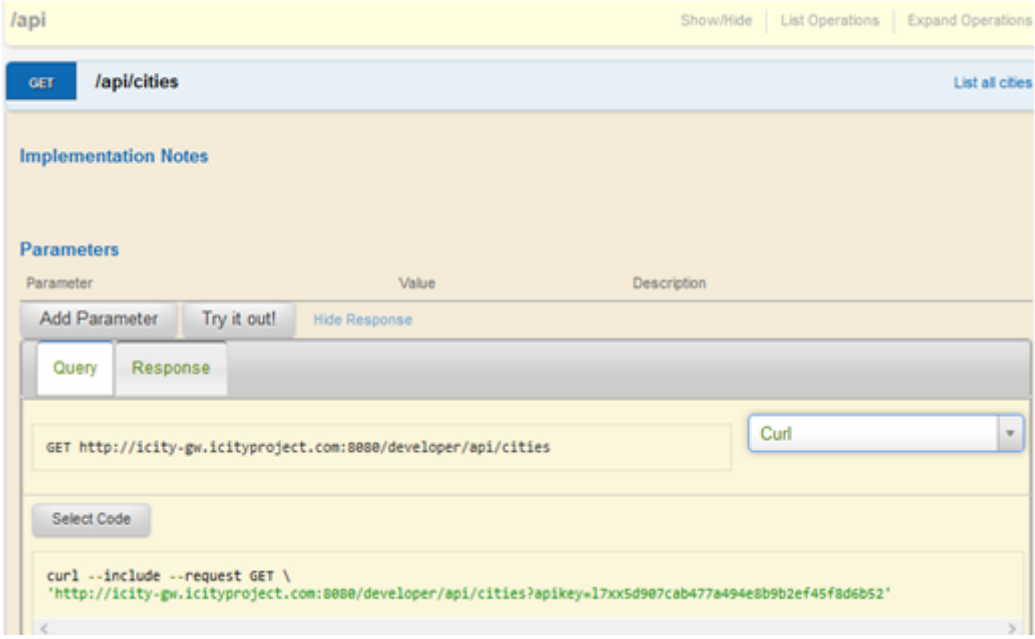

Figure 112 Example Query

4.2 Working with Code Samples

Once you have executed a request, you can view or copy code samples.

4.2.1 To view or copy code samples

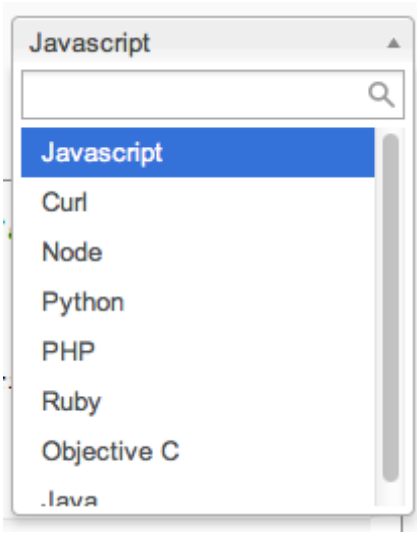

Figure 113 Select the Javascript menu

1. Click the [**Query**] tab.
2. Select a language to display the code sample in from the **Show Code Sample...** drop-down menu.
3. To select code, click [**Select Code**].
4. To copy and paste the selected code, use the standard [**Ctrl**] + **C** and [**Ctrl**] + **V** keyboard commands.

