smar+
en.
ci+y

TOWARDS SMART ZERO CO$_2$ CITIES ACROSS EUROPE

VITORIA-GASTEIZ + TARTU + SØNDERBORG

Deliverable 6.7: Integration and Validation report

WP6, Task 6.7

Date of document

31/01/2018 (M24)

Deliverable Version:	D6.7, V1.0
Dissemination Level:	PU[1]
Author(s):	Jose Luis Izkara and Alberto Armijo (TEC), Felix Larrinaga (MU), Nati Herrasti (ETIC), Mauri Benedito (GIS), Alvaro García (ACC), Andrius Grigas (VG), Jose Luis Hernandez (CAR), Urmas Eero (ET)

[1] PU = Public
PP = Restricted to other programme participants (including the Commission Services)
RE = Restricted to a group specified by the consortium (including the Commission Services)
CO = Confidential, only for members of the consortium (including the Commission Services)

This project has received funding from the European Union' s Horizon 2020 research and innovation programme under grant agreement No 691883

Document History

Project Acronym	SmartEnCity
Project Title	Towards Smart Zero CO2 Cities across Europe
Project Coordinator	Francisco Rodriguez Tecnalia francisco.rodriguez@tecnalia.com
Project Duration	1st February 2016 - 31st July 2021 (66 months)

Deliverable No.	D6.7 Integration and Validation report	
Diss. Level	Public	
Deliverable Lead	*TEC*	
Status		Working
		Verified by other WPs
	X	Final version
Due date of deliverable	31/01/2018	
Actual submission date	31/01/2018	
Work Package	WP 6 - City Information Open Platform (CIOP)	
WP Lead	*MON*	
Contributing beneficiary(ies)	*TEC, MON, ETIC, ACC, CAR, GIS, VG, ET*	

Date	Version	Person/Partner	Comments
26/09/2017	0.1	Jose Luis Izkara and Alberto Armijo (TEC)	First Draft for the ToC
16/10/2017	0.2	Jose Luis Izkara and Alberto Armijo (TEC)	Review of the ToC and completion of the Section 4 (Integration and Validation) and section 5 (Evaluation and Validation in SmartEnCity)
24/10/2017	0.3	Jose Luis Izkara and Alberto Armijo (TEC) Felix Larrinaga (MU) Nati Herrasti (ETIC) Josu Rollón (MTEL) Mauri Benedito (GIS) Alvaro García (ACC) Andrius Grigas (VG)	End-points identification and partial description
01/12/2017	0.4	Jose Luis Izkara and Alberto Armijo (TEC) Felix Larrinaga (MU) Nati Herrasti (ETIC) Josu Rollón (MTEL) Mauri Benedito (GIS) Alvaro García (ACC) Andrius Grigas (VG) Jose Luis Hernandez (CAR) Urmas Eero (ET)	Final Identification and description of End-points Added new section identification in Added-value services testing Added new section identification in Monitoring Testing Description of first deployment of test Identification of Energy Efficiency Test Scenario
04/12/2017	0.5	Jose Luis Izkara and Alberto Armijo (TEC)	Completion of section 1.1 and 1.3
15/12/2017	0.6	Jose Luis Izkara and Alberto Armijo (TEC) Nati Herrasti (ETIC) Mauri Benedito (GIS) Alvaro García (ACC) Andrius Grigas (VG) Urmas Eero (ET)	New section introducing the description of end-points in the framework of the CIOP architecture (Section 6.1) Final deployment of most of the Unit Tests Completion of the identification in Monitoring Testing Detailed description of Integration test 1
12/01/2018	0.7	Jose Luis Izkara and Alberto Armijo (TEC) Nati Herrasti (ETIC) Mauri Benedito (GIS) Alvaro García (ACC)	New section describing the global access system. Completion of the deployment of Integration test 1 Completion of the deployment of Integration test 2

		Jose Luis Hernandez (CAR) Andrius Grigas (VG) Urmas Eero (ET)	Completion of the deployment of Integration test 3 Complete Reports on test (Section 7) Section of added-value services testing has been completed Final conclusions Document consolidation and format
29/01/2018	1.0	Jose Luis Izkara and Alberto Armijo (TEC) Andrius Grigas (VG) Mauri Benedito (GIS) Tõnis Eelma (ISE)	Detailed Integration, qualitative assessment and corrective measures for Integration test 3. Final updates according to reviewers' comments.

Copyright notice

Table of content:

Table of Tables:

Table of Figures:

Abbreviations and Acronyms

Abbreviation/Acronym	Description
API	Application Programming Interface
BVT	Build Verification Test
CIOP	City Information Open Platform
CityGML	City Geography Mark-up Language
CRUD	Create, Read, Update and Delete
ESCO	Energy Savings Company
EC	European Commission
ETL	Extract, Transform and Load
EU	European Union
EV	Electric Vehicle
GIS	Geographic Information Systems
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
HMI	Human Machine Interface
ICT	Information and Communication Technologies
IoT	Internet of Things
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MySQL	My Structured Query Language
NoSQL	No Structured Query Language
OGC	Open Geospatial Consortium
OS	Operating System
OSM	OpenStreetMap
QA	Quality Assurance
RA	Reference Architecture
REST	Representational State Transfer
RTM	Requirement Traceability Matrix
SCADA	Supervisory Control And Data Acquisition
SDLC	Software Development Life Cycle
SmartEnCity	Towards Smart Zero CO2 Cities across Europe
STLC	Software Testing Life Cycle
SQL	Structured Query Language
WCS	Web Coverage Service
WP	Work Package
WMS	Web Map Service

Table 1: Abbreviations and Acronyms

0 Publishable Summary

SmartEnCity focuses on the development of a highly adaptable and replicable systemic approach towards urban transformation into sustainable, smart and resource efficient urban environments in Europe, through the planning and implementation of measures aimed at improving energy efficiency in the main consuming sectors in cities and increasing the supply of renewable energy. This approach will be defined in detail, and subsequently laid out and implemented in the three Lighthouse demonstrators (Vitoria-Gasteiz in Spain, Tartu in Estonia and Sonderborg in Denmark), to be further refined and replicated with the development of Integrated Urban Plans (IUPs) in all participant (both Lighthouse and Follower) Cities.

WP6 aims to devise a common ICT platform that will be the reference for the deployment of the "City Information Open Platform" (CIOP) in each one of the pilot lighthouse projects. The platform will provide a standardized data model to accommodate data from each pilot and will also define standardized services and modules for data consumers, especially relevant are those related to the monitoring of SmartEnCity KPIs, those requested by the EC in the call and those identified as ICT solutions for the project.

Deliverable D6.7 presents the results of Task 6.7 "Integration and Validation" within WP6 of the SmartEnCity project. The main objective for this task is the integration and validation of the different modules of the ICT platform. The task included the following activities: (1) selection of integrity and validation tests; (2) deployment of tests; (3) correction measures and; (4) report on test.

This document presents first a generic methodological approach applicable for the integration and validation of smart city projects. As the CIOP is software asset, the approach takes the basis from the testing and quality assurance of the software development. The selected approach is based on the V model of testing. Several standards, tools and templates have been defined and are available for the implementation of the selected approach to SmartEnCity project. Next, this testing methodology is framed to the specific needs of SmartEnCity to validate the demonstrators to be developed in the project. A check plan for the validation of the different modules of the ICT platform developed in the previous tasks is deeply described, including performance and functional tests, as well as unit tests and integration test. Specific templates are also provided for the deployment of each kind of test. The test plan for the added value services envisaged to be developed for the lighthouses implementations is also detailed, as well as the principles and the main aspects to consider during the operation phase for the monitoring testing. The global user authentication system is described as a tool for integration and evaluation in the SmartEnCity platform. Based on the reference architecture proposed for SmartEnCity, a set of end-points developed in previous tasks of the WP6 have been described in detail and later deployed for testing. The list of end-points tested is representative of each of the CIOP layers defined in the reference architecture and the different types of end-points developed. In the testing deployment, first the end-points have been tested individually following the templates and guidelines described in the methodological approach. Then three different integration scenarios have been defined, including some of the end-points developed. Selected integration scenarios are complementary in terms of domain and lighthouse implementation. The integration tests have been also developed following the defined approach. This report

finally presents the results of the tests, corrective measures and conclusions identified after the work done for the completion of the Task 6.7.

The integration and validation report represents the formal representation of the results of tests for modules of the SmartEnCity CIOP developed in WP6, as well as the specification for integration and validation of the platform and added value services to be then implemented tailored to the specific constraints and expectations of each lighthouse city (WP3, WP4 and WP5).

1 Introduction

1.1 Purpose and target group

This public report constitutes deliverable D6.7 Integration and Validation report due on M24 of the SmartEnCity project. The main objective of task T6.7 Integration and Validation, whose results are reported in this deliverable, is to test and assess the different modules of the ICT platform, following the Reference Architecture principles ideated and developed in WP6 City Information Open Platform (CIOP). The main activities carried out in this task are listed here:

- An integration and validation methodology was provided as a generic software testing framework that covers the best practices in Software Testing Life Cycle (STLC). Some test standards and testing tools widely accepted by developers and testers were introduced to support the STLC methodology. In order to allow the testing results to follow a common approach, some testing templates were introduced
- Based on the STLC methodology, testing tools and test templates, the integration and validation plan for SmartEnCity project modules was presented as a particularization of the STLC methodology
- Next, the end-points to be tested were identified by the developers of the CIOP specific modules and the tests were deployed and reported following the testing templates, which were used to document unit tests and integration tests
- Furthermore, the correction measures derived from the test results were applied and documented accordingly

This report is structured into the following sections:

This section 1 presents the purpose of the document, the main contribution of each partner and the relationship of the current document with other WP and deliverables.

Section 2 presents the objectives of WP6 and the objectives of the task T6.7 in relation to the work package.

Section 3 identifies the integration and validation methodology as a generic software testing framework that covers the best practices in Software Testing Life Cycle (STLC).

Section 4 specifies the diverse types of tests that were identified following the STLC methodology. These tests were performed during the test deployment in section 7.

Section 5 identifies the specific end-point elements to be tested in the City Information Open Platform (CIOP) according to the structure of the Reference Architecture.

Section 6 presents the deployment of tests, according to the tests presented in section 4.

Section 7 presents a summary of the report on tests in a structured way and outlines a summary of correction measures derived from the report on tests

Section 8 presents the conclusions, deviations found and the inputs for other work packages.

Main target group of the information, the test reports and the conclusions collected in this deliverable are the partners in charge of the development of the CIOP platform at use case level. That is at city level in Work Packages 3, 4 and 5. Follower cities could also take advantages of the findings and results produced in this task.

1.2 Contributions of partners

The following Table 2 depicts the main contributions from participant partners in the development of this deliverable.

Participant short name	Contributions
TEC	Task Leader. Responsible of the content of the deliverable. Main contributor of Sections 1, 2, 3, 7 and 8, and contributor of Section 4 (Performance tests, Functional tests, Unit testing and Integration testing), Section 5 (Introduction and GIS structural repo services) and Section 6 (GIS structural repo services and Integration test 1).
ET	Contributor of Section 4 (Monitoring testing) and Section 6 (Integration test 2)
MON	Contributor of Section 5 (Data acquisition about Energy, ETL Processes- Energy Vertical Repo, ETL Processes – Historical Repo, ETL Processes – Structural Repo) and Section 6 (Data acquisition about Energy, ETL Processes- Energy Vertical Repo, ETL Processes – Historical Repo, ETL Processes – Structural Repo and Integration test 1)
ETIC	Contributor of Section 5 (API for Energy Services, ETL Processes – Mobility Repo, API for Mobility Services, API for Citizen Engagement Services) and Section 6 (API for Energy Services, ETL Processes – Mobility Repo, API for Mobility Services, API for Citizen Engagement Services and Integration test 1)
CAR	Contributor of Section 4 (Added-value services testing).
ACC	Contributor of Section 4 (Global Access System).
GIS	Contributor of Section 5 (Energy Application and GIS repo services) and Section 6 (Energy Application, GIS repo services and Integration test 1)
VG	Contributor of Section 5 (API for KPIs and Services for integrating open data) and Section 6 (API for KPIs, Services for integrating open data and Integration test 3)

Table 2 Contribution of partners

1.3 Relation to other activities in the project

The following Table 3 depicts the main relationship of this deliverable to other activities (or deliverables) developed within the SmartEnCity project and that should be considered along with this document for further understanding of its contents.

Deliverable Number	Contributions
D6.2	This demonstrator presents the Reference Architecture for SmartEnCity
D6.3	This demonstrator presents the Reference Architecture for SmartEnCity completed with the Data Models
D6.4	This demonstrator presents the Reference Architecture for SmartEnCity completed with the Interoperability Mechanisms

WP3, WP4 and WP5	The implementation in each lighthouse will agree to the Reference Architecture and the layers and modules defined in it. The integration and validation of each implementation will be performed there.

Table 3 Relation to other activities in the project

2 Objectives

2.1 Objectives of WP

As stated in the Grant Agreement, the overall objective in this work package is to devise a common ICT platform that will be the reference for the deployment of the “City Information Open Platform” in each one of the pilot lighthouse projects. The detailed objectives of the work package are:

- Define the specifications of the platform. Functional and non-functional requirements must be identified considering the overall expected performance of the platform. (Done in SmartEnCityD6.1, 2016).
- Define and provide the infrastructure or technological architecture that will enable gathering information from the different verticals (building retrofitting, district heating, smart grid, smart mobility) and offer data to the consumer applications (web applications, reports, control algorithms etc.) This is the main objective of this task/deliverable (Done in SmartEnCityD6.2, 2017).
- Provide a data model that will accommodate data from different sources such as electric vehicle charging points, appliances and lighting systems in dwellings, district heating Supervisory Control And Data Acquisition (SCADA) systems, data collected by utilities with smart meters, data from building elements (lifts, lighting systems…). (Done in SmartEnCityD6.3, 2017)
- Provide the mechanisms and protocols to ease interconnection between platform modules and to allow data uploading/consuming from the different sources, enhancing interoperability between the platform and other systems. (Done in SmartEnCityD6.4, 2017)
- Provide the mechanisms to build ICT solutions for different stakeholders offering actionable information and recommendations, to empower citizens on decision making in relation to home energy consumption and mobility and to encourage them to reduce their environmental and resources footprint. (Done in SmartEnCityD6.5, 2017)
- Provide mechanisms to build added value service linking the platform to social networks with the objective to boost engagement of stakeholders with the ICT platform and more importantly raise awareness about energy consumption. Also provide mechanisms to build added value services offering data analysis of monitored data, through machine learning big data techniques or business intelligence techniques. (Done in SmartEnCityD6.6, 2017)
- Integrate and validate the different modules of the ICT platform. (This document Deliverable 6.7)

2.2 Objectives of Task 6.7

The main objective for this task (Task 6.7) and its deliverable (D6.7) is to test and assess the different modules of the ICT platform of SmartEnCity. Specific objectives of the Task 6.7 are:

- Identification of a generic integration and validation methodology applicable for smart city projects to be framed to the specific needs of SmartEnCity.

- Selection and definition of integrity and validation tests, including the definition of the test and the testing templates to be completed during the tests
- Deployment of tests of a set of representative modules of the SmartEnCity CIOP platform developed in WP6
- Define and implement at least one end-to-end test to show the integration of the developed modules in a complete scenario.
- Set the basis for integration and validation of the platform and added value services to be then implemented for each lighthouse city of SmartEnCity.

3 Integration and Validation

This chapter presents a generic integration and validation methodology applicable for smart cities projects developed in Task 6.7. Next, this testing methodology will be framed to the specific needs of SmartEnCity to validate the demonstrators developed in the project.

3.1 Introduction

Software testing is a quality assurance activity intended to check whether the actual test results are in line with the expected results and to ensure that the software system is having no sensible deviation from the original business requirements. Software testing also helps the end-users and software developers to identify errors, gaps or missing functional and non-functional requirements that support the business requirements. It can be either done **manually** or using **automated tools**.

In manual testing, developers and end-user testers manually execute test cases without using any automation tools. Any new application or tool must be manually tested before its testing can be automated. Manual testing requires more effort than automated testing, but is required to check automation feasibility. Moreover, manual testing does not require knowledge of any testing tool. On the other hand, automated testing implies using an automation tool to execute the test cases. The automation software tools can also enter test data into the system under test, compare expected and actual results and generate detailed test reports.

There exist two types of software testing techniques, namely, **white box** and **black box testing**. White-box testing or structural testing is a method of testing software that tests internal structures or workings of an application, as opposed to black-box testing, which examines the functionality of an application.

The testing techniques used to assess that an application behaves and looks as expected embrace all the components of the vertical slice of a software suite, i.e. from the frontend to the backend. Some of these testing techniques are:

- **Functional testing**: Functional testing is a type of testing which verifies that each function of the software application operates in conformance with the functional requirement specification. This testing mainly involves black box testing and it is not concerned about the source code of the application. Each and every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results. This testing involves checking of User Interface, APIs, Database, security, client/server applications and functionality of the Application Under Test. The testing can be done either manually or using automation tools
- **Non-functional testing**: Similar to functional requirements, there are non-functional requirements like performance, usability, load factor that are also equally important to support the correct functioning of an application. Thus, non-functional testing is a type of testing which verifies that the software application operates in conformance with the non-functional requirement specification
- **Security testing**: Security enforcement is a set of measures to protect an application against unforeseen actions that cause it to stop functioning or being exploited. Unforeseen actions can be either intentional or unintentional. Security testing is a

variant of non-functional testing, which ensures that system and applications in an organization are free from authorization and authentication weaknesses that may cause a big loss

- **Load testing**: Load testing helps determine how the application behaves when multiple users access it simultaneously (concurrency)
- **Time response testing**: Time response testing is a type of testing to ensure software applications will respond quickly under their expected workload
- **Scalability testing**: Scalability Testing is the ability of a network, system or a process to continue to function well, when changes are done in size or volume of the system to meet growing need
- **API testing**: API is an acronym for Application Programming Interface. It enables communication and data exchange between two separate software systems. A software system implementing an API contains functions/sub-routines which can be executed by another software system
- **GUI testing**: GUI testing is the process of testing the system's Graphical User Interface of the Application Under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars - toolbar, menu bar, dialog boxes and windows, etc.
- **Smoke testing**: Smoke testing, also known as Build Verification Test (BVT), is preliminary test performed to reveal simple failures severe enough to, for example, reject a prospective software release. A smoke tester will select and run a subset of test cases that cover the most important functionality of a component or system, to ascertain if crucial functions of the software work correctly
- **Regression testing**: Regression testing is a type of software testing which verifies that software which was previously developed and tested still performs the same way after it was changed or interfaced with other software. Changes may include software enhancements, patches, configuration changes, bug fixes, etc. The purpose of regression testing is to ensure that those applied changes have not introduced new faults

Depending on the development phase of an application, the types of testing are:

- **Unit testing**: Unit testing of software applications is done during the development (coding) of an application. The objective of unit testing is to isolate a section of code and verify the correctness of its individual parts. In procedural programming, a unit may be an individual function or procedure. Unit testing is usually performed by the developer
- **Integration testing**: In Integration Testing, individual software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. Integration testing focuses on checking data communication amongst these modules. It is also termed Integration and Testing
- **System testing**: System testing is the testing of a complete and fully integrated software product. Usually software is only one element of a larger computer based system. Ultimately, software is interfaced with other software/hardware systems. System testing is actually a series of different tests whose sole purpose is to exercise the full computer based system

3.2 Integration and Validation Methodology

In order to build robust software, developers follow the Software Development Life Cycle (SDLC) methodology. In parallel to software development tasks, developers and testers also follow the Software Testing Life Cycle (STLC), which is the sequence of activities carried out by the testing team from the beginning of the project till the end of the project. Thus, the V model of testing was developed, as an extension to the Waterfall development cycle, where for every phase in the development life cycle there is a corresponding testing phase, as shown in Figure 1. The left side of the model belongs to the Software Development Life Cycle – SDLC, while the right side of the model is Software Test Life Cycle – STLC. The entire figure looks like a V, hence the name of the testing model. As opposed to the V model, the testing phase in a Waterfall model starts after the implementation has been done.

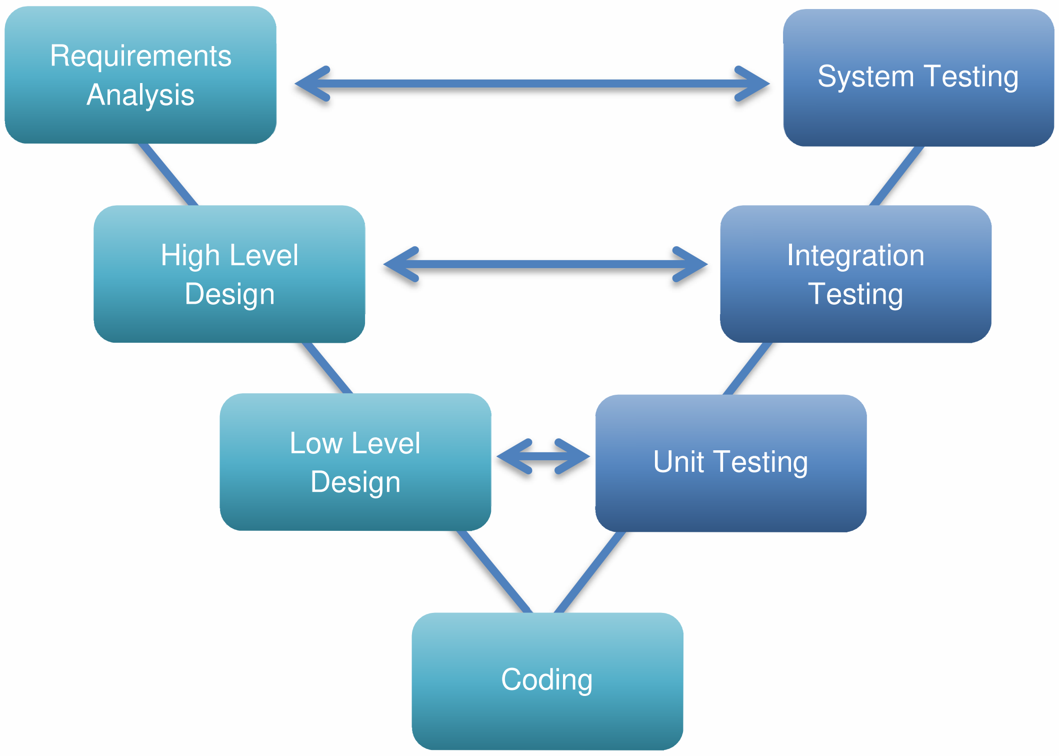


Figure 1 V model of testing

The Software Testing Life Cycle (see

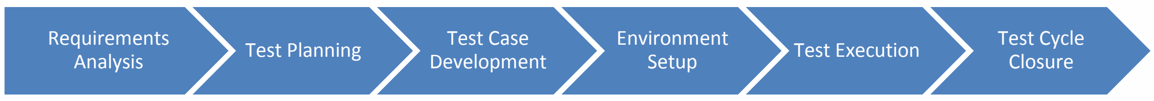


Figure 2) process is executed in a sequence in order to meet the quality goals. Thus, it is not a single isolated activity but it consists of many different tasks that are executed to achieve quality assurance in a software product. There are distinct phases in STLC which are given below:

1. Requirement Analysis
2. Test Planning
3. Test Case Development
4. Environment Setup

5. Test Execution
6. Test Cycle Closure

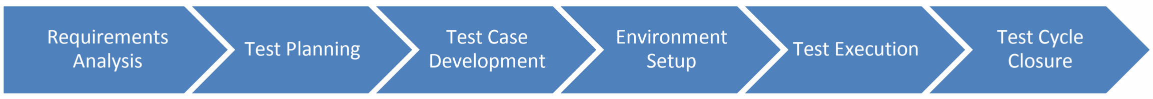


Figure 2 Software Testing Life Cycle

Each of the steps shown in

Figure 2 has some Entry Criteria (known as the minimum set of conditions that should be met before starting the software testing) as well as Exit Criteria (a minimum set of conditions that should be completed in order to continue the software testing) on the basis of which it can be decided whether we can move to the next phase of Testing Life cycle or not. The process phases are described below:

3.2.1 Requirement Analysis

The requirements analysis is the first step in Software Testing Life cycle (STLC), where the testing team reviews the requirements document with both functional and non-functional details to identify the testable requirements.