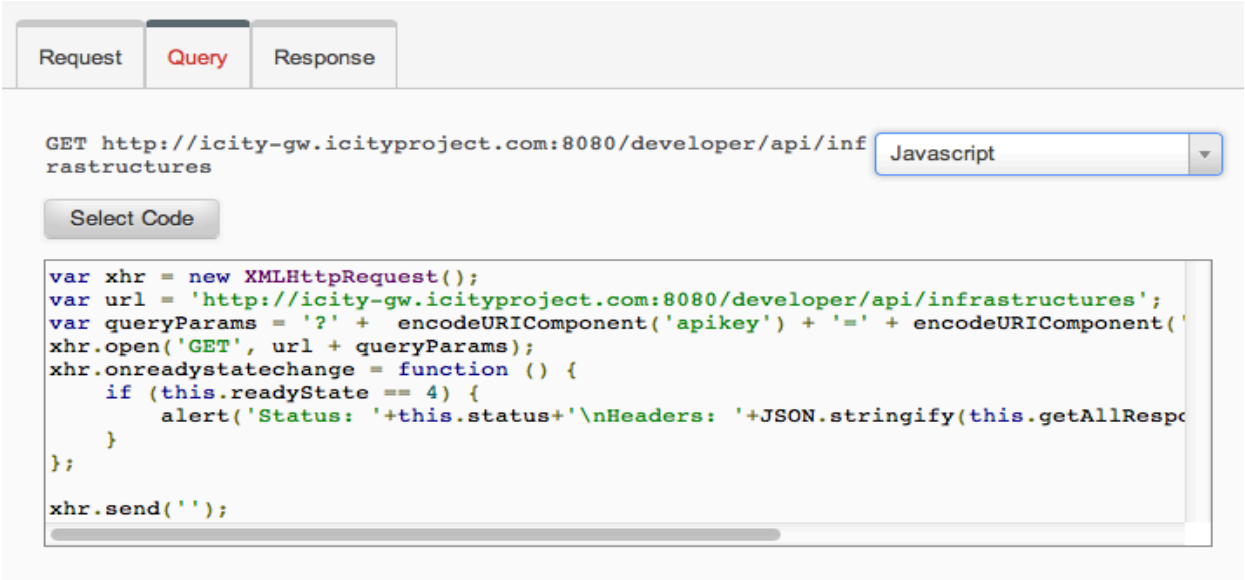

Figure 114 Viewing or copying code samples

4.3 Authenticating an API

In order for a request to execute correctly, an API must be authenticated on the iCity Gateway with an API Key method.

4.3.1 To authenticate an API using an API key

- From the API Explorer page, select the API to authenticate.
- Click [**Authentication**].
- Choose **API Key** from the **Service Authentication** drop-down list.

Note: If you selected an application from the API Key drop-down list, the next 2 steps below are not necessary, because the API Key Name and Value will be pre-populated in these fields.

- Enter the **Name** of the API Key to add. This field is required.

- Enter the **Value** of the API Key to add. This field is required. The API key must be generated on the iCity API Portal.
- Select whether the API Key Type is part of the **Query** parameter, or part of the request **Header**.
- Click [**OK**] to validate your input and add it to the request.

13. APPENDIX IV: Registration process

This document defines the process to follow by a user to become an iCity developer, allowing the access to the platform Information Systems.

1. First of all, the user has to access to the iCity Platform through the main iCity portal or directly to http://icity-devp.icityproject.com. Click the “Sign up” button.

2. First fill the personal information in this tab.

3. The second tab to fill is about additional information.

4. The third and last tab allows the user to start creating an application.

5. After introducing all the information, a message will be sent to the email specified by the user in order to check the address.

6. Once the email has been verified and the platform’s administrators have approved the new user, the access to the platform will be provided.

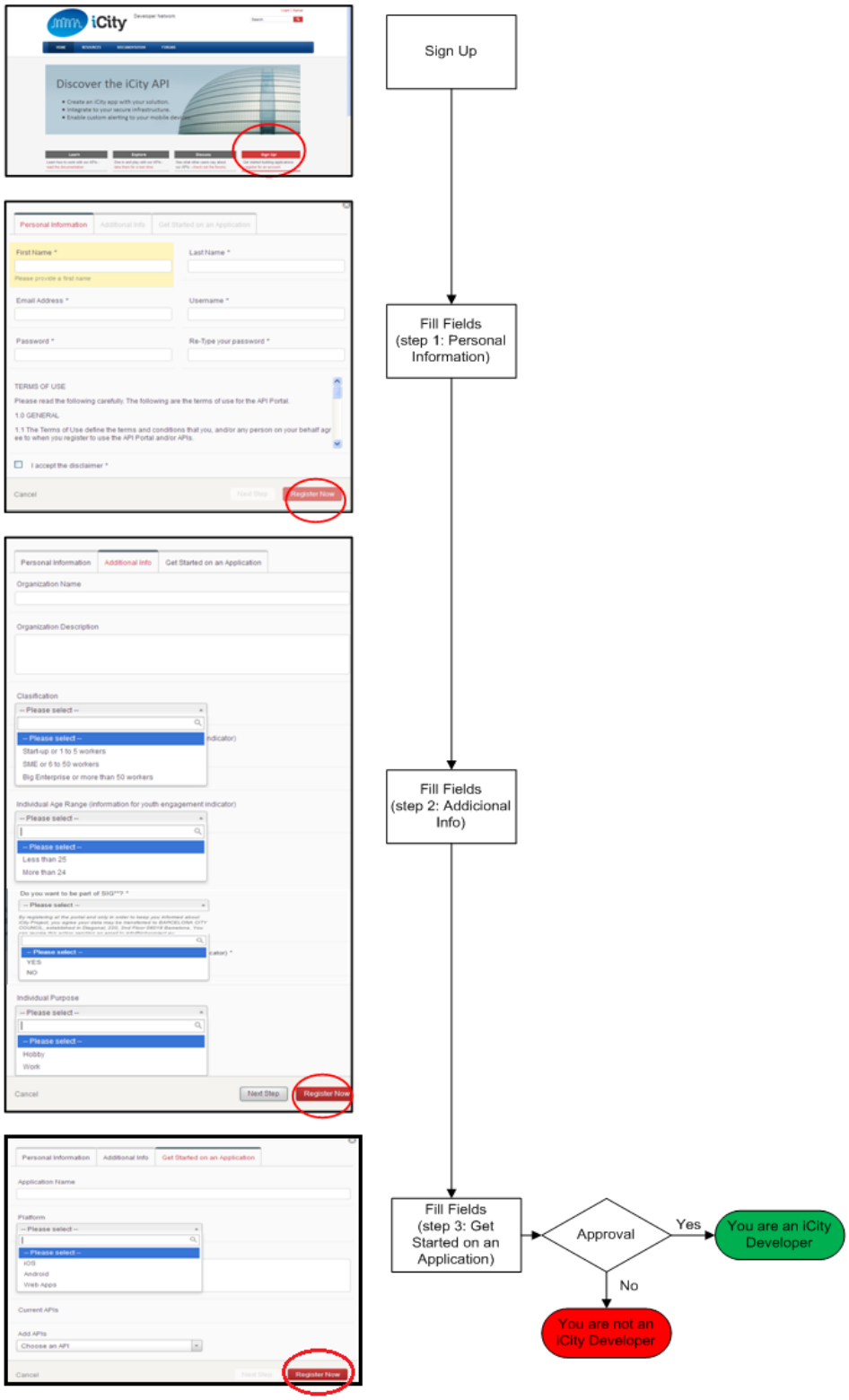

Figure 115 Registration Process

14. APENDIX V: How to use your token

The tokens in the iCity platform represent the administrator's permission to the application for accessing to the platform's data. This token also must be used as an identifier in each request to the platform to obtain any response from it.

Also the tokens can be tested in the "iCity API"/"3.11 API" inside the "Documentation" tab. There all the parameters can be set up and see the response from each request.