In case of any technical issues, the Quality Assurance (QA) team may setup a meeting with the clients and the stakeholders (Technical Leads, Business Analyst, System Architects and Client) to clarify their doubts. Once the QA team is clear with respect to the requirements, they will document the acceptance criteria and get it approved by the customers.

The **activities** to be done during the requirements analysis phase are given below:

- Analyzing the system requirement specifications from the testing point of view
- Preparation of Requirement Traceability Matrix (RTM), which maps the technical requirements with the test cases
- Identifying the testing techniques and testing types
- Prioritizing the tests cases execution
- Analyzing the automation feasibility
- Identifying the details about the testing environment where actual testing will be done

The **deliverables** (Outcome) derived from the requirement analysis phase are:

- Requirement Traceability Matrix (RTM)
- Automation feasibility report

3.2.2 Test Planning

The test planning step initiates after the completion of the requirement analysis phase. In this phase, the QA manager prepares the test plan and test strategy documents. Along with these documents, the QA manager will also come up with the testing effort estimations.

The **activities** to be done in the test planning phase are given below:

- Selection of testing approach
- Preparation of test plan and test strategy documents
- Resource planning and assigning roles and responsibility to them
- Selection of testing tools

The **deliverables** (Outcome) of the test planning phase are:

- Test plan document
- Test strategy document
- Best suited testing approach
- Number of resources, skills required and their roles and responsibilities
- Testing tools to be used

3.2.3 Test Case Development

During this stage, the QA team oversees the writing of the test cases according to the test plan. The team also writes scripts for automation if required. Verification of both the test cases and test scripts are done by peers. Creation of test data is done in this phase.

The **Activities** to be done in Test Case Development phase are summarized below:

- Creation of test cases
- Creation of test scripts if required
- Verification of test cases and automation scripts
- Creation of test data in testing environment

The **deliverables** (Outcome) of test case development phase are:

- Test cases
- Test scripts (for automation if required)
- Test data

3.2.4 Test Environment setup

The testing environment setup phase comprises the setup or installation process of the software and hardware required for testing the application. In this step, the integration of third-party applications is also carried out if required in the project. After setting up the required software and hardware, the installation of the build is tested. Once the installation is completed, the test data is generated and smoke testing is executed on the build to check whether the basic functionalities are working as expected. This phase can be done in parallel with the Test Case Development phase.

Summarizing, the **activities** to be done in the test environment setup phase are given below:

- As per the requirement and architecture document, the list of required software and hardware is prepared
- Setting up of test environment
- Creation of test data
- Installation of build and execution of smoke testing on it

The **deliverables** (Outcome) of the test environment setup phase are:

- Test environment setup is ready
- Test data is created
- Results of smoke testing

3.2.5 Test Execution

The test environment setup should be ready before the test execution is started, where the test cases are executed in the testing environment. While execution of the test cases, the QA team may find bugs, which will be reported against that test case. This bug is fixed by the developer and is retested against the test case by the QA.

The **activities** to be done in the test execution phase are given below:

- Execution of test cases
- Reporting test results
- Logging defects for the failed test cases
- Verification and retesting of the defect
- Closure of defects

The **deliverables** (Outcome) of the test execution phase are:

- Test execution report
- Updated test cases with results
- Bug report

3.2.6 Test Cycle Closure

The Test Execution phase should be finalized before starting the test cycle closure activity. In the test cycle closure phase, the QA team will meet and discuss about the found testing artifacts. The whole purpose of this discussion is to learn lessons from the previous steps.

Summarizing, the **activities** to be done in the test cycle closure phase are given below:

- To evaluate the test completion based on test coverage and software quality
- Documentation of the learning (retrospective) from the project
- Analyzing the test results to find out the distribution of severe defects
- Test closure report preparation

The **deliverable** (Outcome) of the test cycle closure phase is the final report on test closure.

Table 4 briefly explains the Software Testing Life Cycle along with the entry criteria, activity, exit criteria and deliverables associated with each of the previous phases.

STLC phases	Entry Criteria	Activity	Exit Criteria	Deliverables (Outcome)
Requirement analysis	Availability of requirement document both **functional as well as non-functional** Architectural document of the application or the product should be available Acceptance criteria defined and duly signed by the customers	Analysis of system requirement specifications to understand the different business modules and its functionalities To identify the user profile, user interface and user authentication **Types of tests to be performed** on the application or product should be identified Should collect the details about testing priorities Preparation of RTM i.e. Requirement Traceability Matrix Test Environment details should be identified in order to do testing Analysis of automation possibility if it is required	RTM should be signed off The customer should sign off on the test automation feasibility	Requirement Traceability Matrix (RTM) Report on Automation Feasibility if it is applicable
Test Planning	Detailed requirement document Requirement Traceability Matrix (RTM) Automation Feasibility Report	Preparation of Test Plan document Preparation of Test Strategy document To analyze the best suited testing approach for the application or product To analyze the testing techniques and the types of testing to be carried out in	Approved Test Plan document Approved Test Strategy document Document of Effort estimation	Test Plan document Test Strategy document Effort estimation document

		order to maintain the quality Selection of the testing tools Estimation on the testing efforts Resource planning as per the skills required for testing and also assigning roles and responsibility to them		
Test case development	Detailed Requirement document Test Plan and Test strategy documents Automation Feasibility Report	Creation of **test cases** for all the modules or features in the application or product Creation of automation scripts if required Review of test cases and test automation scripts Test data creation	Reviewed Test cases Reviewed Test automation scripts Test data creation ready for testing	Test cases Test automation scripts Test data
Test Environment setup	System design documents should be available Architectural document of the application should be available Environment set-up plan document should be available	Understanding the design and architecture of the application Setting up the **test environment** Installation of required hardware and software in order to start testing the application Integration of any third-party application (if required) Installation of build Creation of **test data**	Environment setup is ready for testing All the required software and hardware are installed Build installation is complete and successful Test data creation is complete Smoke testing is done	Test environment along with test data Smoke test result

		Execution of smoke testing on the build Accepting or rejecting the build as per the smoke test result		
Test Execution	Documents like RTM, Test Plan, Test strategy, Test cases and Test scripts should be ready Test environment should be ready Test data should be ready Integration of third party application (if required) should be successful Smoke testing of the application should be successful	**Execution of test cases** Preparation of test result document **Logging defects** for the failed test cases Mapping of defects with the test cases To update the test cases and test strategy if required Fixed defects should be retested Closure of the defects if they are working as expected Execution of regression testing of the application or product in order to ensure its stability post defect closure	All test cases are executed Defects are logged and tracked for closure	Completed the test case execution Updated the test cases wherever required Defects reported
Test cycle closure	All the test cases are executed and updated Test results are documented Defect logs are available	Evaluation of the test completion on the basis of Test Coverage and Software Quality Preparation of **Test Closure report** Analyzing the test results to find out the distribution of severe defects	Signed off test closure report by the client	Test closure report

Table 4 Software Testing Life Cycle

3.3 Test Standardization and Tools

3.3.1 Test Standardization

Several standardization organizations have developed and implemented different standards to improve the quality of the released software. Some of the widely used standards related to quality assurance are listed below:

ISO/IEC 9126: The fundamental objective of the ISO/IEC 9126 standard is to address some of the well-known human biases that can adversely affect the delivery and perception of a software development project. These biases include changing priorities after the start of a project or not having any clear definitions of "success". By clarifying, then agreeing on the project priorities and subsequently converting abstract priorities (compliance) to measurable values (output data can be validated against schema X with zero intervention), ISO/IEC 9126 tries to develop a common understanding of the project's objectives and goals.

ISO/IEC 9241-11: Part 11 of this standard deals with the extent to which a product can be used by specified users to achieve specified goals with Effectiveness, Efficiency and Satisfaction in a specified context of use. This standard proposed a framework that describes the usability components and the relationship between them. In this standard, the usability is considered in terms of user performance and satisfaction. According to ISO 9241-11, usability depends on context of use and the level of usability will change as the context changes.

ISO/IEC 25000:2005: This standard is commonly known as the standard that provides the guidelines for Software Quality Requirements and Evaluation (SQuaRE). This standard helps in organizing and enhancing the process related to software quality requirements and their evaluations. ISO-25000 replaces the ISO standard ISO-9126

ISOIEC/IEEE 29119: The purpose of this internationally agreed set of standards is to support software testing within any software development life cycle or organisation. ISO/IEC/IEEE 29119 software testing standards are a set of internationally defined documents addressing the software testing concepts, processes, techniques, documents, technologies, and terms. Currently ISO/IEC/IEEE 29119 has five parts. The set of standards use a layered approach to defining software testing, which is common to many ISO standards. This set of standards presents: test definitions and concepts (part 1); test processes (part 2); test documentation (part 3); test techniques (part 4); and keyword-driven testing (part 5).

3.3.2 Testing Tools

In the nested sections below, some testing tools that support automation of tests are presented. The testing tools are classified into two categories; functional and performance testing tools. In its turn, functional testing tools are further on subdivided into API and GUI based tools.

Functional GUI testing tools

GUI testing is the process of testing the system’s Graphical User Interface of the Application Under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars - toolbar, menu bar, dialog boxes and windows, etc. Below is a list of popular used GUI testing tools:

Selenium – Popular open source functional testing tool, which is used for automating web applications for GUI testing purposes

Sikuli – It automates anything that can be seen on the screen. It uses image recognition to identify and control GUI components. It is useful to create automation scripts or when there is no easy access to a GUI's internal or source code.

Robot Framework – It is a generic test automation framework for acceptance testing and acceptance test-driven development (ATDD). It has easy-to-use tabular test data syntax and it utilizes the keyword-driven testing approach. Its testing capabilities can be extended by test libraries implemented either with Python or Java, and users can create new higher-level keywords from existing ones using the same syntax that is used for creating test cases

Watir – It is an open source Ruby library for automating tests. It interacts with a browser the same way people do: clicking links, filling out forms and validating text

HPE Unified Functional Testing (UFT) – This is a functional test tool by HP, formerly known as QTP, which supports keyword and scripting interfaces and features a graphical user interface

Cucumber – It is a software tool used by computer programmers for testing other software. It runs automated acceptance tests written in a behavior-driven development (BDD) style. It allows expected software behaviors to be specified in a logical language that customers can understand, i.e. allows the execution of feature documentation written in business-facing text

SilkTest – It is a tool for automated function and regression testing of enterprise applications

TestComplete – It gives testers the ability to create automated tests for Microsoft Windows, Web, Android (operating system), and iOS applications. Tests can be recorded, scripted or manually created with keyword driven operations and used for automated playback and error logging. It is broken out into three modules; desktop, web and mobile. Each module contains functionality for creating automated tests on that specified platform

Functional API testing tools

In API testing, the automation tool is used to send calls to the API, get the output results and log the system response. Below is a list of popular used API testing tools:

Tricentis – It is a web services testing tool that supports a wide number of protocols, such as HTTP(s) JMS, AMQP, Rabbit MQ, TIBCO EMS, SOAP, REST, IBM MQ, NET TCP

SoapUI NG Pro – It is a tool that provides extensive REST and SOAP API testing capabilities

HPE Unified Functional Testing (UFT) – In addition to GUI testing, UTF can be used to test APIs, since it provides an extensible framework helpful testing the functionality of headless systems that do not have a user interface. It helps to test the headless technologies like Databases and Webservices, JMS, etc. By using the API test conversion tool, it is possible to convert soapUI tests to UFT API tests

Parasoft – It is a feature-rich SOAP testing tool. It provides an interface for automating complex scenarios across the messaging layer, databases, ESBs, and mainframes. It supports automated test scenarios across the broad range of protocols and messages used in APIs

vREST – It provides an online solution for specification, mocking of test cases, automated testing, validation and recording of HTTP RESTful APIs

Postman – It is a lightweight plugin for Google Chrome, which can be used for testing API services. It is a non CLI HTTP client that can interact with most web APIs

HttpMaster – It is a web development tool to automate web application testing, including API testing, service testing and website testing. It is primarily used as web API test tool to automate testing of web API calls

REST-assured – It is a popular framework to test REST services in Java. Testing and validating REST services in Java is harder than in dynamic languages such as Ruby and Groovy. REST Assured brings the simplicity of using these languages into the Java domain

Curl – It is a tool to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP). The command is designed to work without user interaction

Performance testing tools

There are a wide variety of performance testing tools available in market. The tool chosen will depend on many factors such as types of protocol supported, license cost, hardware requirements, platform support, etc. Below is a list of popular used testing tools:

WebLOAD – It is a load testing tool to carry out performance and stress testing in web applications. This load testing tool combines performance, scalability, and integrity as a single process for the verification of web and mobile applications. It can simulate hundreds of thousands of concurrent users making it possible to test large loads and report bottlenecks, constraints, and weak points within an application

HP LoadRunner – This tool is capable of simulating hundreds of thousands of users, putting applications under real life loads to determine their behavior under expected loads. LoadRunner features a virtual user generator, which simulates the actions of live human users.

LoadView Testing – From small targeted tests to millions of users, this tool can find performance bottlenecks and adjust capacity plans accordingly. LoadView offers on-demand, cloud based load testing. It supports test user experience with real browsers, for a complete performance picture

HTTP Load – It is a throughput testing tool aimed at testing web servers by running several http or https fetches simultaneously to determine how a server handles the workload

Jmeter – It is one of tools commonly used for load testing of web and application servers. It can be used as a tool that supports unit tests on database connections with JDBC, FTP, LDAP, Web services, JMS, HTTP and generic TCP connections

3.4 Testing templates

The following subchapters provide sample testing templates that will be reused and adapted to support SmartEnCity evaluation and validation outcomes (testing deliverables).

3.4.1 Test scenarios and test cases

Template for Test Scenarios:

Test Scenario ID	TSn
Test Scenario Title	Name of the test scenario
Description	<Short and clear description of the test scenario>
Number of Test cases	How many test cases will be written for this test scenario

Table 5 Template for test scenarios

Template for Test Cases:

Test Case ID	TSn_TCn	**Scenario ID**	TSn	
Title	<Title of the test case> / <Type of test: functional, non-functional, performance>			
Test Case Description				
<Short and clear description of the test case>				
Test Data	<Specific data that must be used in the testing procedure>			
Acceptance Criteria	<Describe the situation, event, state or output that must be obtained in order to consider a valid execution of the Test Procedure>			
Preconditions	<Environmental conditions needed to perform the test procedure. Necessity of executing a specific Test Procedure before this one in order to provoke a specific output will be indicated here>			
Automated	<If automated, indicate the name of the tool and/or script used. If automated, there is no need to complete the following steps>	**Repeat**	<Indicate the number of repetitions needed. Leave a blank space to annotate the number of repetitions that is being executed. The Tester has to print the Testing Procedure as many times as indicated in this field>	
Steps				
#	**Action**	**Expected Output**	**Output**	**Passes (Y/N)**
1	<Describe the action needed to execute the testing procedure>	<Indicate the expected output of the previous action>	<Space for the tester to handwrite the obtained result>	"Y" If expected output and Output matches. "N" If not.
2	---	---	---	---
Summary & Comments				

<Summary of the results and special events that may occur during the testing process: errors or exceptions found, unexpected behaviours, etc.>

Table 6 Template for test cases

3.4.2 Test summary report template

Template for reporting on tests:

Test Summary Report			
Test Case ID	**Test description**	**Comment**	**Decision**
<Same as in Test Case template>	<Short and clear description of the test case>	<Additional information or problems encountered during execution and differences with the Test Case>	<OK, NOK, POK, NR, NC>
---	---	---	---
---	---	---	---

Table 7 Template for reporting on tests

After executing a test, the decision is defined according to the following rules:

- OK: The test is set to "OK" state when all steps defined in the Test Case are in "Y" state, i.e. the real result is compliant to the expected result
- NOK: The test is set to "NOK" state when all steps of the Test Case are set to "N" state or when the result of a step differs from the expected result
- Partial OK (POK): The test is set to "POK" state when at least one step of the Test Case is set to "N" state or when the result of a step is partially compliant to the expected result. Criteria to set if result is Partial OK may be qualitative
- NOT RUN (NR): The test is set to “NR” when the Test Case has not yet executed
- NOT COMPLETED (NC): The test is set to "NC" state when at least one step of the Test Case is set "NR" state

3.4.3 Correction measures report template

Template for correction measures:

Correction Measures Report			
Test Case IDs	**Qualitative assessment**	**Quantitative results**	**Correction measures**
<Same as in Test Case template>	<Qualitative assessment of the set of tests>	<Statistics about performed tests>	<Summary of the results of the testing. Identifies all resolved issues and summarizes the details of their resolution, and lists any outstanding issues>

e.g. TC1, TC2, TC3, TC4	e.g. All API tests passed. GUIs are not optimized for tablet devices. All performance tests passed, etc.	e.g. % of tests OK, % of tests NOK % of tests POK % of tests NR % of tests NC Give also statistics about bugs and enhancements: • Total number • Number of Critical • Number of Major • Number of minor • Number of enhancements	e.g. TC1 revealed problems with the legibility of visual elements on the chart. The developer team corrected the observed defects and the amended code passed the regression test. However, it was not possible within the time scales of the testing task to perform regression test of the associated navigation requirements. This issue remains outstanding, and should be observed following installation and use in the live environment.

Table 8 Template for correction measures

4 Integration and Validation in SmartEnCity

This section describes in detail the check plan for the validation of the different modules of the ICT platform developed in the previous tasks (T6.1-T6.4) of the SmartEnCity project. Based on the methodological approach described in the previous section (Section 3), this section details the test planning to be carried out in order to verify that each developed component of the platform works as expected in terms of behaviour and performance.

The types of tests to be performed to demonstrate the correct operation of the developments made for the SmartEnCity CIOP platform are classified according to different criteria. Firstly, according to the number of modules involved in the validation, Unit Tests will be defined per module and Integration Test allows to validate the correct functioning of the connection between modules. Focusing on the object to be validated, we identify two main types of tests: those that test behaviour (Functional Tests) and those that analyse performance (Performance Test). Finally, the tests will be adapted to each type of module included in the platform (e.g. Repositories, APIs or Applications)

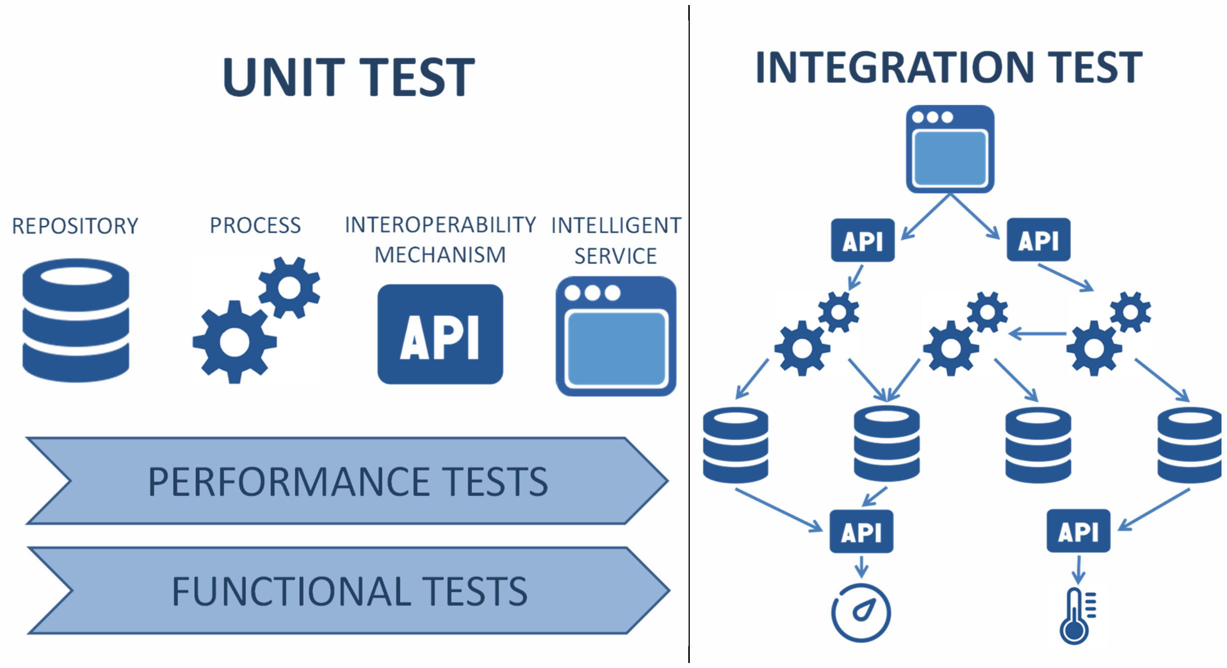


Figure 3 Unit Test and Integration Test

4.1 Performance Tests

The performance tests are tests that allow to identify the correct operation and with an adequate quality of service of the software components. There are different types of tests that allow to evaluate the performance of the system to be validated. The categories considered most relevant to the evaluation in SmartEnCity are:

- **<u>Response Time:</u>** It measures the time that elapses since a request is sent until a response to the request is received. This type of test applies to both transactions and database queries. Response time is important in performance testing because it represents how long a user must wait for a request to be processed. This test is measured as the mean response time of an operation or set of significant operations repeated a representative number of times.

- **Data Volume Load Test:** This test allows to validate the behaviour of a software module against transactions or requests involving a high volume of data. The request or transaction can be an insertion or retrieval. This test allows to identify the limits of data exchange sizes to be made in transactions to maintain an adequate response time. This test should allow comparison of response times of transactions that require different volumes of data.

- **Concurrency Load Test**: This test is intended to validate the behaviour of a module under a large number of transactions or simultaneous requests made by several users. This test is important to verify that the response time is kept within reasonable limits when the system is subjected to significant load by concurrent users. The test must allow to identify the average response time of a set of significant transactions carried out simultaneously by several connected users.

Technologies for conducting performance tests generally depend on the component to be tested (e.g. repository or API) and the technologies used for the development of the component. These performance tests will typically be performed automatically by programming a sequence of commands or script that emulates the operation of a human or another machine. When the requirements for the results of these tests (response times or load requirements) are available in advance, the validation of the tests will be made against those requirements. In case these requirements are not known in advance, the tests will identify limits and intervals that must be analysed later.

The following table must be completed for the performance tests of each module to which the performance test battery applies.

Test Type	Implementation	Result
Response Time	This test is measured as the mean response time of an operation or set of significant operations repeated a representative number of times.	Limits or threshold for the response time
Data volume Load Tests	This test should allow comparison of response times of transactions that require different volumes of data.	Curve representation of response time for different volume of data
Concurrency load Tests	The test must allow identifying the average response time of a set of significant transactions carried out simultaneously by several connected users.	Curve representation of response time for different number of concurrent users.

Table 9 Battery of test to be implemented for performance evaluation

The following table represents the template to be filled with the information of the performance evaluation of each of the modules.

Module Name	<End-point name>
Test Type	**Response Time**
Testing Tool	<Name of the tool or own developed>
Acceptance Criteria	<Expected output to be considered valid>
Implementation	
<This test is measured as the mean response time of an operation or set of significant operations repeated a representative number of times.>	
Result	
<Limits or threshold for the response time.>	
Test Type	**Data volume Load Tests**
Testing Tool	<Name of the tool or own developed>
Acceptance Criteria	<Expected output to be considered valid>
Implementation	
<This test should allow comparison of response times of transactions that require different volumes of data.>	
Result	
<Curve representation of response time for different volume of data.>	
Test Type	**Concurrency Load Tests**
Testing Tool	<Name of the tool or own developed>
Acceptance Criteria	<Expected output to be considered valid>
Implementation	
<The test must allow identifying the average response time of a set of significant transactions carried out simultaneously by several connected users.	
Result	
<Curve representation of response time for different number of concurrent users.>	

Table 10 Template for performance test

4.2 Functional Tests

The objective of the functional tests is to verify that the system or module developed works according to the specifications previously defined. The functional tests prove that the output results of the processes conform to what is expected from the specified input data. Functional tests are usually executed manually although in extraction, transformation and data loading (ETL) processes, these tests could be automated. Among the types of existing functional tests, only acceptance tests will be considered in SmartEnCity. The purpose of acceptance tests is to validate that a system complies with the expected functionalities and allow the user of that system to determine its acceptance from the point of view of its functionality.