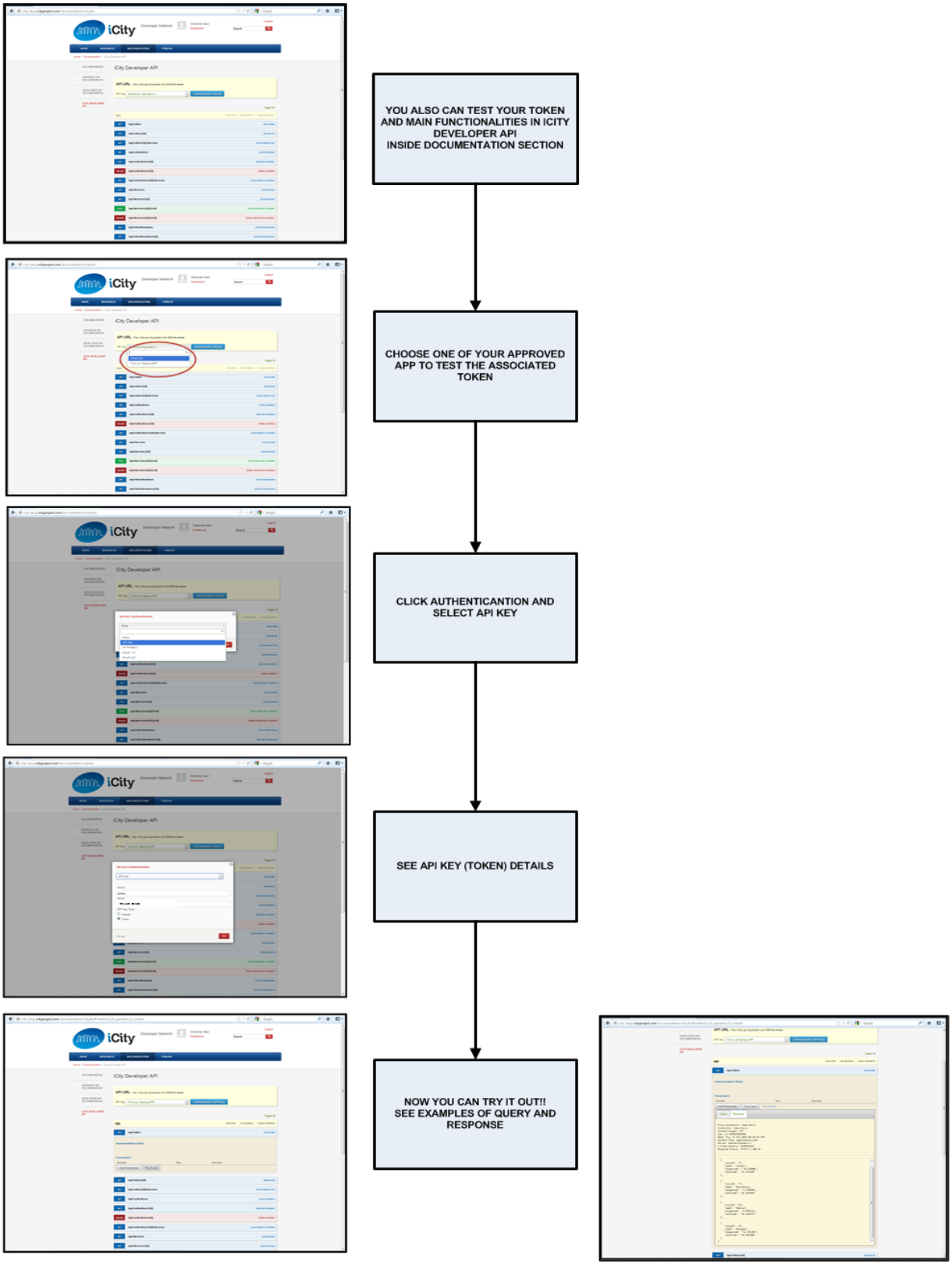

Figure 116 How to use a token

15. APPENDIX VI: App proposal Pre-Validation

This document defines how to create an application and validate it to the testing environment, which is the first part of the complete creation of an application.

1. First a new application must be created by clicking the “Add application” button inside the “Applications” menu.
2. Fill the basic information about the application.
3. Select in the next tab the Information System(s) that the application will use.
4. Select the API that the application will be based on.
5. Fill in the fields in the “Auth” tab.
6. Once all this information has been inserted in the platform, the request will be pending to be approved or not. In both cases the user will receive notifications by email explaining how to proceed.

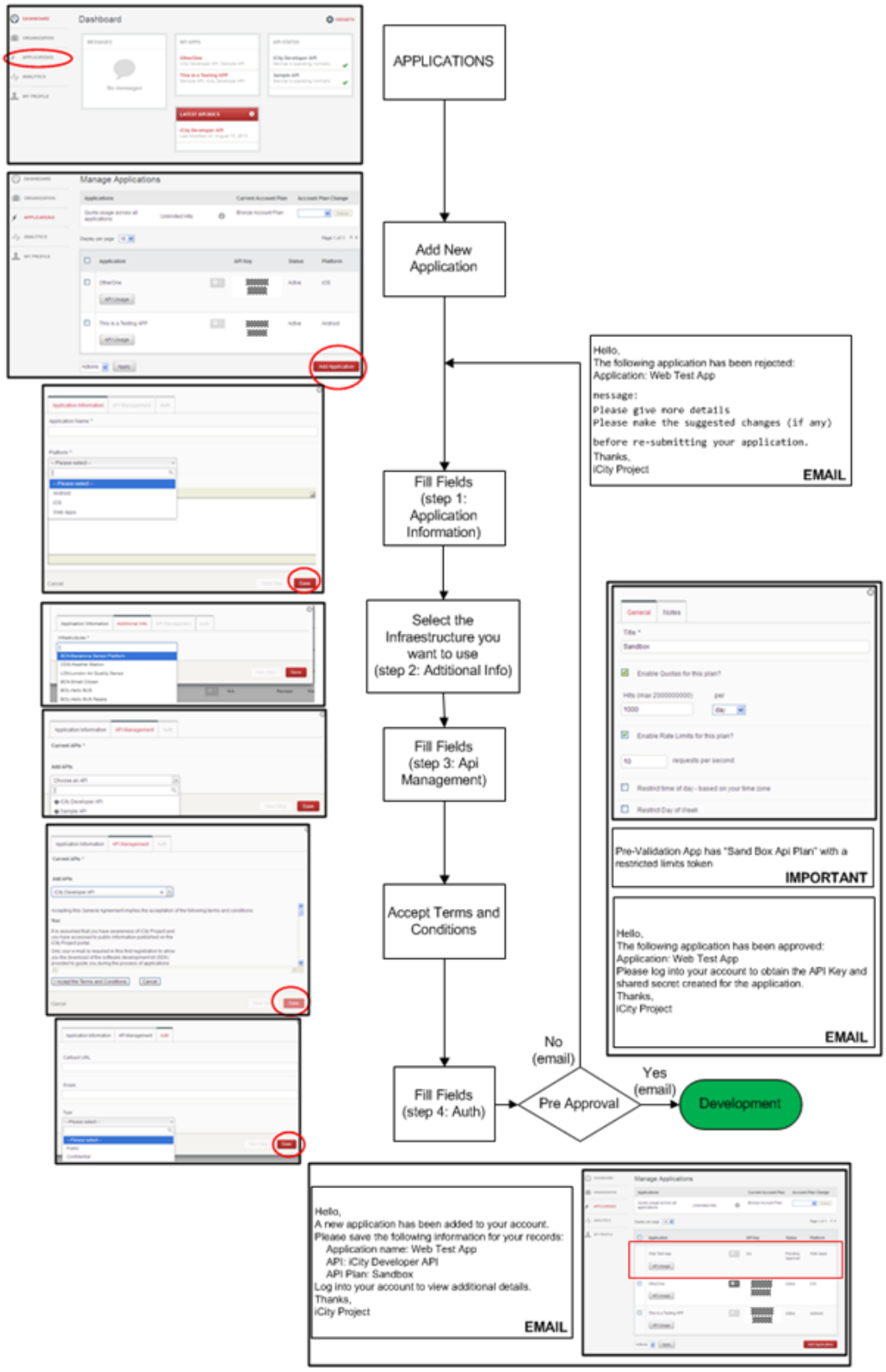

Figure 117 APP Proposal Pre-Validation

16. APPENDIX VII: App proposal Validation

This document defines how to create an application and validate it to the real production environment, which is the final part of the complete creation of an application.

1. Go to the applications list, select one and edit it.
2. Inside “API Management” make a request to the production plan.
3. After this few steps, the request has been made so the platform’s administrators will evaluate if they agree to provide full access to the application.