The following table must be completed for the functional tests of each of the modules to which the functional test applies.

Module Name	<End-point name>			
Test Type	**Functional Test**			
Testing Tool	<Name of the tool or own developed>			
#	**Functionality**	**Input**	**Output**	**Result (Ok/NOk)**
1	<Functionality name and short description>	<Required inputs>	<Expected Outputs>	<Including short explanation about the reason in case of NOk>
2				

Table 11 Template for functional tests in API testing

Module Name	<End-point name>			
Test Type	**Functional Test**			
Testing Tool	<Name of the tool or own developed>			
Test Case Title	<Title of the test case>			
Test Case Description				
<Short and clear description of the test case>				
Steps				
#	**Action**	**Expected Output**	**Output**	**Passes (Y/N)**
1	<Describe the action needed to execute the testing procedure>	<Indicate the expected output of the previous action>	<Space for the tester to handwrite the obtained result>	"Y" If expected output and Output matches. "N" If not.
2	---	---	---	---
Summary & Comments				
<Summary of the results and special events that may occur during the testing process: errors or exceptions found, unexpected behaviours, etc.>				

Table 12 Template for functional tests in GUI testing

4.3 Unit Testing

The unit tests to be carried out in SmartEnCity allow to identify that each developed module works correctly in isolated mode. These tests will be performed for each of the modules without the need to do a complete system deployment or partial integration with the rest of the platform components. Four different types of software components have been identified in the reference architecture of SmartEnCity (Repositories, Processes, Interoperability Mechanisms and Intelligent Services) (see Figure 3). The tests to be carried out for each of the different types of identified components are detailed below. Subsequently, the following sections will detail the specific tests to be performed for each of the components developed for the SmartEnCity platform, section 5 identifies the end-points or components to be tested and in section 6 are described the tests developed for each one of the components.

4.3.1 Repositories Testing

The implementation of the SmartEnCity reference architecture (described in Deliverable 6.3) identifies 8 different repositories (Vertical, KPI, Historical, Structural, GIS Structural, GIS, Configuration and Real Time repositories). The test of a repository consists basically in securing CRUD (Create, Read, Update and Delete) operations. Since it is a reference platform, the process of validating repositories should be independent of the database technologies used for its implementation. All repositories of the SmartEnCity reference platform are exposed to the rest of the platform modules through an interface (API). Thus, the implementation of the tests of the repository will be carried out by testing the interfaces developed to expose the repository to the other modules.

SmartEnCity repositories will perform both performance tests and functional tests (see Table 9 and **¡Error! No se encuentra el origen de la referencia.**).

4.3.2 Processes Testing

Knowledge layer in SmartEnCity reference architecture implementation integrates the components for information treatment, management and exploitation. Data transformation and processing is one of the main purposes of the knowledge layer. Every software module which collects data, transform them and provide an output can be considered a process. SmartEnCity reference architecture integrates different kinds of modules included in this category. Software modules that support the extraction, transformation and loading (ETL) of data are part of this category, but also those modules able to generate patterns or behaviours from a collection of data. The analytical data treatment through business intelligence processes is also a core capability of the reference architecture. The processing of the geospatial information in order to extract relevant data is also a module which belongs to this category.

The list of processing modules developed in the implementation of the SmartEnCity reference architecture is completed in the following section (see Section 5). For those modules exposing an API to the rest of the modules, the API will be tested. For those modules implemented as a script, commands in the script will be launched and outputs will be analysed.

As a general rule the processes in SmartEnCity will require both types of test (performance and functional) (see Table 9 and **¡Error! No se encuentra el origen de la referencia.**). Special cases will be analysed to perform only some of them.

4.3.3 Interoperability Mechanisms Testing

Interoperability mechanisms are used in SmartEnCity reference architecture to implement two of the layers (Acquisition / Interconnection Layer and Interoperability Layer). Interoperability mechanisms are implemented to provide an interface to offer different functionalities and services to third parties that are interested on consuming those resources.

Web services and APIs are the most used mechanisms for interoperability in SmartEnCity reference architecture. Both Web services and APIs can be tested in a similar way. Tools for testing web services and APIs send calls to the interface, get output and log the system's response.

The list of tools existing for testing Web services and APIs is really huge, most of them are open source and for free. Most of them are focused on the functional testing of the interfaces. Performance tests require other tools not so accessible, but quite simple scripts or programs can be developed for such purpose without great effort.

Interoperability mechanisms in SmartEnCity will require the implementation of both performance tests and functional tests (see Table 9 and **¡Error! No se encuentra el origen de la referencia.**).

4.3.4 Intelligent Services Testing

Intelligent services in SmartEnCity reference architecture are services and applications within the area of different vertical domains (energy, environment, mobility, etc.) that have been developed based on the Smart City infrastructure and available data sources. The intelligent services layer interacts with the platform through the interoperability layer and consumes the data generated in the knowledge layer.

Several intelligent services have been developed in SmartEnCity for any of the three verticals addressed in the project (see the complete list in Section 5).

Testing the final application generally means to test a tool through a GUI. A set of case studies is defined in order to cover all or the most of the functionalities specified. These tests are usually performed manually.

Intelligent Services in SmartEnCity will require in all cases the implementation of functional tests while performance tests will be carried out only when any specific performance criteria is critical for it usage (see Table 9 and **¡Error! No se encuentra el origen de la referencia.**).

4.4 Integration Testing

The purpose of the integration tests is to ensure that each of the modules developed works as expected once integrated with the rest of the modules of the platform. That means validating the developments through end-to-end tests (See Figure 3). These tests will be performed mostly manually.

To carry out the SmartEnCity integration tests, the following steps will be performed:

1. **Definition of test scenarios:** That means the identification and brief description of one or several test scenarios that present a representative sample of the operational flow of the components of the different layers of the SmartEnCity reference architecture.
2. **Identify the test environment:** It will detail the hardware, software and network configuration components that will be used to carry out the tests of each of the previously described scenarios.
3. **Identify acceptance criteria**: For each of the scenarios the acceptance criteria to consider for the validation of the scenarios will be established. When it is not possible to establish detailed acceptance values for each of the criteria, the results of the tests will represent the mean values obtained and at the end of the tests the corresponding conclusions will be extracted according to the resulting values.
4. **Define testing plan**: The detailed test plan for each scenario will include the following information:
    a. Scenario Name
    b. Scenario Description
    c. Test Environment
        i. Hardware Environment
        ii. Software Environment
        iii. Network Configuration
    d. Acceptance Criteria
        i. Criterion
        ii. Value
    e. Involved Components / Modules
        i. Component Name
        ii. Technology
        iii. Query / Transaction
        iv. Result

The following table provides all the information necessary to be completed for each of the test scenarios

Scenario Id:	<ID representing the scenario>	
Scenario Name:	<Name of the Scenario>	
Scenario Description	<Short description of the scenario>	
Test Environment		
Hardware	**Software**	**Network Config.**
<Description of the hardware	<Description of the software	<Description of the network

environment used for the testing>	environment used for the testing>	configuration used for the testing>

Acceptance Criteria	
Criterion	**Value**
<Name of the criterion for acceptance>	<Value for acceptance>

List of Modules Involved				
#	**Module**	**Technology**	**Query / Transaction**	**Result**
1	<Module name>	<Technology/Protocol used in the connection>	<Query or Transaction performed to implement the connection>	<Ok / NOk. Including short explanation about the reason in case NOk>

Table 13 Template for Integration Tests

4.5 Added-value Services Testing

Within tasks T3.7, T4.7 and T5.7, the added value services will be implemented. Therefore, they need to be tested in order to ensure their proper behaviour. There are two required steps when talking about added value services:

1. Functional tests which ensures the implementation complies with the design (mainly requirements). This set of tests is usually performed with dummy data under a controlled environment. In this way, the information is injected by the quality assurance responsible, being the output a pre-defined value.
2. Integration tests that are run with the aim of assuring the added value service is working under a real environment and complying with the requirements. That is to say, accessing databases, information of the platform, etc.

Nevertheless, these are not the only tests that need to be rendered and, as mentioned before, performance tests are necessary to ensure certain quality parameters. Among them, response time, concurrence load and data load are pivotal. Complementary, other tests are also useful, for instance, extensibility, interoperability and modularity.

However, before the definition of the test, the list of added value services is required. Although the procedures are more or less standards, some particularities need to be taken into consideration. Table 14 is based on the identification of added-value services carried out in previous tasks of WP6 (mainly T6.1 and T6.6) and gathers the added value services identifying the scope and the city that implements it.

Type of service	Description	SCOPE				LIGHTHOUSE		
		ESCO	City	Intervention area	City council	Vitoria-Gasteiz	Tartu	Sonderborg
Energy assessment	**Energy use forecast** Estimation of the energy needs at district level for assuring the comfort values and the supply into all the dwellings. In this way, weather forecast would help to estimate the individual dwelling energy needs which will lie in an estimation of the required energy to fulfill the energy demand. Then, a set of recommendations could be emitted to the ESCo with the aim at managing energy in an efficient way.	X				X		
	Home Energy Consumption Monitoring This service will monitor electrical energy consumption on dwellings. It will collect electrical consumption information from metering devices installed in the dwellings and store them in a central location. This service will provide an HMI for the residents to let them know about their energy consumption patterns, set consumption goals and thresholds, receive advice on how to reduce their consumption, and will allow comparing their consumption with other residents in their area. The goal of the services is to empower the resident to engage in energy consumption reduction.			X		X	X	
	TV broadcast service for energy information where a new TV channel will be established to inform about energy consumption in the building.			X		X		
	Integration of the building/district energy consumption on top of GIS maps.				X	X		
	Integrated electrical and thermal network energy management systems (at home, building and district level)			X			X	
	Monitoring of most relevant Key Performance Indicators				X	X	X	X

	Indoor KPIs (e.g. temperature, humidity, energy) at different levels (dwelling, building, district). Outdoors KPIs (e.g. street lights, temperature, speed of wind, humidity, waste collection).							
	<u>Heating performance</u> This service will show the performance indicator of heating system at the block level. This will show the difference between energy produced and transferred to the block and energy received back. Performance comparison can be performed – before intervention and after.			X				X
	<u>CO2 performance</u> Service showing the total CO2 emission levels generated while producing all energy required for the block. Performance comparison can be performed – before intervention and after. Comparison of performance with solar panels and without.				X			X
	<u>Solar panels</u> • **Production** Service showing total energy generated at installation site at specific time interval together with CO2 emission levels and electricity prices. • **Consumption** Service showing total energy consumed generated by solar panels at installation site at specific time interval together with CO2 emission levels and electricity prices. • **Performance** Indicator showing how much the solar panels are utilized at the installation site.				X			X
	<u>Electricity consumption per m2</u> Service showing kWh used per m2 in the house. House parameters like size, condition, insulation, people, weather, degree days etc. are evaluated. Comparison can be made – before intervention and after in order to apply the protocol for energy assessment according to			X				X

	WP7 specification.							
Sustainable mobility	**Charger point business** Service showing business level of charging point.				X			X
	Number of charging CO2 emissions reduction An analysis of the CO2 emissions that will be avoided can be made on the basis of km driven and the equivalent fuel consumption.				X	X	X	X
	Charging points geolocation information Information about the availability of EVs recharging points		X			X		X
	Charging point performance Charger point utilization in terms of CO2 and electricity price levels.				X			X
	Geolocation of rental cars and bikes Geolocation tracking for billing purposes and KPI calculations		X					
	Geolocation of public transport Geolocation tracking of the bus location and occupancy values to inform citizens about the status for the next upcoming bus. It is fed by real-time information about the location of the bus and the occupancy index with the aim of providing useful information to the inhabitants to make use of the public transport.		X			X	X	
Citizen engagement	**CO2 savings** Information on saved CO2 emission levels gathered from charging points, electricity consumption in living blocks, etc.				X			X
	Surveys Launching surveys in the district about the building retrofitting preferences of the neighbors			X		X	X	X

	Feedback Providing information about success stories in building retrofitting			X			X	X
	Feedback about failures in the services (e.g. EVs charging points, public transportation, bike sharing) in order to provide support to the maintenance of the city facilities.		X				X	
	Information about schedules and routes of public transportation. Instead of dynamic information on real-time, this service is static in order to provide inhabitants with information about the routes and services that are provided by public transportation.		X				X	
Integrated infrastructure	**CO2 status** Monitoring of present status of savings of CO2 emission levels. Indicators "green", "yellow", "blue".				X			X
	Electricity price status Monitoring of present status of electricity price. Indicators "green", "yellow", "blue".				X			X
Industry	**CO2 status** Service showing industry site/equipment (like water pump) performance in terms of CO2 emission levels.				X			X

Table 14 List of added value services

Once the services are defined, the test plan may be defined for the added value services. Basically, all the services are similar; they make use of data from the platform to run calculations or algorithms so as to present a result in a graphical interface, via app or TV channel or common Web access through browser. In this way, a common and generic test plan is defined, in spite of the fact that each service has its own peculiarities. However, detailing all the test cases would increase the size of the document and a general approach applies for all the services. Specific values for expected outputs need to be detailed when the test will be performed under task T3.7, T4.7 and T5.7.

Table 15 summarizes in a generic way the functional tests to be carried out in order to assure the expected functionalities under a controlled environment with dummy data. Table 16 summarizes the integration tests which test the services under real conditions (data). Both tables are based on the templates defined in previous sections (Section 4.2 and Section 4.4)

Finally, the set of performance tests defined in section 4.1 will be applied to measure the performance. Apart from the aforementioned tests, there are other performance indicators that could to be extracted; in this case, they are necessary:

- Extensibility: This indicator measures the possibilities of extending the capabilities of the added value service. It means the ability of the system to tolerate additional features.
- Interoperability: This indicator provides information about the capability of the added value service to interconnect to different platforms under the same interface specifications.
- Modularity: This measures the degree of partitioning of the added value service in order to increment the manageability of the component.
- User-friendliness: Finally, in terms of visualization, this indicator obtains how intuitive the service is.

Module Name	Added value services			
Test Type	**Functional Test**			
Testing Tool	Own developed			
Test Case Title	Added Value Services test			
Test Case Description				
This test plan treats the evaluation of the added value services from the functional perspective with the aim of checking the requirements are correctly implemented. For this purpose, dummy data are used in order to have a controlled environment.				
Steps				
#	**Action**	**Expected Output**	**Output**	**Passes (Y/N)**
1	Functionality "n" is working according to the requirement.	The output is basically a checklist to determine whether the functionality is covered or not. Then, a mapping between	To be filled in T3.7, T4.7 and T5.7.	"Y" If expected output and Output matches. "N" If not.

		requirements and real behaviour of the service is made.		
2	Calculations are made on basis of the specifications.	The expected output is aggregated data with values according to the theoretical calculations (controlled environment).	To be filled in T3.7, T4.7 and T5.7.	"Y" If expected output and Output matches. "N" If not.
3	Algorithms run the operations with the data to obtain the result.	The algorithms run under its defined specifications and with the theoretical result from dummy data.	To be filled in T3.7, T4.7 and T5.7.	"Y" If expected output and Output matches. "N" If not.
4	Results are visualized under a user-friendly screen.	The graphical interface shows the information that is obtained as result of the aforementioned actions.	To be filled in T3.7, T4.7 and T5.7.	"Y" If expected output and Output matches. "N" If not.
Summary & Comments				
When errors, bugs or exceptions are encountered, as the tests are run under controlled conditions, therefore the contingency plan is to solve these bugs as common procedures in software development.				

Table 15 Functional tests plan for added value services

Scenario Id:	Integration of added value services in the platform	
Scenario Name:	Intelligent services integration	
Scenario Description	Once the controlled test and run and the services tests are passed, next step is to work under real environment and real data. Therefore, these tests are dedicated to the tests with real data.	
Test Environment		
Hardware	**Software**	**Network Config.**
Data server, mobile and/or personal computer.	Database, app and/or service.	IP communication between the data repositories of the platform and the deployed services.
Acceptance Criteria		
Criterion	**Value**	
Functional service	The service is presenting the results as expected.	
Data retrieval	Data are obtained from the repositories.	
List of Modules Involved		

#	Module	Technology	Query / Transaction	Result
1	Internal data repositories	REST API to access data.	select * from data_repository where condition1 and condition2	To be filled when testing in T3.7, T4.7 and T5.7.
2	Added value service	iOS and/or Android app and/or Web browser.	HTTP GET/POST	To be filled when testing in T3.7, T4.7 and T5.7.
3	GIS function	OGC	GIS query	To be filled when testing in T3.7, T4.7 and T5.7.
4	External data repositories	REST API to access data.	JSON queries/XML files	To be filled when testing in T3.7, T4.7 and T5.7.
5	TV channel	DTV	Broadcast image	To be filled when testing in T3.7, T4.7 and T5.7.

Table 16 Integration test plan for added value services

4.6 Monitoring Testing

4.6.1 Introduction

The result of SmartEnCity CIOP will be deployed as a set of web applications and the corresponding IT infrastructures. This platform is supported on a complex digital environment composed of devices, technologies and diverse networks that can have effect on the final result received by the user.

Therefore, it is necessary within the process of evaluation and validation of applications based on the SmartEnCity platform to continuously monitor and test their networks, servers, applications and business logic to ensure a satisfactory service.

4.6.2 Monitoring Topics

The checklist for monitoring testing contains the following topics:

Heartbeat page / Service availability monitor

Simple webpage displaying service(s) status. It is usually updated manually by Network Operation Center (NOC) or any other responsible party to notify customers on service outages or degradation. It can provide additional details such as estimations on service recovery. This will greatly reduce load on customer support when SmartEnCity CIOP is experiencing troubles in normal operation.

Service monitoring

Service design must consider and provide monitoring endpoints. This will simplify monitoring setup and also guarantee that monitoring tools monitor correct parameters. While it is possible to monitor services without specific monitoring endpoints, it will be less effective since developers of the service have great in-depth knowledge of how service should behave and what are the critical parameters to monitor. Additional checks might be implanted without specific monitoring endpoints – such as memory consumption, system load and response times. It would be preferable that system can detect anomalies in services instead of having static thresholds.

Performance monitoring

Performance is usually affected by one of the following: changes in service itself, changes in usage patterns or changes in infrastructure. Detecting changes in services performance and understating the cause will help to prevent service degradation or even service outage. Something as simple as software upgrade can cause performance problems and affect users' ability to use the service. Performance monitoring is done by collecting service and system metrics, visualizing both historical and real-time data and creating alerts when predefined thresholds are met. Performance tools are usually part of the monitoring solution.

Device monitoring

In addition to service monitoring, SmartEnCity CIOP needs to monitor different devices, such as sensors and other devices connected to the platform. While generic monitoring tools provide majority of the functionality needed for device monitoring, it is important to consider additional requirements that are specific to devices:

- Mechanisms to detect when device is dead or malfunctioning
- Keep track when device was online last time
- Garbage detection: devices data deviates from preset parameters, this could indicate that device is malfunctioning

Security and auditing

Logs from each service needs to be collected and stored in central location. Monitoring tools must detect and alert unauthorized access attempts. Those attempts can be detected, but not limited to, in one of the following areas:

- Multiple login failures for a user
- Login attempts from IPs from different countries/regions
- Login spam – repeated attempts to log in as a different user from the same IP address
- Other anomalies such as trying to access undefined endpoints

Monitoring team

This team is responsible for monitoring the services and alerting service providers when service degradation or outage occurs. Responsibilities of the team:

- Opens and manages incidents when service failure occurs
- Handles communication between stakeholders

- Monitors and detects trends in performance monitoring tools and detects events that could lead to service degradation
- Does Tier 1 support in case of service failures – performs actions that are documented by service provider and are needed to validate failure or recover services state

4.6.3 Monitoring Tools

There are many tools for the continuous monitoring of digital environments. Most users combine several different tools to monitor and run their IT infrastructure. The tools are generally available in three distribution models: software as a service (SaaS), open source software and proprietary solutions.

Of these existing tools Nagios and Zabbix stand out among the open source and free solutions, Acronis and LogicMonitor between software as a service and Paessler and ManageEngine among proprietary solutions. Those and other similar tools must be configured according to each Lighthouse specifics, there's no one tool that can solve the monitoring problem in general.

Also, it is important to keep in mind that monitoring is an ongoing process that requires human resources that must be taken into account when selecting service and development providers.

4.7 Global Access System

The purpose of this module within the platform is to develop a global user authentication system that allows access to the data obtained from the different demo areas of both KPIs and end-points and display them as a friendly way.

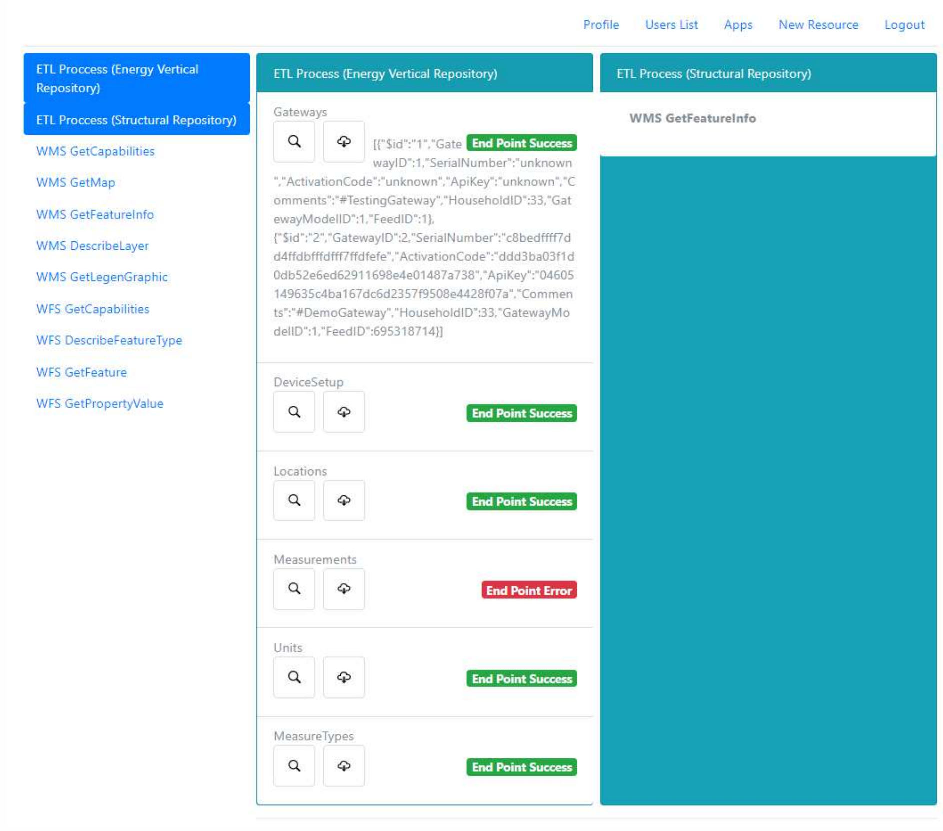
This system is defined as the entrance and global access point to collect and compare indicators and performance of different cities included in the SmartEnCity CIOP.