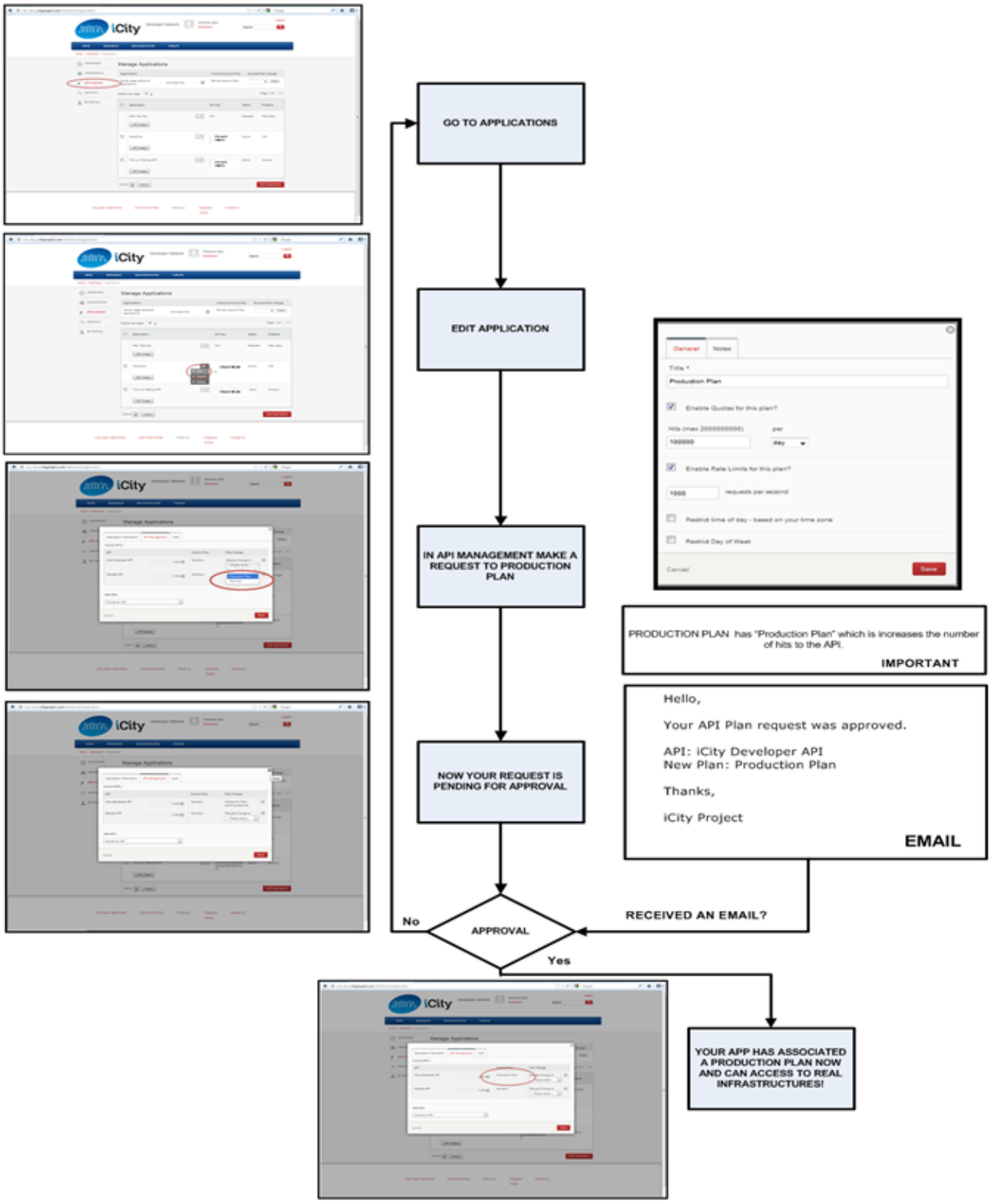

Figure 118 APP Proposal Validation

17. APPENDIX VIII: iCity API model

This document defines the data structure of the iCity API, there are four main entities in it:

- Cities: All the data is separated by the city where it's located.
- Information Systems: This represents all the open Information Systems shared by each city.
- Manufacturers: These set of elements can represent the manufacturers of a whole Information System or a single device.