The main functionalities of the global access system are:

1. Restricted access to registered users with different profiles: administrator, city manager, services company, etc.
2. Shown figures and graphs with values gathered from ETL Processes grouped by e.g. category, building or city, and the choice of change the measures type and units gathered from same webservices.
3. Service, performance and device monitoring of the system components by creating alerts when anomalies are detected (e.g. device malfunction, response time exceeded or service not available)

Three main areas are envisaged for this access system in SmartEnCity CIOP:

1. **End Points area:** This page allows access to endpoints displayed in each demo showing if the connection has been successful and report the response in JSON format


Profile Users List Apps New Resource Logout

ETL Proccess (Energy Vertical Repository)
ETL Proccess (Structural Repository)
WMS GetCapabilities
WMS GetMap
WMS GetFeatureInfo
WMS DescribeLayer
WMS GetLegenGraphic
WFS GetCapabilities
WFS DescribeFeatureType
WFS GetFeature
WFS GetPropertyValue

ETL Process (Energy Vertical Repository)

Gateways — End Point Success

```
[{"$id":"1","GatewayID":1,"SerialNumber":"unknown","ActivationCode":"unknown","ApiKey":"unknown","Comments":"#TestingGateway","HouseholdID":33,"GatewayModelID":1,"FeedID":1},
{"$id":"2","GatewayID":2,"SerialNumber":"c8bedffff7dd4ffdbfffdfff7ffdfefe","ActivationCode":"ddd3ba03f1d0db52e6ed62911698e4e01487a738","ApiKey":"04605149635c4ba167dc6d2357f9508e4428f07a","Comments":"#DemoGateway","HouseholdID":33,"GatewayModelID":1,"FeedID":695318714}]
```

DeviceSetup — End Point Success

Locations — End Point Success

Measurements — End Point Error

Units — End Point Success

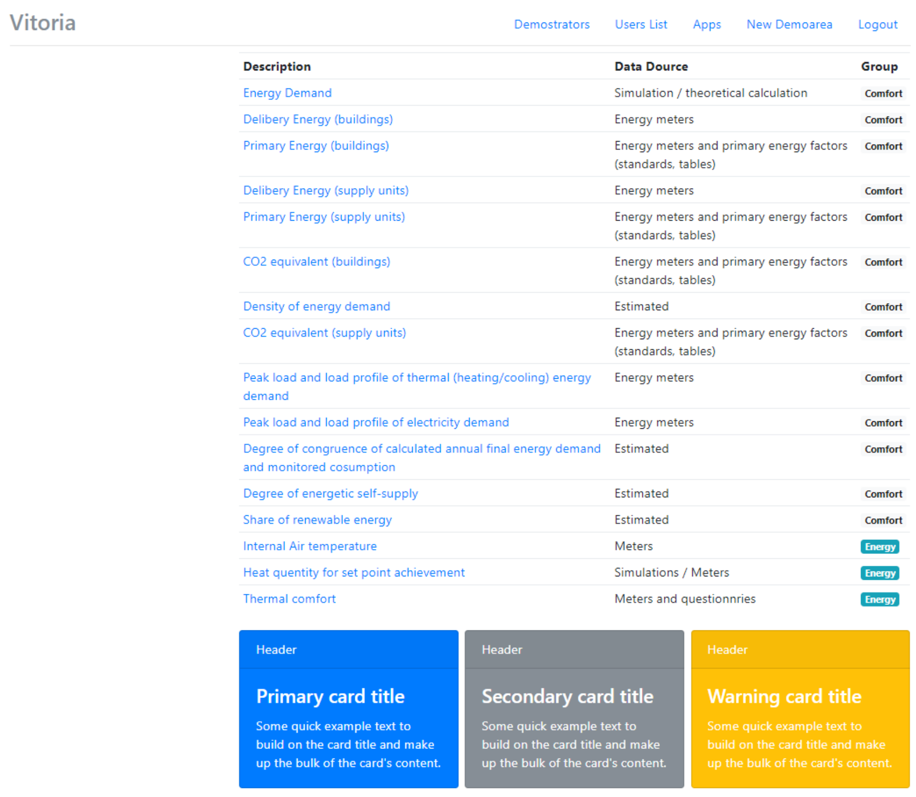
MeasureTypes — End Point Success

ETL Process (Structural Repository)

WMS GetFeatureInfo

Figure 4 Global Access System End Points area

2. **KPIs area:** This page allows access to information of the different KPIs showing each one in the most optimal way through widgets, graphs or tables


Vitoria

Demostrators Users List Apps New Demoarea Logout

Description	Data Dource	Group
Energy Demand	Simulation / theoretical calculation	Comfort
Delibery Energy (buildings)	Energy meters	Comfort
Primary Energy (buildings)	Energy meters and primary energy factors (standards, tables)	Comfort
Delibery Energy (supply units)	Energy meters	Comfort
Primary Energy (supply units)	Energy meters and primary energy factors (standards, tables)	Comfort
CO2 equivalent (buildings)	Energy meters and primary energy factors (standards, tables)	Comfort
Density of energy demand	Estimated	Comfort
CO2 equivalent (supply units)	Energy meters and primary energy factors (standards, tables)	Comfort
Peak load and load profile of thermal (heating/cooling) energy demand	Energy meters	Comfort
Peak load and load profile of electricity demand	Energy meters	Comfort
Degree of congruence of calculated annual final energy demand and monitored cosumption	Estimated	Comfort
Degree of energetic self-supply	Estimated	Comfort
Share of renewable energy	Estimated	Comfort
Internal Air temperature	Meters	Energy
Heat quantity for set point achievement	Simulations / Meters	Energy
Thermal comfort	Meters and questionnries	Energy

Header
Primary card title
Some quick example text to build on the card title and make up the bulk of the card's content.

Header
Secondary card title
Some quick example text to build on the card title and make up the bulk of the card's content.

Header
Warning card title
Some quick example text to build on the card title and make up the bulk of the card's content.

Figure 5 Global Access System KPIs area

3. **Administrator area:** Administrator page allows logging in, abouts and new user registration. If the registration form is completed correctly, the tool requests confirmation through an email. Next page shows the demo areas list available.

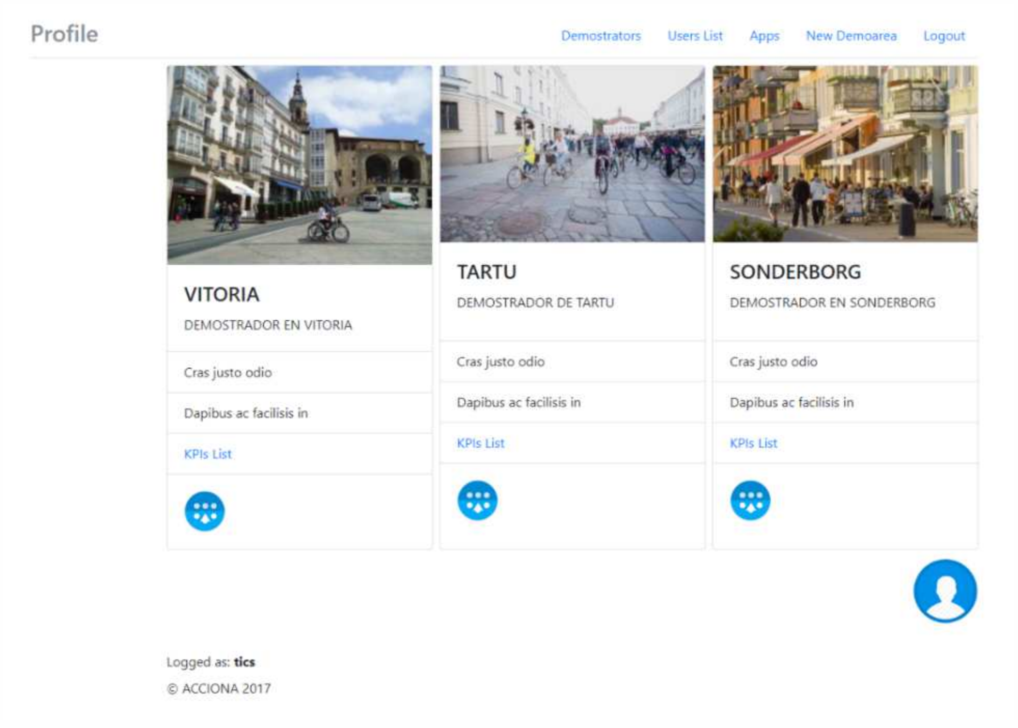


Figure 6 Global Access System Administrator area

5 Identification of end-point elements to Test

5.1 Introduction

The reference architecture proposed for SmartEnCity is presented in the following Figure (Figure 7). End-points developed in SmartEnCity CIOP fits the proposed layers and the corresponding architecture.

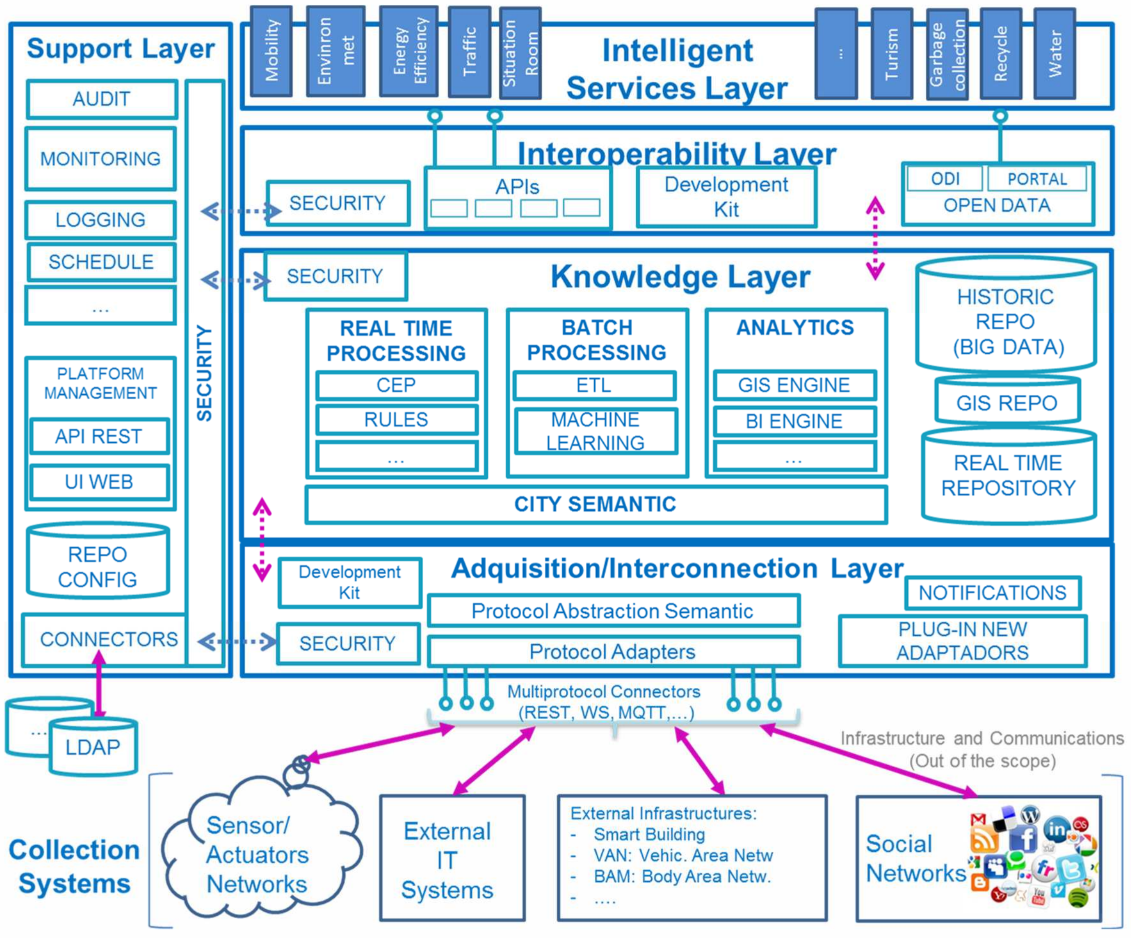


Figure 7 Smart Cities General Architecture

SmartEnCity CIOP is composed of a set of end-points which provides the functionality for some of the modules identified in the reference architecture. Modules are integrated in the supporting layers of the architecture. The following table shows the list of end-points to be tested, including the layer of the CIOP and the type of the end-point.

CIOP-LAYER	END-POINT	END-POINT TYPE
Acquisition / Interconnection Layer	Data Acquisition about Energy	Interoperability Mechanism
	Services for integrating Open Data	Interoperability Mechanism
Knowledge Layer	ETL Processes (Energy Vertical Repository)	Process
	ETL Processes (Mobility	Process

	Repository)	
	ETL Processes (Historical Repository	Process
	ETL Processes (Structural Repository)	Process
Interoperability Layer	API for Energy Services	Interoperability Mechanism
	API for Mobility Services	Interoperability Mechanism
	API for Citizen Engagement Services	Interoperability Mechanism
	GIS Repo Services	Repository
	GIS Structural Data Repository Services	Repository
Intelligent Services Layer	Energy Application	Intelligent Service
	KPI Calculation Services	Intelligent Service
	Global Access System	Intelligent Service

Table 17 List of end-point to test

For each of the end-points developed in the SmartEnCity CIOP, the following information is collected and included in the following tables.

- **End-Point Name:** The name of the end-point
- **End-Point Type:** The end-points including in the COIP platform are of heterogeneous nature, being the most common types: API, Webservice, Database and Application.
- **Protocol:** The way the end-point communicates with others to access and collect information can be very diverse, some examples: REST, SOAP, MQTT, AMQP, or JDBC.
- **Method:** This parameter indicates how data is sent when communicating with other modules. It is associated to each protocol. Some of the most common methods are: GET, POST, Client-Server or Publisher-Consumer.
- **Input Data Format:** Defines the format of the input data of the end-point. Examples are: Text, CSV, XML, JSON, or User Interaction.
- **Output Data Format:** Defines the format of the output data of the end-point. Examples are: Text, CSV, XML, JSON, or User Visualization.
- **CIOP Layer:** Indicates the layer of the reference architecture to which the end-point belongs.
- **Description:** Detailed description of the end-point and the main features, functionalities or services provided by the end-point.

5.2 Data Acquisition about Energy

End-Point Name	Data Acquisition about Energy
End-Point Type	WebService
Protocol	REST

Method	POST
Input Data Format	Text / CSV
Output Data Format	HTTP Response Codes (2XX, 4XX, 5XX …)
CIOP Layer	Acquisition / Interconnection Layer
Description	
Information coming from different sensors as temperature sensors and energy consumption meters is collected on time series tables (Real Time Repository)	
Services	
URL	http://${IP_REPO_REALTIME}/activate/${GATEWAY_ACTIVATION}.csv http://${IP_REPO_REALTIME}/v2/feeds/${GATEWAY_FEEDID}/datastreams/10.csv http://${IP_REPO_REALTIME}/api/${GATEWAY_FEEDID}.csv" --data-raw "${SENSOR_DATA}
Observations	
Test these services is a complex process since first there is a registration and stream selection actions. To test the services we should use cURL (https://curl.haxx.se/) For the provisioning: curl --http1.1 -H "Host: provisioning.connectedenvironments.com" --url "http://${IP_REPO_REALTIME}/activate/${GATEWAY_ACTIVATION}.csv" For the agreement on the stream: curl --http1.1 -H "api.pachube.com" -H "X-PachubeApiKey: ${GATEWAY_APIKEY}" --url "http://${IP_REPO_REALTIME}/v2/feeds/${GATEWAY_FEEDID}/datastreams/10.csv" For the data: curl --http1.1 -X PUT -H "www.pachube.com" -H "X-PachubeApiKey: ${GATEWAY_APIKEY}" -H "Content-Type: text/csv" --url "http://${IP_REPO_REALTIME}/api/${GATEWAY_FEEDID}.csv" --data-raw "${SENSOR_DATA}" Possible testing parameters: IP_REPO_REALTIME: 217.76.242.53 GATEWAY_SERIAL="c8bedffff7dd4ffdbfffdfff7ffdfefe" GATEWAY_ACTIVATION="ddd3ba03f1d0db52e6ed62911698e4e01487a738" GATEWAY_FEEDID="695318714" GATEWAY_APIKEY="04605149635c4ba167dc6d2357f9508e4428f07a" IP_REPO_REALTIME="217.76.242.53" export SENSOR_DATA="30.50,480,243,55,156.7,0,0,99,0,0,0"	

Table 18 Description of Data Acquisition about Energy

5.3 ETL Processes (Energy Vertical Repository)

End-Point Name	ETL Processes (Energy Vertical Repository)		
End-Point Type	WebService		
Protocol	REST		
Method	GET, POST, DELETE		
Input Data Format	JSON		
Output Data Format	JSON		
CIOP Layer	Knowledge Layer		
Description			
This REST API enables the storage of energy data. It is used in the ETL process to transform real time data into data for the Energy Vertical. Real Time data is aggregated and stored in the Energy Vertical Repository using this service. Energy data is linked to structural data. Data consumption is also enabled using the GET method.			
Services			
URL	http://energyrepo.azurewebsites.net/api/{table}		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
table	Gateways (Other options: DeviceSetups, Locations, Measurements, Units & MeasureTypes)		JSON representation of each element
URL	http://energyrepo.azurewebsites.net/api/{table}		
Method	POST		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
body	"{ ""SerialNumber"": ""c8bedffff7dd4ffdbfffdfff7ffdfefe"", ""ActivationCode"": ""ddd3ba03f1d0db52e6ed62911698e4e01487a738"", ""ApiKey"": ""04605149635c4ba167dc6d2357f9508e4428f07a"", ""Comments"": ""#DemoGateway"", ""HouseholdID"": 1, ""GatewayModelID"": 1, ""FeedID"": 695318714 }"		Status of the new element
table	Gateways (Other options: DeviceSetups, Locations, Measurements, Units & MeasureTypes)		
URL	http://energyrepo.azurewebsites.net/api/{table}/{id}		
Method	Delete		
Input		**Output**	

Parameter	Value	Parameter	Value
table	DeviceSetups (other options: Gateways, Locations, Measurements, Units, MeasureTypes…)		JSON representation of each element
id	2		Status of the deleted element

Table 19 Description of ETL Processes (Energy Vertical Repository)

5.4 API for Energy Services

End-Point Name	API for Energy Services
End-Point Type	API
Protocol	REST
Method	GET, POST
Input Data Format	JSON
Output Data Format	JSON
CIOP Layer	Interoperability Layer
Description	
The API for Energy Services provides the mechanisms for offering the elaborated data in the Knowledge Layer to the Added Value Services and applications related with Energy. All the information related with energy will be uploaded, aggregated and processed within Knowledge Layer and a set of APIs will be created in order to pass all the relevant data to the intelligent applications.	
Services	

Name	Getting Api Key		
URL	http://{domain_name}/activate/{name}.csv		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
name	389b1b08d8a67c4216d8075ad1f15dc8ae3cb6fc		(plain text) {apikey},{id},11,0,1,2,3,4,5,6,7,8,9,10
Name	Last activity		
URL	http://{domain_name}/v2/feeds/{id}/datastreams/10.csv		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
apikey			(text plain) 2017-07-27T12:39:30.000394Z,1720
Name	Send data		

URL	http://{domain_name}/api/{id}.csv		
Method	POST		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
apikey			
body	values of 11 channels CSV		

Table 20 Description of API for Energy Services

5.5 Energy Application

End-Point Name	Energy Application
End-Point Type	Application
Protocol	N/A
Method	N/A
Input Data Format	WMS / WFS
Output Data Format	User Visualization
CIOP Layer	Intelligent Services Layer
Description	
The Energy Efficiency Service can offer a dashboard of summary of the collected data from the households in an aggregated way. This application shows the average, maximum and minimum energy values per building displayed in charts over the time.	
Functionalities	
Functionality	**Expected Output**
Navigation through the 2D Map	The map moves, zooms according to the user interaction
Visualization of map layers	Show / hide selected layer
Visualization of elements information	Show a window containing the alphanumeric information of the selected object
Present report	Show a window containing a report with the selected fields and window size.

Table 21 Description of Energy Application

5.6 ETL Processes (Mobility Repository)

End-Point Name	ETL Processes (Mobility Repository)

End-Point Type	Script
Protocol	bash
Method	N/A
Input Data Format	SQL table
Output Data Format	SQL table
CIOP Layer	Knowledge Layer

Description
This ETL process is in charge of transforming real time data and information coming from mobility devices into data for the Mobility Vertical. Real time data is cleaned, aggregated, processed and stored in the Mobility Vertical Repository using this service. The processed and elaborated data will be provided to the intelligent applications that need mobility information.

Services			
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
ItemID		ItemID	
Timestamp	(selected range) from "date" to "date + period"	Timestamp	"New timestamp"
		Latitude	Single value from list
		Longitude	Single value from list
		Battery	Average value
		Odometer	Average value
		Speed	Average value

Table 22 Description of ETL Processes (Mobility Repository)

5.7 API for Mobility Services

End-Point Name	API for Mobility Services
End-Point Type	API
Protocol	REST or MQTT
Method	POST or Messages
Input Data Format	JSON
Output Data Format	JSON
CIOP Layer	Interoperability Layer
Description	

The API for Mobility Services provides the mechanisms for offering the elaborated data in the Knowledge Layer to the Added Value Services and applications related with mobility. All the information related with mobility will be uploaded, aggregated and processed within Knowledge Layer and a set of APIs will be created in order to pass all the relevant data to the intelligent applications.			
Services			
Name	Sending data (REST or MQTT)		
URL	http://{domain_name}/devices/{deviceId}/messages/events?api-version=2016-02-03		
Method	POST		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
Authorization	"connection string" of the IoT endpoint		
Content-type	application/json		
raw payload	{ "ItemId": 999, "Odometer": 500, "Speed": 35.4, "Battery": 50.2, "Latitude": 44.0, "Longitude": -1.0 }		

Table 23 Description of API for Mobility Services

5.8 API for Citizen Engagement Services

End-Point Name	API for Citizen Engagement Services	
End-Point Type	API	
Protocol	REST	
Method	GET, POST	
Input Data Format	JSON	
Output Data Format	JSON	
CIOP Layer	Interoperability Layer	
Description		
The data related to Citizen Engagement is mainly provided by smartphone apps and inquires fulfilled by citizens for giving their opinion and evaluation of city services. The results of the surveys are directly stored into the historic repository where they will be analysed and processed to obtain the information to be provided to Intelligent applications related to citizen's sentiments and needs.		
Services		
Name	Get all survey	
URL	http://{domain_name}/API/Social/GetAllSurvey	
Method	GET	
Input		**Output**

Parameter	Value	Parameter	Value
		Array of items	{ "id, year, value", "id, year, value", …}
Name	Get question result		
URL	http://{domain_name}/API/Social/GetQuestionResult?QuestionID={QuestionID}&year={year}		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
QuestionID	24	Result	"value"
Year	2017		

Table 24 Description of API for Citizen Engagement Services

5.9 ETL Processes (Historical Repository)

End-Point Name	ETL Processes (Historical Repository)		
End-Point Type	Script		
Protocol	Bash calling a Hadoop file system command (hdfs put)		
Method	N/A		
Input Data Format	Postgre SQL backup		
Output Data Format	No output is produced by the script. Results for the command are registered in log file. Backup is stored in HDFS (new folder identified with timestamp)		
CIOP Layer	Knowledge Layer		
Description			
Script launched by a cron process that extracts real time data from Real Time Repository (Postgre SQL backup) and stores it in Hadoop file system (folder with timestamp name). Internal process not available through a network connection.			
Services			
Name	RTRepoHistorical		
URL	SERVER_LOCALHOST/user/ciop/RTRepo/"$currentts"		
Method	hdfs put (Hadoop file system command)		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
input file (Postgre SQL backup file)	/tmp/"$currentts".sql	None	None

Table 25 Description of ETL Processes (Historical Repository)

5.10 ETL Processes (Structural Repository)

End-Point Name	ETL Processes (Structural Repository)		
End-Point Type	Webservice		
Protocol	REST		
Method	GET		
Input Data Format	JSON		
Output Data Format	JSON		
CIOP Layer	Knowledge Layer		
Description			
This REST API offers information about structural data. It is used in the ETL Process to transform real time data into data for the Energy Vertical. It is used by the Real Time Repository to associate real time data with structural data.			
Services			
URL	http://structuralrepo.azurewebsites.net/api/{table}/{id}		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
table	households (other options: building, district & cities)	JSON representation of each element	{ "$id": "1", "HouseholdID": 27, "Address": "6-A", "ResidentsNumber": 2, "SquareMeters": 30, "BuildingID": null, "Exposure": null }
id (optional)	27		

Table 26 Description of ETL Processes (Structural Repository)

5.11 GIS Repo Services

End-Point Name	GIS Repo Services
End-Point Type	Web Service
Protocol	SOAP
Method	GET, POST, Client Server

Input Data Format	URI, POST
Output Data Format	WFS (GML, GeoJson, .shp, Json, CSV) WMS (gif, geotiff, kml, png, svg, json, text)
CIOP Layer	Interoperability Layer

Description
In this repository it will be kept the information to describe geographically the city area, so it will score the 2D geometry of the common city elements as well as the alphanumerical info associated to them.

Services	
Name	WMS GetCapabilities
URL	http://geoservergis.azurewebsites.net/geoserver/wms?{service}&{version}&{request }
Method	GET

Input		**Output**	
Parameter	**Value**	**Param**	**Value**
SERVICE	WMS		Server Capabilities XML: <WMS_Capabilities xmlns="http://www.opengis.net/wms" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.3.0" updateSequence="348" xsi:schemaLocation="http://www.opengis.net/wms http://geoservergis.azurewebsites.net:80/geoserver/schemas/ wms/1.3.0/capabilities_1_3_0.xsd"> <Service> <Name>WMS</Name> <Title>GeoServer Web Map Service</Title> [...]
VERSION	2.0.0		
REQUEST	GetCapabilities		

Name	WMS GetMap
URL	http://geoservergis.azurewebsites.net/geoserver/wms?{service}&{version}&{request}&{layers}&{styles}&{ crs}&{bbox}&{width}&{height}&{format}
Method	GET

Input		**Output**	
Parameter	**Value**	**Param**	**Value**
SERVICE	WMS		Map Image:
VERSION	2.0.0		
REQUEST	GetMap		
layers	smartencity:bench		
styles	point		
crs	EPSG:25830		
bbox	520527.38458945166,47416 47.821567042,529572.1722 914433,4747224.157644480 5		

width	100		
height	100		
format	image/png		

Name	WMS GetFeatureInfo
URL	http://geoservergis.azurewebsites.net/geoserver/wms?{service}&{version}&{request}&{layers}&{styles}&{crs}&{bbox}&{width}&{height}&{query_layers}&{x}&{y}
Method	GET

Input		**Output**	
Parameter	**Value**	**Param**	**Value**

SERVICE	WMS		Results for FeatureType 'smartencity:bench':
			-- 	
			the_geom = [GEOMETRY (Point) with 1 points]
			CODIGO = {77A48135-E708-4E6B-9DAD-A54108BA8CD2}
			ZONA = 8
			SITUACION = IGNACIO RUIZ DE LUZURIAGA
			MODELO = B005
			FEC_ULT_IN = null
			ESTAD_CONS = BUENO
			DESPERFECT = NO
			DESCR_DESP =
			PINTADAS = NO
			APTITUD = APTA
			OBSERVACIO = MADERA
			CARTO_CAMP = CAMPO
			FEC_BAJA = null
			BAJA = NO
			-- 	

VERSION	2.0.0		
REQUEST	GetFeatureInfo		
layers	smartencity:bench		
styles	point		
crs	EPSG:25830		
bbox	520527.38458945166,4741647.821567042,529572.1722914433,4747224.1576444805		
width	200		
height	200		
query_layers	smartencity:bench		
x	100		
y	100		

Table 27 Description of GIS Repo Services

5.12 GIS Structural Repo Services

End-Point Name	GIS Structural Repo Services
End-Point Type	Web Service
Protocol	SOAP
Method	POST
Input Data Format	XML
Output Data Format	XML
CIOP Layer	Interoperability Layer
Description	
Allows the retrieval of the data contained in the structural repo. The access is done in a standard way using Web Feature Service (WFS), which is defined by the Open Geospatial Consortium. Building, Thematic Surfaces and Geometric Surfaces can be accessed with this service.	

Services			
Name	GIS Structural Repo – Web Feature Service		
URL	http://3dcity.tecnalia.com:80/ServiciosWeb/services/smartengasteizv1		
Method	POST		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
XML Body	WFS Simple - GetFeature operation	XML	The information of the requested buildings, thematic surfaces or surface geometries

Table 28 Description of GIS Structural Repo Services

5.13 API for KPIs

End-Point Name	API for KPIs
End-Point Type	API
Protocol	REST
Method	GET
Input Data Format	JSON
Output Data Format	JSON
CIOP Layer	Interoperability Layer

Description

Shows KPI data related to the different domains of the project.

- Energy, includes energy consumption and production information.
- Mobility, like last location, route, total kw recharged in EV stations, etc.
- Citizen Involvement, like engagement degree, number of logins, number of downloads etc.
- ICT, like number of sensing systems installed, mobility equipment connected, percentage of building connected, etc.

Data can be either calculated on request of pre-calculated by automated services "KPI Calculation Services".

Services			
Name	Get electricity consumption		
URL	http://{domain_name}/api/kpi/GetElectricityConsumption		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**

		number	Total kw consumed
Address of the building	string	number	Kw consumed in the requested building
Services			
Name	Get electricity production		
URL	http://{domain_name}/api/kpi/GetElectricityProduction		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
		number	Total kw produced
Address of the building	string	number	Kw produced in the requested building
Name	Get last position		
URL	http://{domain_name}/api/kpi/GetLastPosition		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
Device id	number	JSON	Location id and timestamp
Name	Get route		
URL	http://{domain_name}/api/kpi/GetRoute		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
Device id	number	JSON	Array of locations ids and timestamps
from	timestamp		
to	timestamp		
Name	Get charged kw from charging stations		
URL	http://{domain_name}/api/kpi/GetStationsCharged		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
		number	Total kw charged in charging stations

id	number	number	Kw charged in requested charging station
Name	Get visitors		
URL	http://{domain_name}/api/kpi/GetVisitors		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
period		number	All visitors
period	day/week/mon/3mon/6mon/year	number	Visitors per requested period
Name	Get engagement degree		
URL	http://{domain_name}/api/kpi/GetEngagementDegree		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
		percent	Engagement degree (calculated value)
Name	Count number of devices installed		
URL	http://{domain_name}/api/kpi/CountDevices		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**

		JSON	{
			"electricity": 11,
			"gps": 11,
			"solar": 11,
			"water": 11,
			}

type	electricity/gps/solar/water	number	Number of devices per requested type

Table 29 Description of API for KPIs

5.14 Services for Integrating Open Data

End-Point Name	Services for integrating Open Data information
End-Point Type	Webservice
Protocol	REST / SOAP
Method	GET

Input Data Format	TXT / CSV		
Output Data Format	CSV / XML		
CIOP Layer	Acquisition / Interconnection Layer		
Description			
Fetches Open Data from 3rd party applications like forecast of electricity prices from NordPool, forecast of CO2 emission levels from Energinet.dk. Implementation and deployment is done inside devices itself – Acquisition / Interconnection Layer – devices send requests at specific interval to 3rd party applications to get data. However, this implementation and deployment can be done in Interoperability layer, depending on where data aggregation will take place.			
Services			
Name	Get electricity prices		
URL	http://{domain_name}/api/OpenData/GetElectricityPrices		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
		JSON	Array of prices for next day
Name	Get CO2 emission levels		
URL	http://{domain_name}/api/OpenData/GetCO2Emissions		
Method	GET		
Input		**Output**	
Parameter	**Value**	**Parameter**	**Value**
		JSON	Array of CO2 emission levels for next day

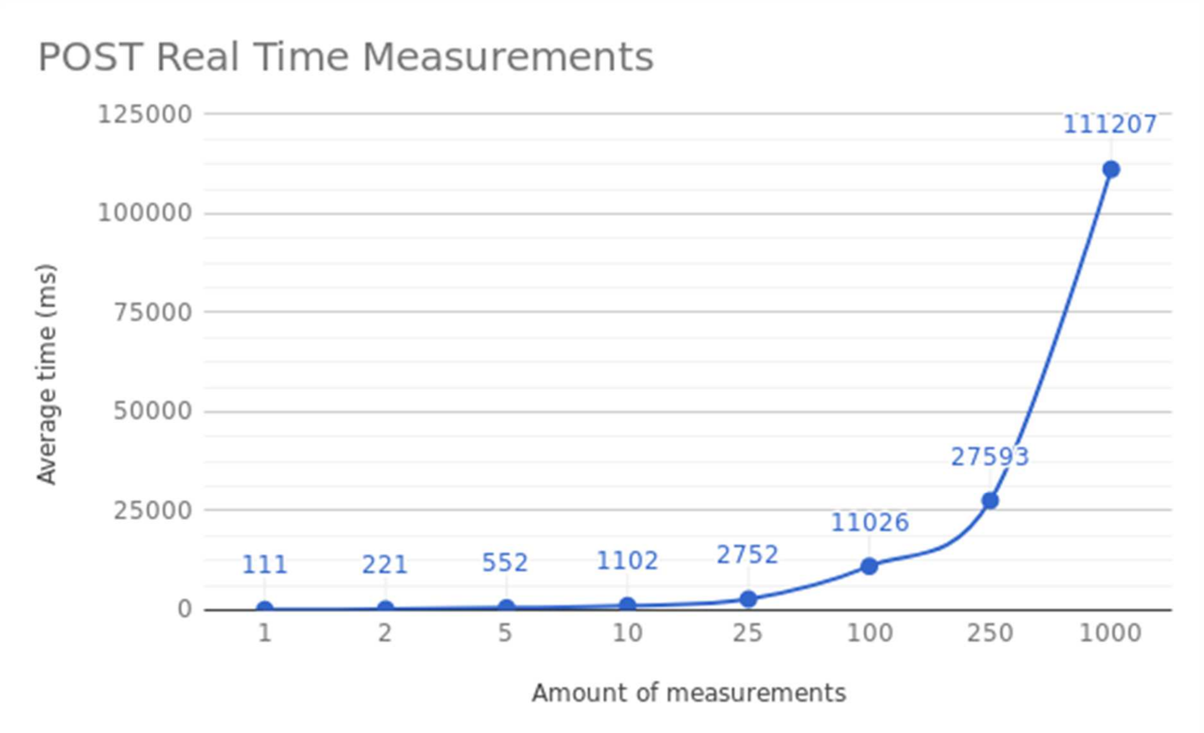
Table 30 Description of Services for Integrating Open Data

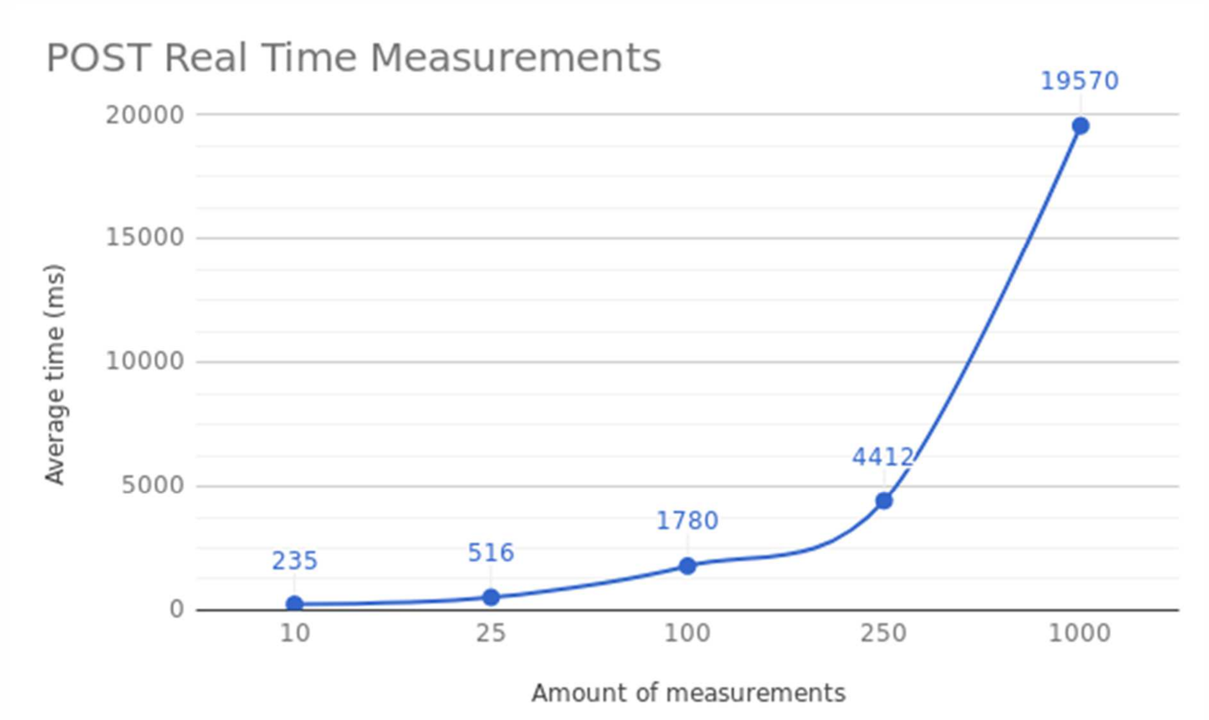
6 Deployment of tests

6.1 Unit Tests

6.1.1 Data Acquisition about Energy

6.1.1.1 Performance Test

Module Name	Data Acquisition about Energy
Test Type	**Response Time**
Testing Tool	Own developed tool (serial)
Acceptance Criteria	The average of the response time must be below 250 milliseconds pe measurement in order to be considered valid.
Implementation	
This test writes simulated sensor data and stores it in the real time repository using REST API. This request has been performed 20 times and average response time has been calculated.	
Result	
Result in millisecond: 110 ms/measurement  The threshold for the response time is set in 250 milliseconds. So, the response time test has been successfully passed.	
Test Type	**Concurrency Load Tests**
Testing Tool	Own developed tool (parallel)
Acceptance Criteria	Response time must be below 30 seconds when 1000 sensors writing at the

	same time.
Implementation	
This test writes simulated sensor data and stores it in the real time repository using REST API. Requests are carried out simultaneously by several connected sensors. In order to do that, a request has been performed by different number of users: 10, 25, 100, 250 and 1000.	
Result	
As can be seen in the following figure, the bigger is the number of sensors, the more time is needed in writing the data when performing the same request. However, for storing 1000 sensors' measurements it still continues below the response time threshold (30 seconds). 	

6.1.1.2 *Functional Test*

Module Name	Data Acquisition about Energy
Test Type	**Functional Test**
Testing Tool	cURL

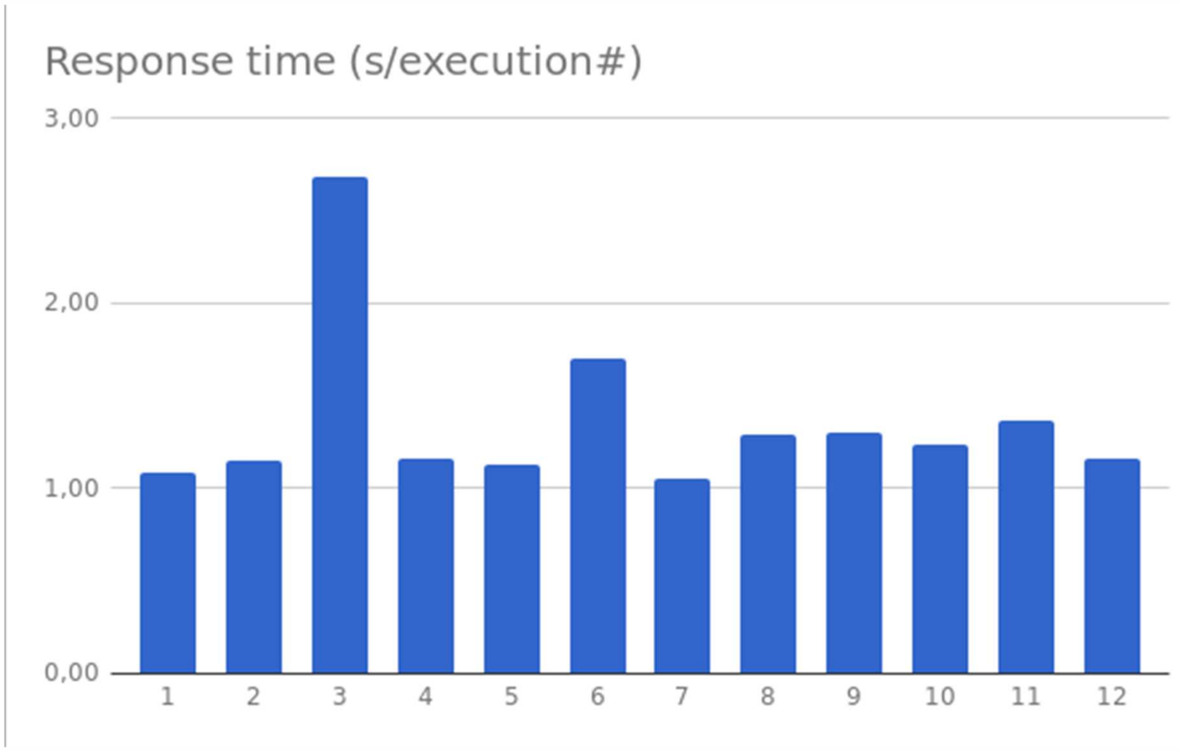
#	Functionality	Input	Output	Result (Ok/ NOk)
1	Activate a sensor	export GATEWAY_SERIAL="c8bedffff7dd4ffdbfffdfff7ffdfefe" export GATEWAY_ACTIVATION="ddd3ba03f1d0db52e6ed62911698e4e01487a738" export GATEWAY_FEEDID="695318714" export GATEWAY_APIKEY="04605149635c4ba167dc6d2357f	04605149635c4ba167dc6d2357f9508e4428f07a, 695318714,11,0,1,2,3,4, 5,6,7,8,9,10	Ok

		9508e4428f07a" export IP_REPO_REALTIME="217.76.242.53" export SENSOR_DATA="30.50,480,243,55,156.7,0,0,99,0,0,0" curl --http1.1 -H "Host: provisioning.connectedenvironments.com" --url "http://${IP_REPO_REALTIME}/activate/${GATEWAY_ACTIVATION}.csv"		
2	Simulate a sensor's provisioning	curl --http1.1 -H "api.pachube.com" -H "X-PachubeApiKey: ${GATEWAY_APIKEY}" --url "http://${IP_REPO_REALTIME}/v2/feeds/${GATEWAY_FEEDID}/datastreams/10.csv"	2017-12-18T10:42:22+00:00,0 (Current Date)	Ok
3	Simulate a sensor measurement	curl --http1.1 -X PUT -H "www.pachube.com" -H "X-PachubeApiKey: ${GATEWAY_APIKEY}" -H "Content-Type: text/csv" --url "http://${IP_REPO_REALTIME}/api/${GATEWAY_FEEDID}.csv" --data-raw "${SENSOR_DATA}"	Ok	Ok

6.1.2 ETL Processes (Energy Vertical Repository)

6.1.2.1 Performance Test

Module Name	ETL Processes (Energy Vertical Repository)
Test Type	**Response Time**
Testing Tool	Own developed tool (Script logs)
Acceptance Criteria	The average of the response time must be below 3 seconds for the whole process in order to be considered valid
Implementation	
The script reads from database, reads the list of DeviceSetups and Gateways from the Vertical repository and calls a REST API to store the aggregated measurement. The script contemplates writing timestamps for the different stages. This test writes the time consumed in each task. This request has been performed 12 times, and average response time for the whole process has been calculated.	
Result	
Result in millisecond: 1360 ms/script execution	



The threshold for the response time is set in 3 seconds. So, the response time test has been successfully passed.

Test Type	**Data volume Load Tests**
Testing Tool	Own developed tool (Python script with requests library)
Acceptance Criteria	Response time must be below 500 milliseconds per measurements wher storing 1000 measurements in a row.

Implementation

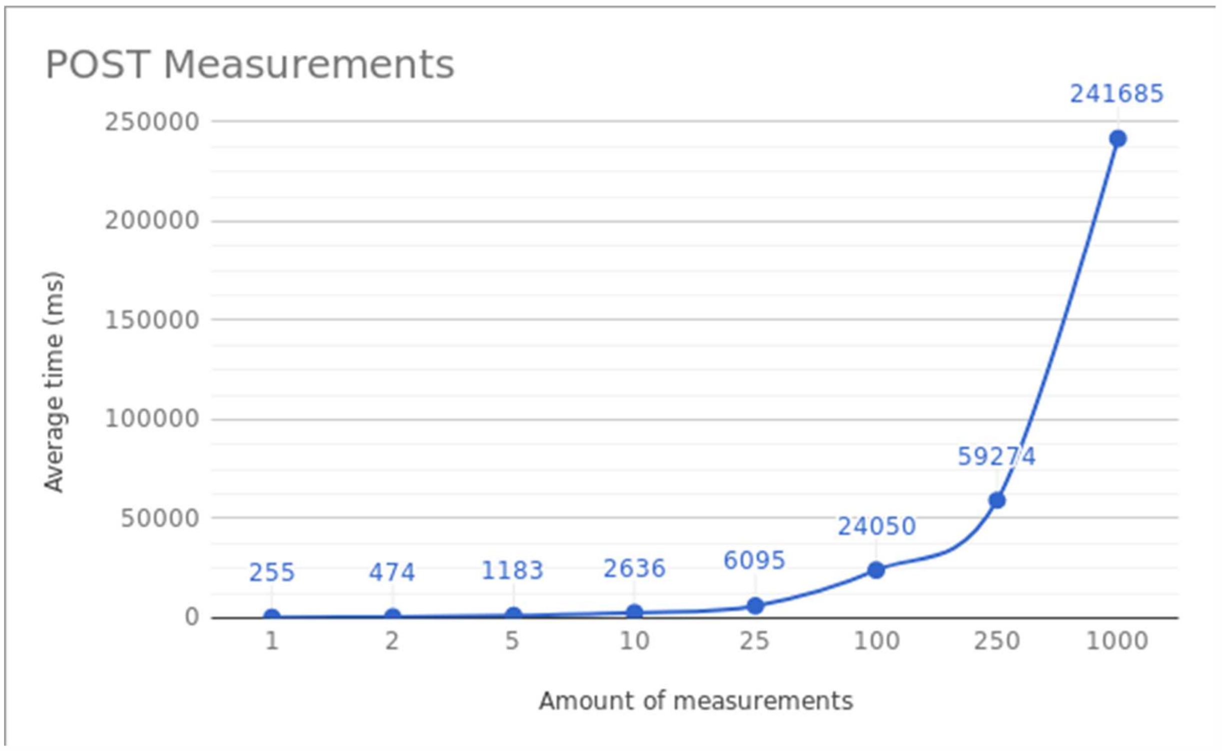
This test simulates large storage of data using the REST API provided by the Energy Vertical repository. A regular script sends 2 measurements per dwelling every hour. The simulation sends up to 1000 measurements (500 dwellings) at one time.

Each request has been performed 25 times and mean response time has been calculated.

Result

As can be seen in the following figure, the bigger is the volume of data, the more time is needed in store the data. However, the amount of time spent for storing each measurement is still below 500ms. Therefore, the test shows that 4min will be spend each hour to store 500 dwellings' data:

- Average measurement storage time: 241ms.