- Elements: These are the concrete devices located along the city which belongs to a concrete Information System.
    - Properties IN: Set of required parameters to send in each request to the Information System.
    - Properties OUT: Set of parameters returned in the response from an Information System.

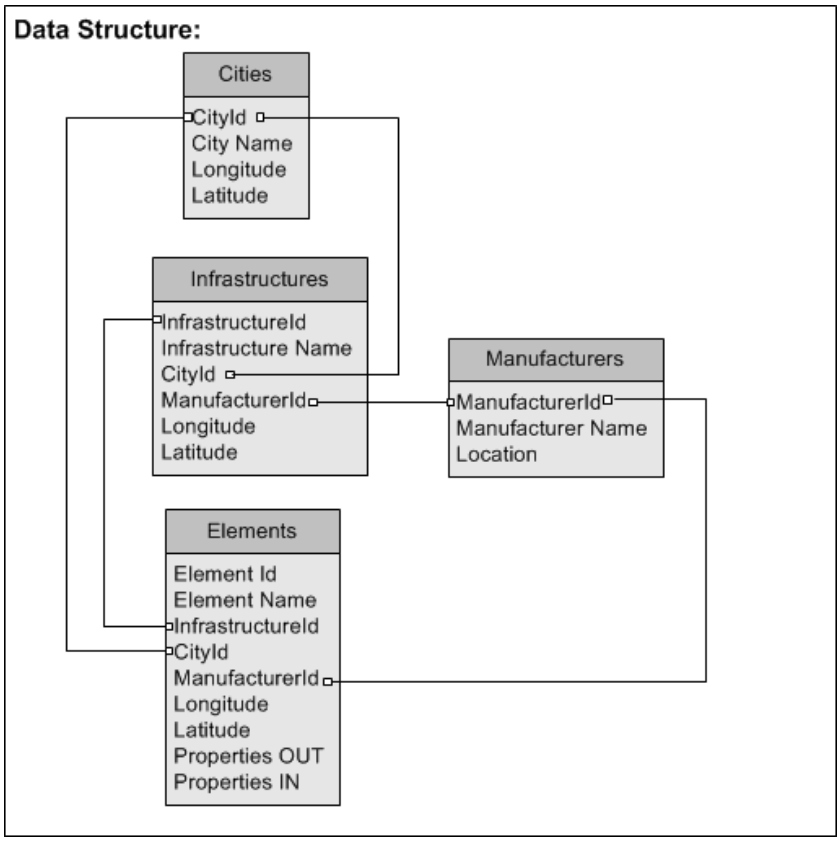

Figure 119 iCity API Model