6.1.2.2 *Functional Test*

Module Name	ETL Processes (Energy Vertical Repository)
Test Type	**Functional Test**
Testing Tool	Rester (Firefox add-on)

#	Functionality	Input	Output	Result (Ok/ NOk)
1	Request all measurement information	http://energyrepo.azurewebsites.net/api/measurements	[{ "$id": "1", "MeasurementID": 3, "Value": 21, "EstimatedCost": 0.865, "StartDate": "2017-06-09T12:20:28.22", "EndDate": "2017-06-09T12:20:28.22", "StartDateTimestamp": 1497005093, "EndDateTimestamp": 1497005393, "DeviceSetupID": 1 }, …]	Ok
2	Request specific	http://energyrepo.azurewebsites.net/api/DeviceSetups/1	{ "$id": "1", "DeviceSetupID": 1,	Ok

	Device Setup information		"SetupDate": null, "MeasureTypeID": 1, "UnitID": 1, "ConstantConversionValue": 1, "LocationID": 1, "ManufacturingYear": 2017, "GatewayID": 1, "ChannelNumber": 0 }	

3	Request specific Gateway information	http://energyrepo.azurewebsites.net/api/Gateways	{
			"$id": "2",
			"GatewayID": 2,
			"SerialNumber": "c8bedffff7dd4ffdbfffdfff7ffdfefe",
			"ActivationCode": "ddd3ba03f1d0db52e6ed62911698e4e01487a738",
			"ApiKey": "04605149635c4ba167dc6d2357f9508e4428f07a",
			"Comments": "#DemoGateway",
			"HouseholdID": 33,
			"GatewayModelID": 1,
			"FeedID": 695318714
			}
				Ok

6.1.3 API for Energy Services

6.1.3.1 Performance Test

Module Name	API Energy Services
Test Type	**Response Time**
Testing Tool	Own developed tool
Acceptance Criteria	The average of the response time must be below 2 seconds in order to be considered valid
Implementation	
This test evaluates the performance of the response time when an application asks for some data to API Energy Services. The test consists in 100 times repetition of calling the API Energy Services. Each time, the time spent since the calling of the API and the reception of the data will be gathered.	
Result	
Result in milliseconds: 1640 The threshold for the response time is set in 2 seconds. So, the response time test has been successfully passed.	
Test Type	**Data volume Load Tests**
Testing Tool	Own developed tool
Acceptance Criteria	Response time must be below 3 seconds when requesting less than 200K o energy data

Implementation

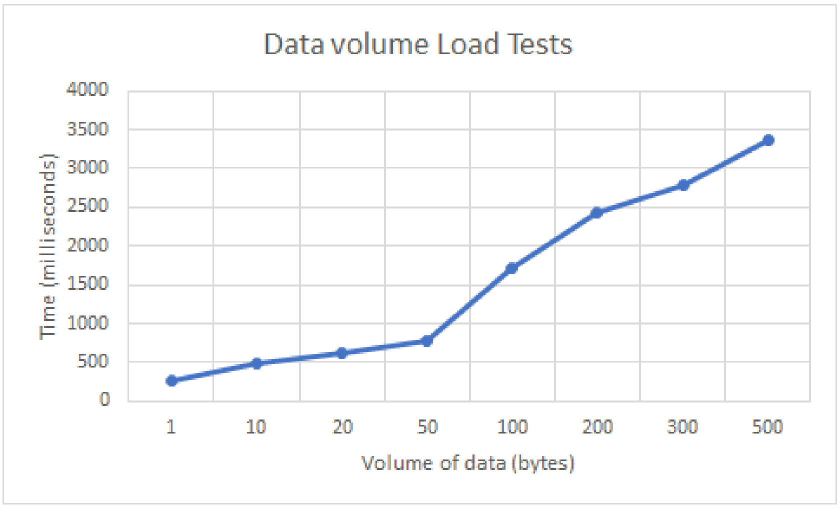
This test should allow comparison of response times of transactions that require different volumes of data. The response time will be different depending on the quantity of data to be processed and sent by the different functions of API Energy Services.

A program that implements the request process of different amount of data has been developed. The amount of data requested will be variable among small data (less than 100 bytes) to more heavy data (up to 200K).

Each request has been performed 100 times and the response time has been calculated.

Result

The following figure presents the results where a request for small quantity of data lasts around 100 ms. and an amount of data bigger than 200K lasts more than 3 seconds.



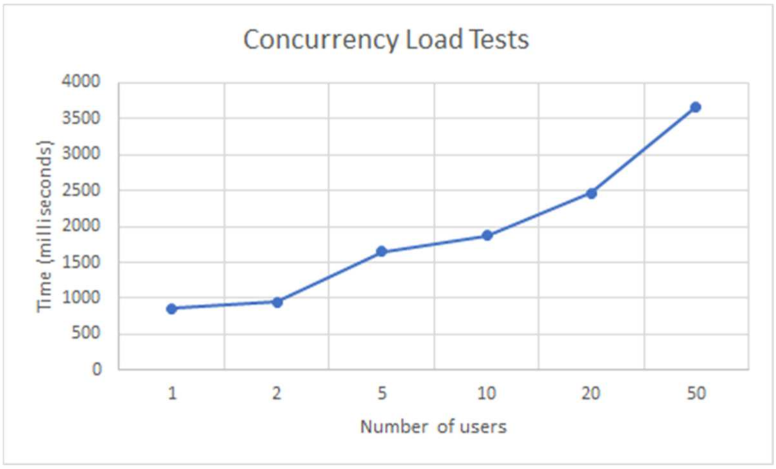
Test Type	**Concurrency Load Tests**
Testing Tool	Own developed tool
Acceptance Criteria	Response time must be below 3 seconds when 20 users requesting data for a period of 6 months at the same time.

Implementation

The test must allow identifying the average response time when requesting data for a period of 6 months. Requests are carried out simultaneously by several connected users. In order to do that, a request has been performed by different number of users: 1, 2, 5, 10, 20 and 50.

Result

As can be seen in the following figure, the bigger is the number of users, the more time is needed in retrieving the data when performing the same request. However, for requesting the desired amount of data for 20 concurrent users it still continues below the response time threshold (3 seconds).



6.1.3.2 Functional Test

Module Name	API for Energy Services			
Test Type	**Functional Test**			
Testing Tool	Own developed tool			
#	**Functionality**	**Input**	**Output**	**Result (Ok/ NOk)**
1	Request of energy consumption of a building last month	http://domain_name/GetBuildingConsumption?BuildingID=6&DateStart="2017-6-1"&DateEnd="2017-6-30"	{ "BuildingID": 6, "TotalConsumption": "16.84", "Unit": "MW" }	Ok
2	Request of energy consumption of a building for the last six months	http://domain_name/GetBuildingConsumption?BuildingID=6&DateStart="2017-1-1"&DateEnd="2017-6-30"	{ "BuildingID": 4, "TotalConsumption": "125.684", "Unit": "MW" }	Ok
3	Request of energy consumption of a building for the last year	http://domain_name/GetBuildingConsumption?BuildingID=6&DateStart="2016-1-1"&DateEnd="2016-12-31"	{ "BuildingID": 19, "TotalConsumption": "235.36", "Unit": "MW" }	Ok
4	Request of energy consumption of the buildings of a neighborhood (less than 5 buildings) for last month	http://domain_name/GetNeighborhoodConsumption?NeighborhoodID=3&DateStart="2017-6-1"&DateEnd="2017-6-30"	[{ "NeighborhoodID": 3, "Buildings": { "BuildingID": 6, "TotalConsumption": "16.84", "Unit": "MW" }, { "NeighborhoodID": 3 "BuildingID": 11, "TotalConsumption": "14.68", "Unit": "MW" } }]	Ok
5	Request of energy consumption of the buildings of a neighborhood (less than 5 buildings) for last six months	http://domain_name/GetNeighborhoodConsumption?NeighborhoodID=3&DateStart="2017-1-1"&DateEnd="2017-6-30"	[{ "NeighborhoodID": 3, "Buildings": { "BuildingID": 6, "TotalConsumption": "84.37", "Unit": "MW" }, { "NeighborhoodID": 3 "BuildingID": 11, "TotalConsumption": "69.44", "Unit": "MW" } }]	Ok
6	Request of energy consumption of the buildings of a neighborhood (less than 5 buildings) for last year	http://domain_name/GetNeighborhoodConsumption?NeighborhoodID=3&DateStart="2016-1-1"&DateEnd="2016-12-31"	[{ "NeighborhoodID": 3, "Buildings": { "BuildingID": 6, "TotalConsumption": "200.47", "Unit": "MW" }, { "NeighborhoodID": 3	Ok

			"BuildingID": 11, "TotalConsumption": "173.58", "Unit": "MW" } }]	

6.1.4 Energy Application

Module Name	Application ENE
Test Type	**Functional Test**
Testing Tool	User Interaction
Test Case Title	Navigation through the 2D Map

Test Case Description

Test to check the possibility of navigating on the map. Different ways of navigation are going to be tested: panning the map, zoom in/out

Steps

#	Action	Expected Output	Output	Passes (Y/N)
1.	Pan the map: User click with the left mouse button and drag the map in the desired direction: up, down, right and left	The map moves according to user's direction	The map moves according to user's direction	Y
2.	Zoom map at cursor location: User double click on the point of interest with the left mouse button. Roll the mouse wheel forward to scale the map to the cursor location or roll the mouse wheel back to reduce map scale to the cursor location	The map zooms in according to de user's click	The map zooms in according to de user's click	Y
3.	Map Zoom in/Zoom out: Click on the PLUS (+) or MINUS (-) button to zoom in/out on the map	The map zoom changes	The map zoom changes	Y
4.	Initial zoom according to the study area: Open the viewer	The initial map is shown with the initial zoom covering the study area	The initial map is shown with the initial zoom covering the study area	Y

Summary & Comments

The application passes the tests for Map Navigation

Test Case Title	Visualization of Map Layers

Test Case Description

Test to check the possibility of interacting with the map layers.

Steps

#	Action	Expected Output	Output	Passes (Y/N)
1.	Show map layers menu: Place the cursor over the map layers menu icon	The list of available layers for visualization is displayed	The list of available layers for visualization is displayed	Y
2.	Show/hide layers: User clicks on the checkbox beside the layer he/she wants to show or hide	The selected layers on the layers menu should act in consequence: Show or hide	The selected layers on the layers menu should act in consequence: Show or hide	Y
3.	Show default basemap layer: Open the viewer	It provides a basemap as a background by default.	It provides a basemap as a background by default.	Y

Summary & Comments

The application passes the tests for Map Layers.

Test Case Title	Visualization of elements information

Test Case Description

Test to check the possibility to Identify and consult the alphanumeric information of the layers loaded on the map.

Steps

#	Action	Expected Output	Output	Passes (Y/N)
1.	Identify: In order to identify an object displayed, place the cursor on the map and click the left mouse button at the point where we want the information to be consulted	A window containing the alphanumeric information of the geographic object is displayed.	A window containing the alphanumeric information of the geographic object is displayed.	Y

Summary & Comments

The application passes the tests for Visualization of elements information

Test Case Title	Present Report

Test Case Description

Test to check the possibility of interacting with the reports

Steps

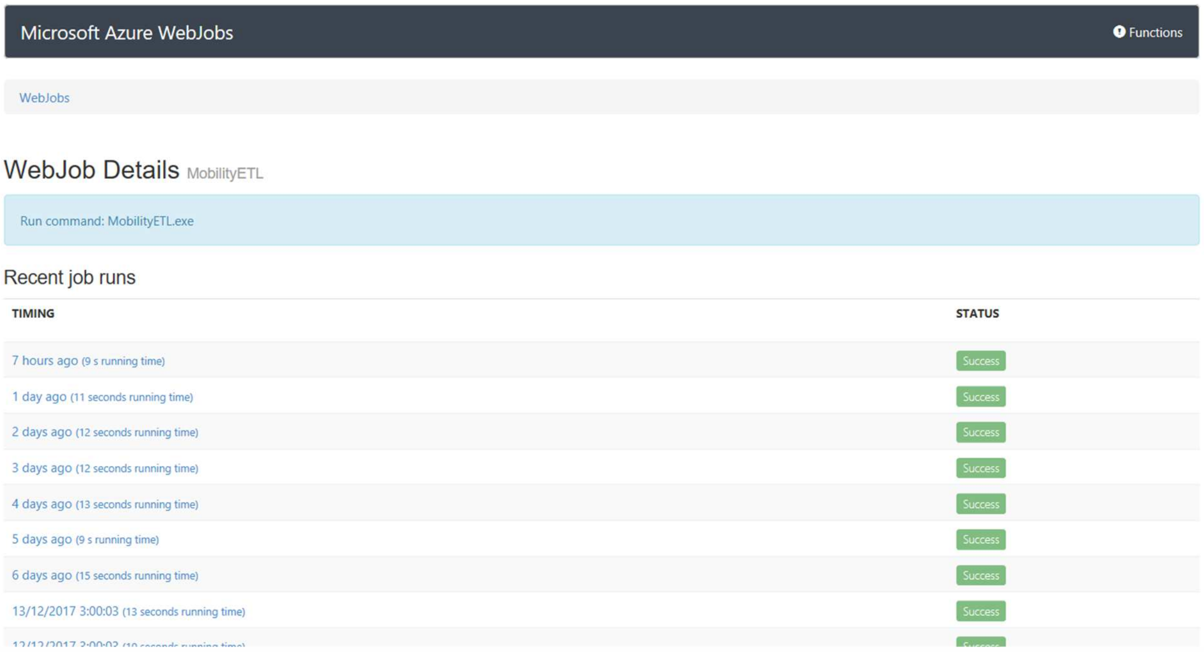
#	Action	Expected Output	Output	Passes (Y/N)
1.	Display reports of KPIs: Open the viewer	The report window appears on the bottom right-hand corner of the viewer	The report window appears on the bottom right-hand corner of the viewer	Y
2.	Display data according to	View the data according	View the data	Y

	selected fields: Select different fields in the report like address or date	to the selected fields	according to the selected fields	
3.	Access the full screen mode: Click on the icon for full screen	The reports opens in full screen mode	The reports opens in full screen mode	Y
4.	Display in a larger size each part of the report: Click on the icon for larger size	The report enlarges the element selected to enlarge.	The report enlarges the element selected to enlarge.	Y
5.	Share the URL: Click on the icon for Share URL	It generates a URL for sharing the report	It generates a URL for sharing the report	Y
Summary & Comments				
The application passes the tests for reports				

6.1.5 ETL Processes (Mobility Repository)

6.1.5.1 Performance Test

Module Name	ETL Processes (Mobility Repository)
Test Type	**Response Time**
Testing Tool	Own developed tool (Azure logs)
Acceptance Criteria	Not receiving timeout messages and calculate average values without exceeding 30 seconds for each mobility object in order to be considered valid
Implementation	
This test runs the established CRON to update and feed a Vertical repo, gathering information from real-time repo (mobility section). This request has been performed several times not only to measure the response time but also to test the reliability of the CRON instance.	
Result	



Microsoft Azure WebJobs — Functions

WebJobs

WebJob Details MobilityETL

Run command: MobilityETL.exe

Recent job runs

TIMING	STATUS
7 hours ago (9 s running time)	Success
1 day ago (11 seconds running time)	Success
2 days ago (12 seconds running time)	Success
3 days ago (12 seconds running time)	Success
4 days ago (13 seconds running time)	Success
5 days ago (9 s running time)	Success
6 days ago (15 seconds running time)	Success
13/12/2017 3:00:03 (13 seconds running time)	Success

The previous figure presents the executions made through the previous days and each line represents an ETL process with the information of “when”, “how long” and “how went”.

Actually, there is no real data, nor devices inside the platform, just a couple simulated devices sending telemetry information. Nonetheless, the acceptance criteria are accomplished because the whole process (having two devices) needs less than 30 seconds to complete the execution. So, the response time test has been successfully passed.

6.1.5.2 Functional Test

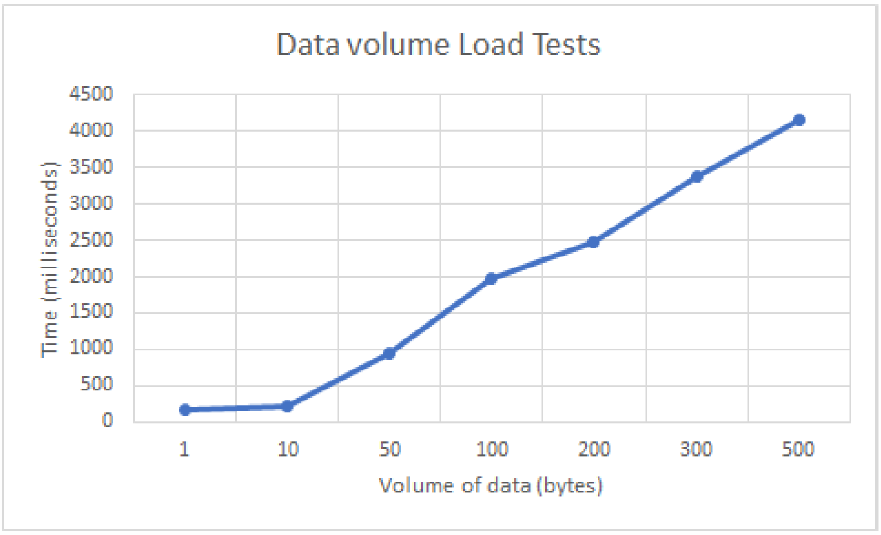
Module Name	ETL Processes (Vertical Repository)
Test Type	**Functional Test**
Testing Tool	Visual Studio and SQL Server Manager

#	Functionality	Input	Output	Result (Ok/ NOk)
4	Request all telemetry information	SELECT * FROM [Measurement_table] WHERE {Date} = ‘selected_date’ AND {ItemID} = ‘identifier’	“Table of Measurement objects”	Ok
5	Update value	UPDATE ‘Item_table’ set {Timestamp} = ‘DateTime.Now’, {Latitude} = List.Latitude.Last(), {Longitude} = List.Longitude.Last(), {Batery} = Avg(List.Battery), {Odometer} = Avg(List.Odometer), {Speed} = Avg(List.Speed)	1 row(s) affected)	Ok

6.1.6 API for Mobility Services

6.1.6.1 Performance Test

Module Name	API Mobility Services
Test Type	**Response Time**
Testing Tool	Own developed tool
Acceptance Criteria	The average of the response time must be below 2 seconds in order to be considered valid
Implementation	
This test evaluates the performance of the response time when a mobility application asks for some data to API Mobility Services. The test consists in 100 times repetition of calling the API Mobility Services. Each time, the time spent since the calling of the API and the reception of the data will be gathered.	
Result	
Result in milliseconds: 1206 The threshold for the response time is set in 2 seconds. So, the response time test has been successfully passed.	
Test Type	**Data volume Load Tests**
Testing Tool	Own developed tool
Acceptance Criteria	Response time must be below 4 seconds when requesting less than 300K of mobility data
Implementation	
This test should allow comparison of response times of transactions that require different volumes of data. The response time will be different depending on the quantity of data to be processed and sent by the different functions of API Mobility Services. A program that implements the request process of different amount of data has been developed. The amount of data requested will be variable among small data (less than 100 bytes) to more heavy data (up to 300K). Each request has been performed 100 times and the response time has been calculated.	
Result	
The following figure presents the results where a request for small quantity of data lasts around 100 ms and an amount of data bigger than 300K lasts less than 4 sec.	



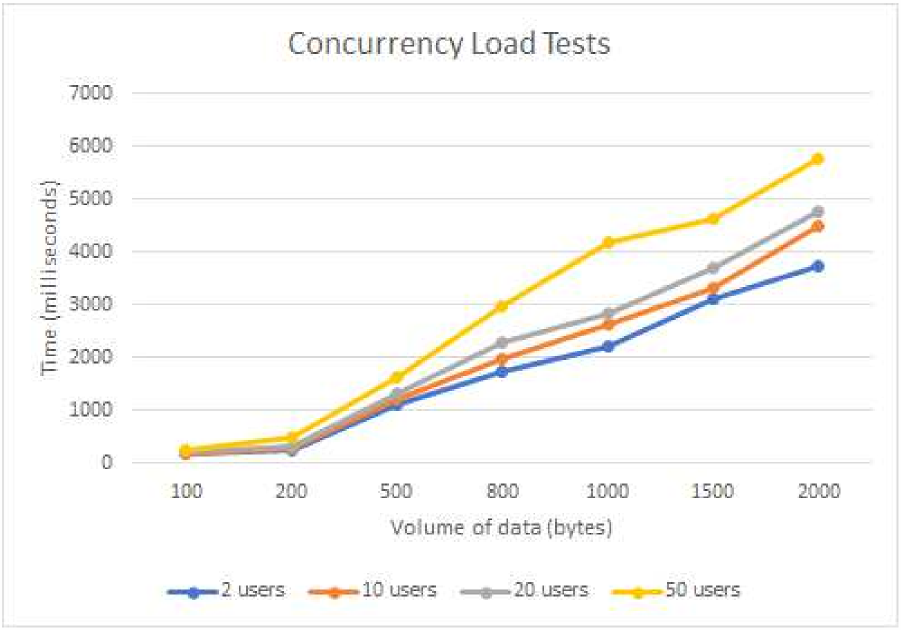
Test Type	**Concurrency Load Tests**
Testing Tool	Own developed tool
Acceptance Criteria	Response time must be below 5 seconds when 20 users requesting 1500K bytes of data at the same time.

Implementation

The test must allow identifying the average response time when requesting 1500K bytes. Requests are carried out simultaneously by several connected users. In order to do that, a request has been performed by different number of users: 2, 10, 20 and 50.

Result

As can be seen in the following figure, the bigger is the number of users, the more time is needed in retrieving the data when performing the same request. However, requesting data for 50 concurrent users it still continues below the response time threshold (5 seconds for 1500K bytes of data).



6.1.6.2 Functional Test

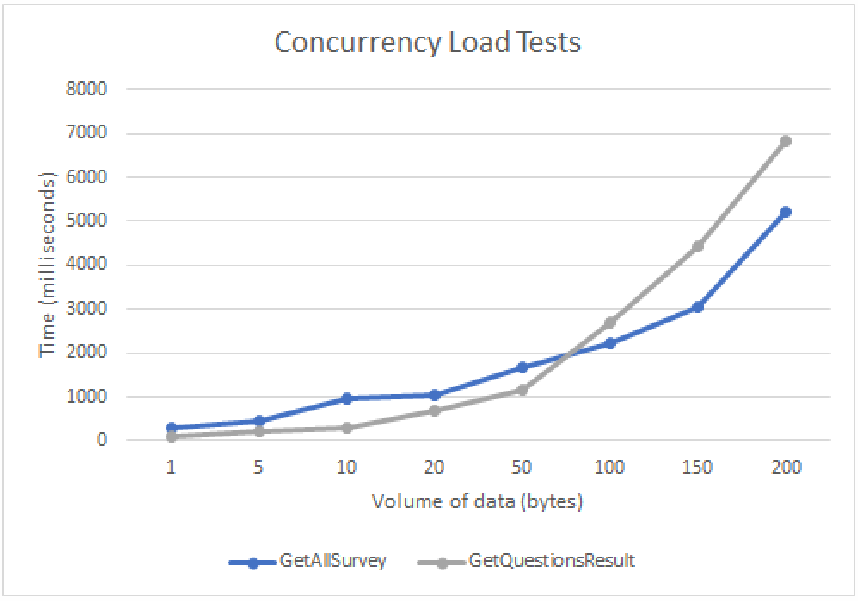
Module Name	API for Mobility Services
Test Type	**Functional Test**
Testing Tool	Own developed tool

#	Functionality	Input	Output	Result (Ok/ NOk)
1	Request of the route of a Mobility device (bicycle, car, bike) during the current day	http://domain_name/mobility/GetRoute?VehicleID=4&Date="2017-10-16"	{ "RouteID": 1, "RouteName": "sample string 2", "RouteCreated": "2017-10-16", "VehicleID": 4, "Locations": [{ "LocationID": 1, "Name": "sample string 2", "Latitude": 3.1, "Longitude": 4.1, "RouteId": 5 }, { "LocationID": 1, "Name": "sample string 2", "Latitude": 3.1, "Longitude": 4.1 }] }	Ok
2	Request of the battery consumption and other data of a Mobility device (bicycle, car, bike) during the current day	http://domain_name/mobility/GetTelemetryAverage?VehicleID=5&DateStart="2017-10-7"&DateEnd="2017-10-7"	{ "VehicleID": 5, "Odometer": 1.1, "Speed": 1.1, "Battery": 1.1, "Latitude": 1.1, "Longitude": 1.1 }	Ok
3	Request of the routes of several Mobility devices (bicycle, car, bike) (less than 5 devices) during the current day	http://domain_name/mobility/GetRoute?Vehicle=4+5&Date="2017-10-16"	[{ "RouteID": 7, "RouteName": "sample string 2", "RouteCreated": "2017-10-16", "VehicleID": 4, "Locations": [{ "LocationID": 1, "Name": "sample string 2", "Latitude": 3.1, "Longitude": 4.1, "RouteId": 5 }, { "LocationID": 1, "Name": "sample string 2", "Latitude": 3.1, "Longitude": 4.1 }] }, { "RouteID": 6, "RouteName": "sample string 2", "RouteCreated": "2017-10-16", "VehicleID": 5, "Locations": [{ "LocationID": 1, "Name": "sample string 2", "Latitude": 3.1, "Longitude": 4.1, "RouteId": 5 }, { "LocationID": 1, "Name": "sample string 2", "Latitude": 3.1, "Longitude": 4.1 }] }]	Ok
4	Request of the battery consumptions and other data of several Mobility devices (bicycle, car, bike) (less than 5 devices) during the current day	http://domain_name/mobility/GetTelemetryAverage?Vehicle=9+17&DateStat="2017-8-8"&DateEnd="2017-8-8"	[{ "VehicleID": 9, "Odometer": 1.1, "Speed": 1.1, "Battery": 1.1, "Latitude": 1.1, "Longitude": 1.1 }, { "VehicleID": 17, "Date": "2017-8-8", "Odometer": 1.1, "Speed": 1.1, "Battery": 1.1, "Latitude": 1.1, "Longitude": 1.1 }]	Ok
5	Request of the battery consumption and other data of a Mobility device (bicycle, car, bike) during for last six months	http://domain_name/mobility/GetTelemetryAverage?Vehicle=7&DateStat="2017-5-1"&DateEnd="2017-11-30"	{ "VehicleID": 7, "Odometer": 1.1, "Speed": 1.1, "Battery": 1.1, "Latitude": 1.1, "Longitude": 1.1 }	Ok
6	Request of the battery consumption and other data of a Mobility device (bicycle, car, bike) during for last year	http://domain_name/mobility/GetTelemetryAverage?Vehicle=21&DateStat="2016-1-1"&DateEnd="2016-12-31"	{ "VehicleID": 21, "Odometer": 1.1, "Speed": 1.1, "Battery": 1.1, "Latitude": 1.1, "Longitude": 1.1 }	Ok

6.1.7 API for Citizen Engagement Services

6.1.7.1 Performance Test

Module Name	API Citizen Engagement Services
Test Type	**Response Time**
Testing Tool	Own developed tool
Acceptance Criteria	The average of the response time must be below specified time in order to be considered valid
Implementation	
This test evaluates the performance of the response time when an application asks for some data to API Citizen Engagement Services. The test consists in 100 times repetition of calling the API Citizen Engagement Services. This test has been launched on two services to get a mean response time: GetAllSurvey and GetQuestionsResult.	
Result	
GetAllSurvey request (desired response time, 3 seconds) Average time in milliseconds: 1638 GetQuestionsResult request (desired response time, 5 seconds) Average time in milliseconds: 3968 Each service has different threshold response time. In any event, the response time test has been successfully passed on both.	
Test Type	**Data volume Load Tests**
Testing Tool	Own developed tool
Acceptance Criteria	Response time must be below 3 seconds when requesting less than 100K o citizen's data
Implementation	
This test should allow comparison of response times of transactions that require different volumes of data. The response time will be different depending on the quantity of data to be processed and sent by the different functions of Citizen Engagement Services. A program that implements the request process of different amount of data has been developed. The amount of data requested will be variable among small data (less than 100 bytes) to more heavy data (up to 100K). Each request has been performed 100 times and the average response time has been calculated.	
Result	
The following figure presents the results of each function requesting an amount of 100K of data. On both cases, the test is accomplished successfully.	



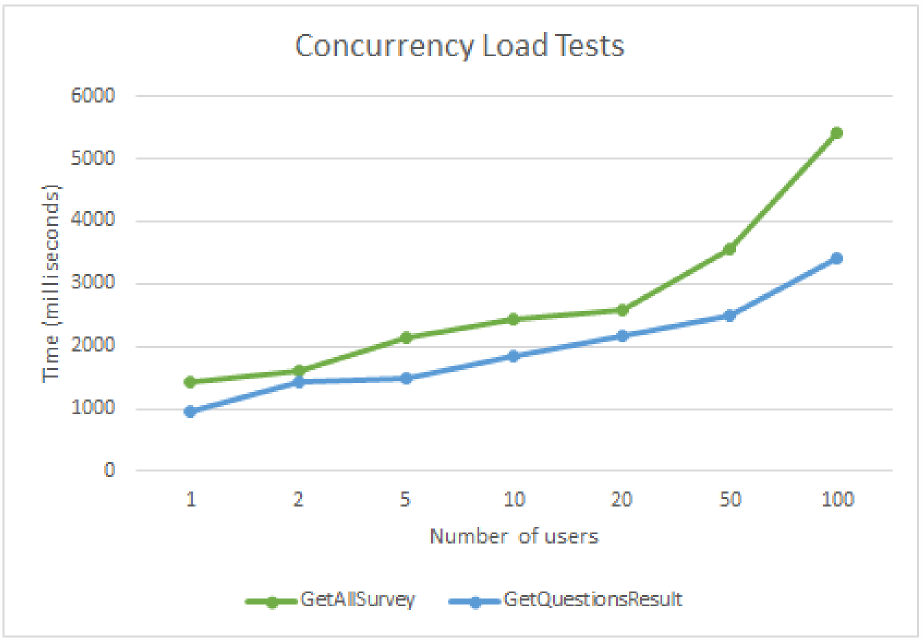
Test Type	**Concurrency Load Tests**
Testing Tool	Own developed tool
Acceptance Criteria	Response time must be below 3 seconds when 10 users requesting 500K bytes of data at the same time.

Implementation

The test must allow identifying the average response time when requesting 500K bytes. Requests are carried out simultaneously by several connected users. In order to do that, a request has been performed by different number of users: 1, 2, 10, 20, 50 and 100.

Result

The following figure indicates, even having a 10 concurrent user requests, the system is able to provide a response time below 3 seconds in order to have an acceptable reply on both functions (3 seconds for a 500K bytes petition of data).



6.1.7.2 Functional Test

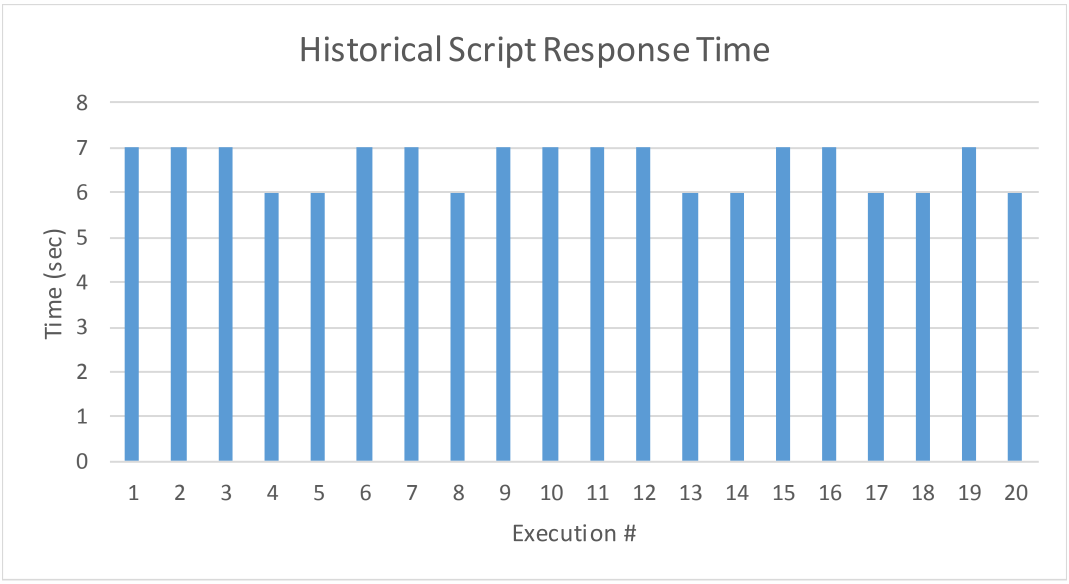
Module Name	API for Citizen Engagement Services
Test Type	**Functional Test**
Testing Tool	Own developed tool

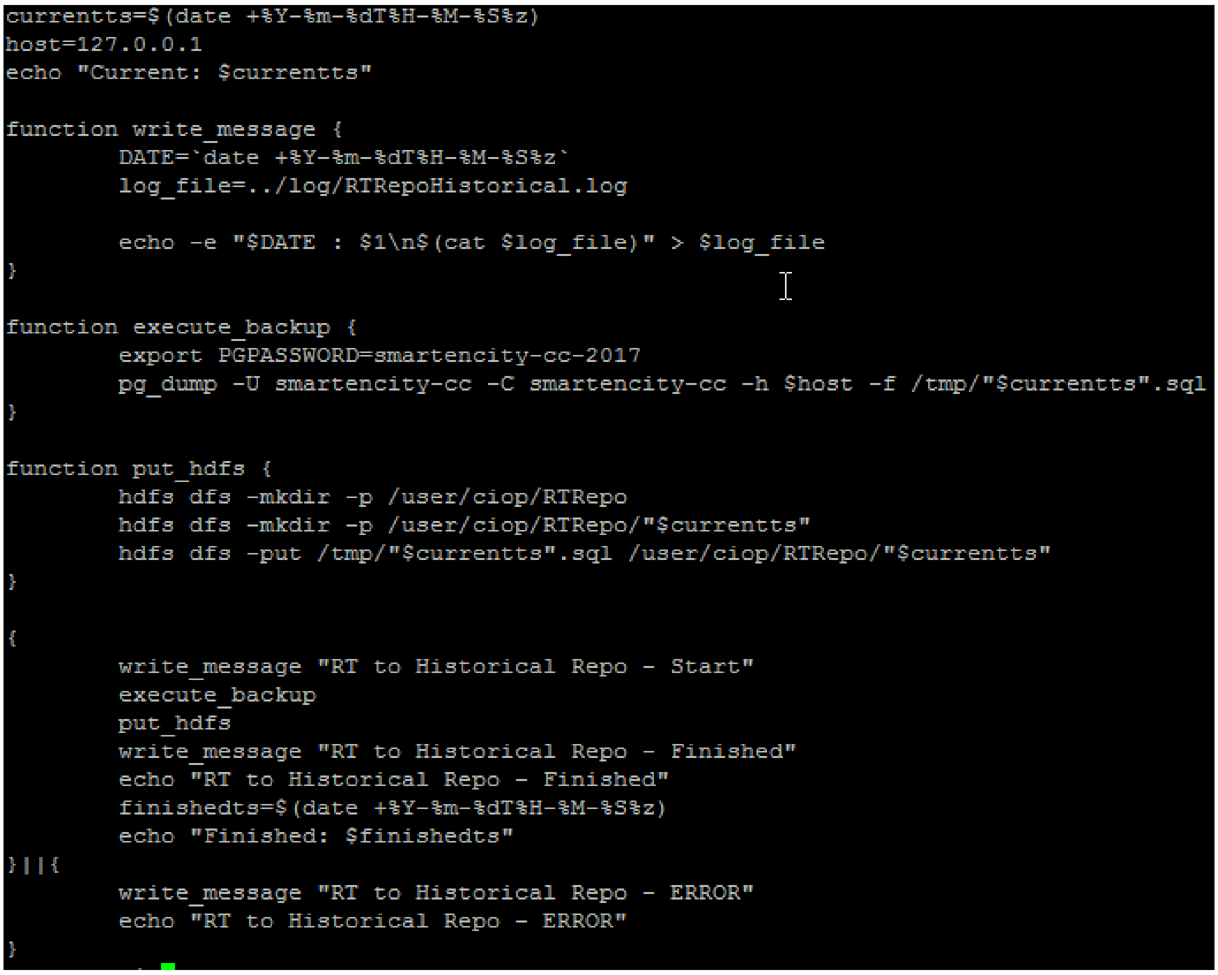
#	Functionality	Input	Output	Result (Ok/ NOk)
1	Request of all results of a specific survey	http://domain_name/GetAllSurvey?topic="poll2016"	[{ "survey": "poll2016", "questions": { "question": "#1", "valoration": 3 }, { "question": "#2", "valoration": 8 } } , { "survey": "poll2016", "questions": { "question": "#1", "valoration": 9 }, { "question": "#2", "valoration": 4 } }, {…}]	Ok
2	Request the result of a specific question of the survey	http://domain_name/ GetQuestionsResult?question ="2"	[{ "question": "#2", "valoration": 5 }, { "question": "#2", "valoration": 8 }, {…}]	Ok

6.1.8 ETL Processes (Historical Repository)

6.1.8.1 Performance Test

Module Name	ETL Processes (Historical Repository)
Test Type	**Response Time**
Testing Tool	Own developed tool (Script logs) See Figure 8
Acceptance Criteria	Not receiving time out messages and that the average response time do no exceeds 30 seconds for the whole process in order to be considered valid

Implementation
The script reads from database (SQL dump) and calls Hadoop filesystem to store the database dump. The script writes time consumed during the execution. This request has been performed 20 times and mean response time for the whole process has been calculated. The tests considered the same amount of registers in SQL (about 30000 registers).
Result
The average result in milliseconds is: 6619 The threshold for the response time is set in 30 seconds. The response time for the test is well below the threshold so the test has successfully passed. 

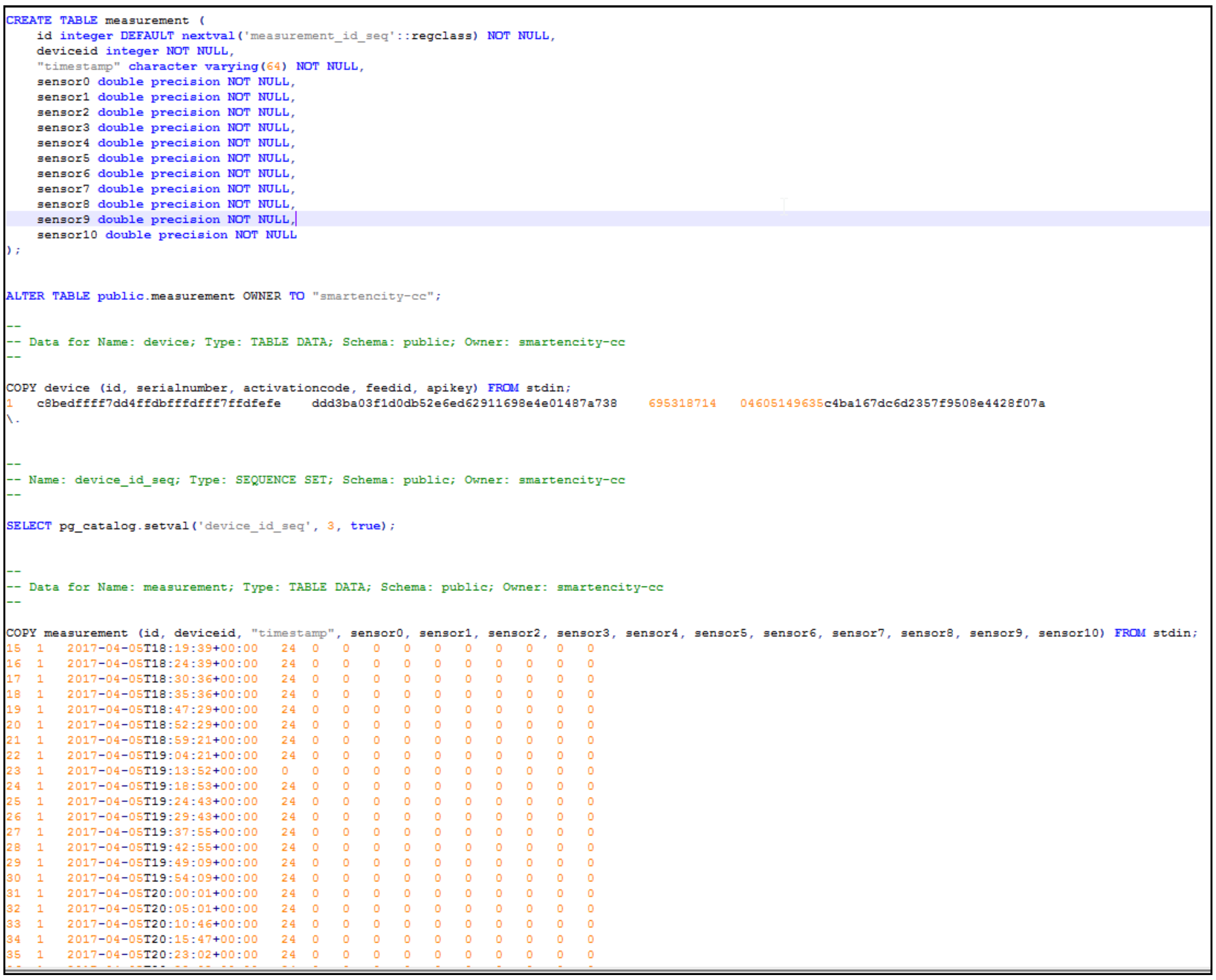


```
currentts=$(date +%Y-%m-%dT%H-%M-%S%z)
host=127.0.0.1
echo "Current: $currentts"

function write_message {
        DATE=`date +%Y-%m-%dT%H-%M-%S%z`
        log_file=../log/RTRepoHistorical.log

        echo -e "$DATE : $1\n$(cat $log_file)" > $log_file
}

function execute_backup {
        export PGPASSWORD=smartencity-cc-2017
        pg_dump -U smartencity-cc -C smartencity-cc -h $host -f /tmp/"$currentts".sql
}

function put_hdfs {
        hdfs dfs -mkdir -p /user/ciop/RTRepo
        hdfs dfs -mkdir -p /user/ciop/RTRepo/"$currentts"
        hdfs dfs -put /tmp/"$currentts".sql /user/ciop/RTRepo/"$currentts"
}

{

        write_message "RT to Historical Repo - Start"
        execute_backup
        put_hdfs
        write_message "RT to Historical Repo - Finished"
        echo "RT to Historical Repo - Finished"
        finishedts=$(date +%Y-%m-%dT%H-%M-%S%z)
        echo "Finished: $finishedts"
}||{
        write_message "RT to Historical Repo - ERROR"
        echo "RT to Historical Repo - ERROR"
}
```

Figure 8 Historical repository HDFS storing script

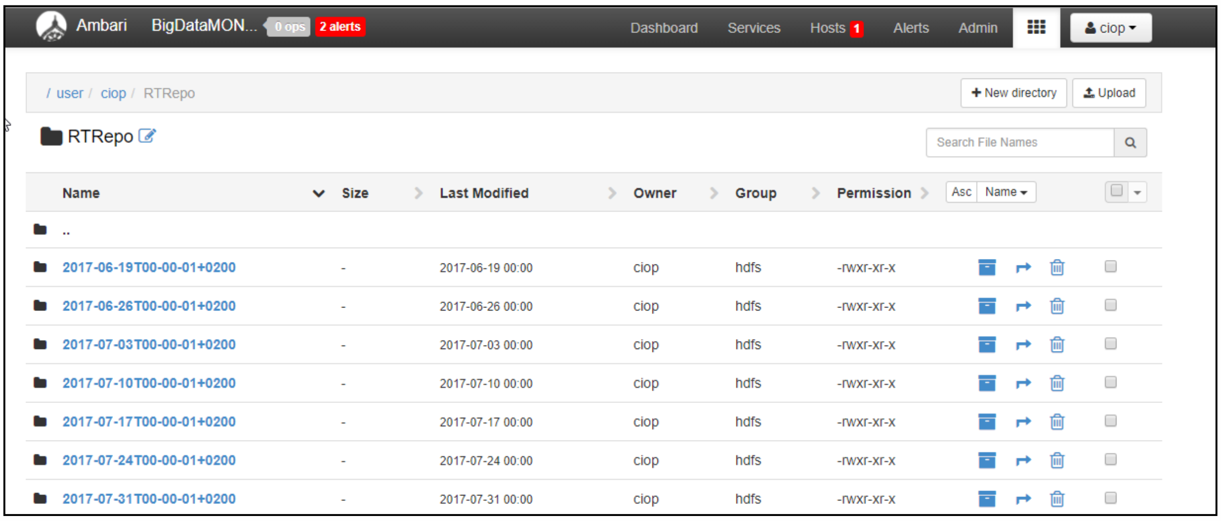
6.1.8.2 Functional Test

Module Name	ETL Processes (Historical Repository)			
Test Type	**Functional Test**			
Testing Tool	Run own development script			
#	**Functionality**	**Input**	**Output**	**Result (Ok/ NOk)**
1	Backup in HDFS from SQL dump	Input file (Postgre SQL backup file) See summary of csv file in Figure 9	None SQL Dump is stored in a folder using Hadoop (see Figure 10)	Ok



```
CREATE TABLE measurement (
    id integer DEFAULT nextval('measurement_id_seq'::regclass) NOT NULL,
    deviceid integer NOT NULL,
    "timestamp" character varying(64) NOT NULL,
    sensor0 double precision NOT NULL,
    sensor1 double precision NOT NULL,
    sensor2 double precision NOT NULL,
    sensor3 double precision NOT NULL,
    sensor4 double precision NOT NULL,
    sensor5 double precision NOT NULL,
    sensor6 double precision NOT NULL,
    sensor7 double precision NOT NULL,
    sensor8 double precision NOT NULL,
    sensor9 double precision NOT NULL,
    sensor10 double precision NOT NULL
);


ALTER TABLE public.measurement OWNER TO "smartencity-cc";

--
-- Data for Name: device; Type: TABLE DATA; Schema: public; Owner: smartencity-cc
--

COPY device (id, serialnumber, activationcode, feedid, apikey) FROM stdin;
1   c8bedffff7dd4ffdbfffdfff7ffdfefe    ddd3ba03f1d0db52e6ed62911698e4e01487a738    695318714   04605149635c4ba167dc6d2357f9508e4428f07a
\.


--
-- Name: device_id_seq; Type: SEQUENCE SET; Schema: public; Owner: smartencity-cc
--

SELECT pg_catalog.setval('device_id_seq', 3, true);


--
-- Data for Name: measurement; Type: TABLE DATA; Schema: public; Owner: smartencity-cc
--

COPY measurement (id, deviceid, "timestamp", sensor0, sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8, sensor9, sensor10) FROM stdin;
15  1   2017-04-05T18:19:39+00:00   24  0   0   0   0   0   0   0   0   0   0
16  1   2017-04-05T18:24:39+00:00   24  0   0   0   0   0   0   0   0   0   0
17  1   2017-04-05T18:30:36+00:00   24  0   0   0   0   0   0   0   0   0   0
18  1   2017-04-05T18:35:36+00:00   24  0   0   0   0   0   0   0   0   0   0
19  1   2017-04-05T18:47:29+00:00   24  0   0   0   0   0   0   0   0   0   0
20  1   2017-04-05T18:52:29+00:00   24  0   0   0   0   0   0   0   0   0   0
21  1   2017-04-05T18:59:21+00:00   24  0   0   0   0   0   0   0   0   0   0
22  1   2017-04-05T19:04:21+00:00   24  0   0   0   0   0   0   0   0   0   0
23  1   2017-04-05T19:13:52+00:00   0   0   0   0   0   0   0   0   0   0   0
24  1   2017-04-05T19:18:53+00:00   24  0   0   0   0   0   0   0   0   0   0
25  1   2017-04-05T19:24:43+00:00   24  0   0   0   0   0   0   0   0   0   0
26  1   2017-04-05T19:29:43+00:00   24  0   0   0   0   0   0   0   0   0   0
27  1   2017-04-05T19:37:55+00:00   24  0   0   0   0   0   0   0   0   0   0
28  1   2017-04-05T19:42:55+00:00   24  0   0   0   0   0   0   0   0   0   0
29  1   2017-04-05T19:49:09+00:00   24  0   0   0   0   0   0   0   0   0   0
30  1   2017-04-05T19:54:09+00:00   24  0   0   0   0   0   0   0   0   0   0
31  1   2017-04-05T20:00:01+00:00   24  0   0   0   0   0   0   0   0   0   0
32  1   2017-04-05T20:05:01+00:00   24  0   0   0   0   0   0   0   0   0   0
33  1   2017-04-05T20:10:46+00:00   24  0   0   0   0   0   0   0   0   0   0
34  1   2017-04-05T20:15:47+00:00   24  0   0   0   0   0   0   0   0   0   0
35  1   2017-04-05T20:23:02+00:00   24  0   0   0   0   0   0   0   0   0   0
```

Figure 9 SQL Dump input file (current.sql)



/ user / ciop / RTRepo

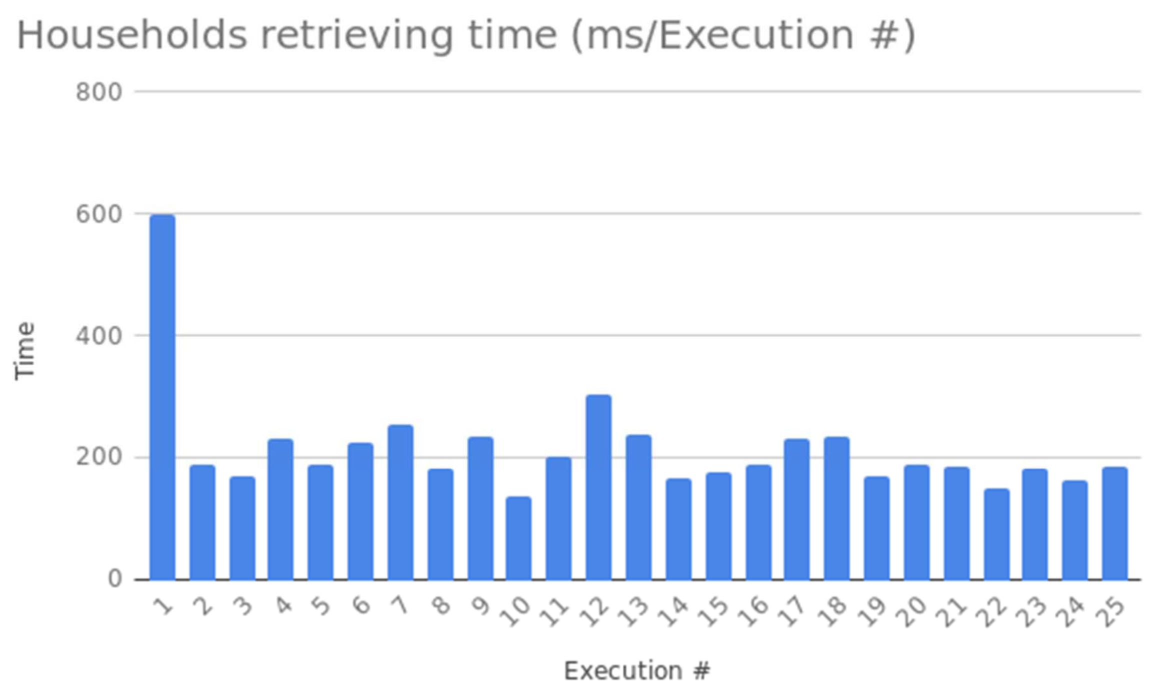
RTRepo

Name	Size	Last Modified	Owner	Group	Permission
..					
2017-06-19T00-00-01+0200	-	2017-06-19 00:00	ciop	hdfs	-rwxr-xr-x
2017-06-26T00-00-01+0200	-	2017-06-26 00:00	ciop	hdfs	-rwxr-xr-x
2017-07-03T00-00-01+0200	-	2017-07-03 00:00	ciop	hdfs	-rwxr-xr-x
2017-07-10T00-00-01+0200	-	2017-07-10 00:00	ciop	hdfs	-rwxr-xr-x
2017-07-17T00-00-01+0200	-	2017-07-17 00:00	ciop	hdfs	-rwxr-xr-x
2017-07-24T00-00-01+0200	-	2017-07-24 00:00	ciop	hdfs	-rwxr-xr-x
2017-07-31T00-00-01+0200	-	2017-07-31 00:00	ciop	hdfs	-rwxr-xr-x

Figure 10 CIOP Hadoop folders

6.1.9 ETL Processes (Structural Repository)

6.1.9.1 Performance Test

Module Name	ETL Processes (Structural Repository)
Test Type	**Response Time**
Testing Tool	Rester (Firefox Add-on)
Acceptance Criteria	The average of the response time must be below 500 milliseconds for the whole process in order to be considered valid
Implementation	
This test calls the REST API (Structural) to get information about dwellings. This request has been performed 25 times and average response time has been calculated.	
Result	
Result in millisecond: 216ms  The threshold for the response time is set in 500 milliseconds. So, the response time test has been successfully passed.	

6.1.9.2 Functional Test

Module Name	ETL Processes (Energy Vertical Repository)
Test Type	**Functional Test**
Testing Tool	Rester (Firefox add-on)

#	Functionalit	Input	Output	Resul

	y			t (Ok/ NOk)

4	Request all household information	http://structuralrepo.azurewebsites.net/api/Households	[
			  {
			    "$id": "1",
			    "HouseholdID": 1,
			    "Address": "2",
			    "ResidentsNumber": 4,
			    "SquareMeters": 90,
			    "BuildingID": 1,
			    "Exposure": "NW"
			  },
			…
			]
				Ok

5	Request specific Building information	http://structuralrepo.azurewebsites.net/api/Buildings/1	{
			  "$id": "1",
			  "BuildingID": 1,
			  "Street": "Avenida de los derechos humanos",
			  "Number": "30",
			  "DistrictID": 3,
			  "GisID": "590472000001"
			}
				Ok

6.1.10 GIS Repo Services

Module Name	GIS Repo services
Test Type	**Response Time**
Testing Tool	Own developed tool
Acceptance Criteria	The average of the response time must be below 2 seconds in order to be considered valid
Implementation	
This test obtains the building data contained in the GIS repo and calculates the mean response time. Using the buildings layer, two services are tested: GetMap and GetFeatureInfo This request has been performed 100 times and mean response time has been calculated.	
Result	
GetMap request: Average time in millisecond: 1189 **GetFeatureInfo Request:** Average time in millisecond: 220.2 The threshold for the response time is set in 2 seconds. So, the response time test has been successfully passed.	
Test Type	**Data volume Load Tests**
Testing Tool	Own developed tool

Acceptance Criteria	Response time must be below 2 seconds when requesting 100 entities in row.

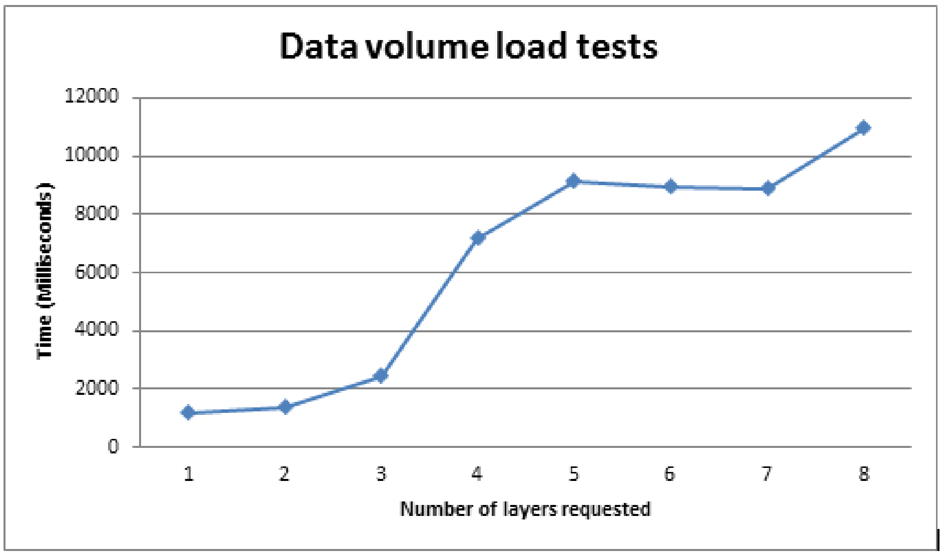
Implementation

This test should allow comparison of response times of transactions that require different volumes of data. In this case the GetMap request is going to be done increasing the number of layers requested.

Each request has been performed 100 times and mean response time has been calculated.

Result

As can be seen in the following figure, the bigger is the volume of data, the more time is needed in retrieve the data. However, the common use of requests is requesting one layer each time and the results for one layer are below the response time threshold (2 seconds).



Test Type	**Concurrency Load Tests**
Testing Tool	JMeter
Acceptance Criteria	

Implementation

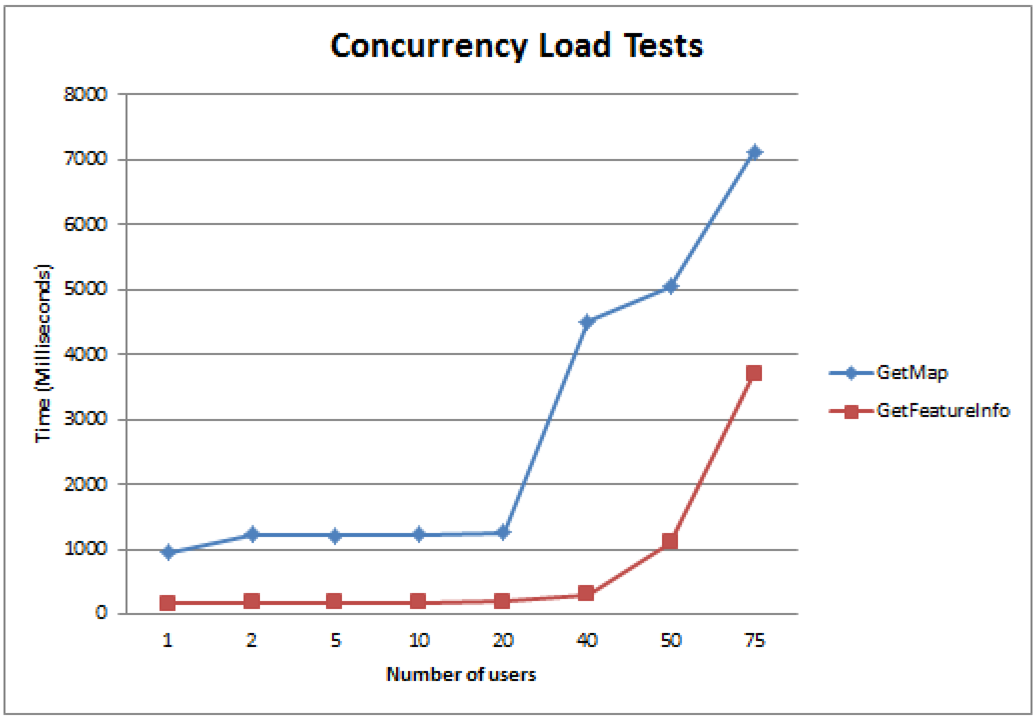
This test must allow identifying the average response time when performing several requests at the same time to different services. In this case two services are going to be tested: GetMap, GetFeatureInfo

Result

As can be seen, the bigger is the number of users, the more time is needed in retrieving the data for the same request.

For GetFeatureInfo request the response time remains constant until 50 request at the same time.

The GetMap response time is higher because the response is bigger but it also remains constant until 20 request at the same time.



6.1.11 GIS Structural Repo Services

6.1.11.1 Performance Test

Module Name	GIS Structural Repo services
Test Type	**Response Time**
Testing Tool	Own developed tool
Acceptance Criteria	The average of the response time must be below 2 seconds in order to be considered valid
Implementation	
This test obtains all the building data contained in the GIS structural repo and calculates the mean response time. Normally just a few amount of buildings are requested, and not all of them. This request has been performed 100 times and mean response time has been calculated.	
Result	
Result in millisecond: 1134 The threshold for the response time is set in 2 seconds. So, the response time test has been successfully passed.	
Test Type	**Data volume Load Tests**
Testing Tool	Own developed tool
Acceptance Criteria	Response time must be below 2 seconds when requesting 100 entities in a

	row.

Implementation

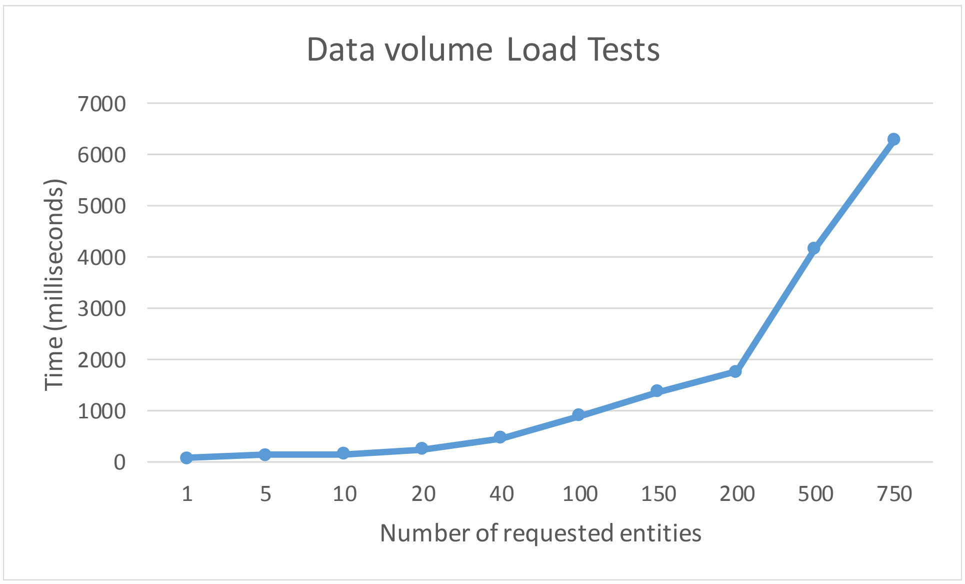
This test should allow comparison of response times of transactions that require different volumes of data. Because of that, in each request different number of surface geometries has been requested: 1, 5, 10, 20, 40, 100, 150, 200, 500 and 750.

Normally the surface geometries of a specific building or city element is requested, so requesting 750 surface geometries in a row is not a normal request.

Each request has been performed 100 times and mean response time has been calculated.

Result

As can be seen in the following figure, the bigger is the volume of data, the more time is needed in retrieve the data. However, for request with less than 200 surface geometries it still continues below the response time threshold (2 seconds).



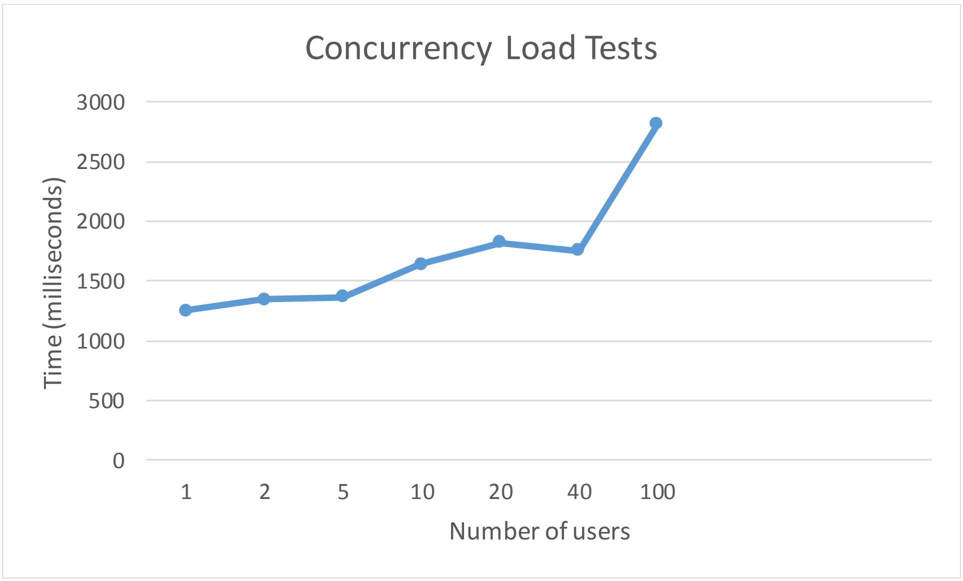
Test Type	**Concurrency Load Tests**
Testing Tool	Own developed tool
Acceptance Criteria	Response time must be below 2 seconds when 10 users requesting 150 entities at the same time.

Implementation

The test must allow identifying the average response time when requesting 150 surface geometry entities. Requests are carried out simultaneously by several connected users. In order to do that, a request has been performed by different number of users: 1, 2, 5, 10, 20, 40 and 100.

Result

As can be seen in the following figure, the bigger is the number of users, the more time is needed in retrieving the data when performing the same request. However, for requesting 150 surface geometry entities for 40 concurrent users it still continues below the response time threshold (2 seconds).



6.1.11.2 Functional Test

Module Name	GIS Structural Repo services
Test Type	**Functional Test**
Testing Tool	Deegree tool

#	Functionality	Input	Output	Result (Ok/ NOk)
1	Request all buildings information	<wfs:Query typeName='app:building_smartengasteizv1'/> </wfs:GetFeature>	<app:building_smartengasteizv1 gml:id="APP_BUILDING_SMARTENGASTEIZV1_322"> <app:id>322</app:id> <app:building_root_id> 321</app:building_root_id> <app:building_parent_id> 321</app:building_parent_id> <app:function>1000</app:function> <app:class>1000</app:class> <app:storeys_above_ground> 7</app:storeys_above_ground> <app:year_of_construction> 1962-01-01 </app:year_of_construction> <app:measured_height> 15.859760319895258</app:measured_height> ………. </app:building_smartengasteizv1> ………. <app:building_smartengasteizv1 gml:id="APP_BUILDING_SMARTENGASTEIZV1_337" > <app:id>337</app:id> …………	Ok
2	Request specific building information	<wfs:Query typeName='app:building_smartengasteizv1'><ogc:Filter><ogc:PropertyIsLike wildCard="*" singleChar="#" escapeChar="!"><ogc:PropertyName>app:id</o	<app:building_smartengasteizv1 gml:id=" 322"> <app:building_root_id>321</app:building_root_id> <app:building_parent_id>321</app:building_parent_id> <app:function>1000</app:function> <app:class>1000</app:class> <app:storeys_above_ground>7 </app:storeys_above_ground>	Ok

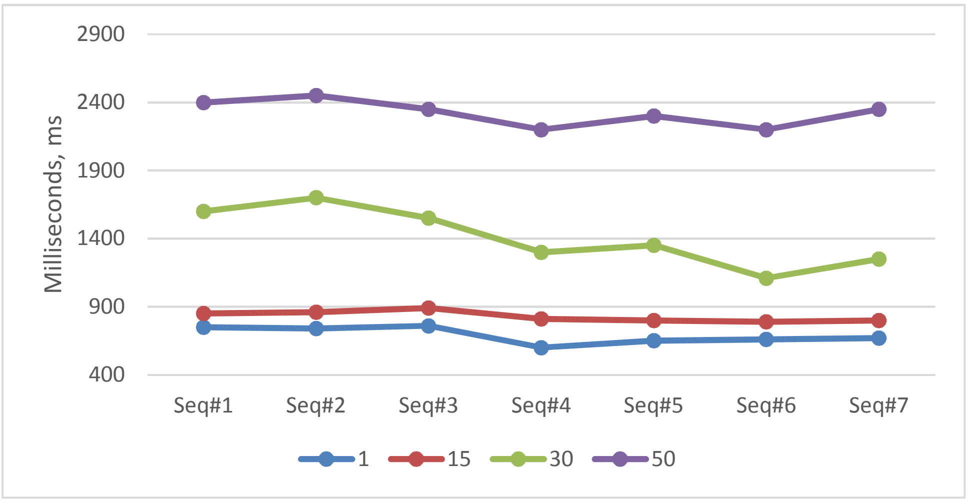
		gc:PropertyName><ogc:Literal>322</ogc:Literal></ogc:PropertyIsLike></ogc:Filter></wfs:Query>	<app:year_of_construction>1962-01-01</app:year_of_construction> <app:measured_height>15.85</app:measured_height> … </app:building_smartengasteizv1>	
3	Request all thematic surface information	<wfs:Query typeName='app:thematic_surface_smartengasteizv1'></wfs:Query>	<app:thematic_surface_smartengasteizv1 gml:id=" 449"> <app:id>449</app:id> <app:building_id>448</app:building_id> <app:objectclass_id>34</app:objectclass_id> <app:lod2_multi_surface_id>2332</app:lod2_multi_surface_id> </app:thematic_surface_smartengasteizv1> <app:thematic_surface_smartengasteizv1 gml:id="450”> <app:id>450</app:id> <app:building_id>448</app:building_id> <app:objectclass_id>34</app:objectclass_id> …………. </app:thematic_surface_smartengasteizv1> ………………………..	Ok
4	Request specific thematic surface information	<wfs:Query typeName='app:thematic_surface_smartengasteizv1'><ogc:Filter><ogc:PropertyIsLike wildCard="*" singleChar="#" escapeChar="!"><ogc:PropertyName>app:id</ogc:PropertyName><ogc:Literal>449</ogc:Literal></ogc:PropertyIsLike></ogc:Filter></wfs:Query>	<app:thematic_surface_smartengasteizv1 gml:id=" 449"> <app:id>449</app:id> <app:building_id>448</app:building_id> <app:objectclass_id>34</app:objectclass_id> <app:lod2_multi_surface_id>2332</app:lod2_multi_surface_id> </app:thematic_surface_smartengasteizv1>	Ok
5	Request all surface geometry information	<wfs:Query typeName='app:surface_geometry_smartengasteizv1'></wfs:Query>	<app:surface_geometry_smartengasteizv1 gml:id="2"> <gml:boundedBy> <gml:Envelope srsName="EPSG:25830"> <gml:lowerCorner>526429.995 4744340.146</gml:lowerCorner> <gml:upperCorner>526456.509 4744367.256</gml:upperCorner> </gml:Envelope> </gml:boundedBy> <app:id>2</app:id> <app:parent_id>1</app:parent_id> <app:root_id>1</app:root_id> <app:geometry> <gml:Polygon gml:id="2_APP_GEOMETRY" srsName="EPSG:25830"> <gml:exterior> <gml:LinearRing> <gml:posList>526432.307 4744356.543 515.183 526429.995 4744354.210 515.496 526431.847 4744351.752 515.340 526435.507 4744346.894 515.667 526437.529 4744344.211 515.641 526440.593 4744340.146 515.877 526443.518 4744342.331 515.566 526447.297 4744345.153 515.392 526449.299 4744346.649 515.336 526456.509 4744352.035 515.086 526445.138 4744367.256 514.921 526437.928 4744361.869 515.321 526436.599 4744360.876 515.357 526432.307 4744356.543 515.183 526432.307 4744356.543 515.183</gml:posList> </gml:LinearRing> </gml:exterior>	Ok

			`</gml:Polygon>` `</app:geometry>` `</app:surface_geometry_smartengasteizv1>` `<app:surface_geometry_smartengasteizv1 gml:id="3">` …………………..	
6	Request specific surface geometry information	`<wfs:Query typeName='app:surface_geometry_smartengasteizv1'><ogc:Filter><ogc:PropertyIsLike wildCard="*" singleChar="#" escapeChar="!"><ogc:PropertyName>app:id</ogc:PropertyName><ogc:Literal>2</ogc:Literal></ogc:PropertyIsLike></ogc:Filter></wfs:Query>`	`<app:surface_geometry_smartengasteizv1 gml:id="2">` `<gml:boundedBy>` `<gml:Envelope srsName="EPSG:25830">` `<gml:lowerCorner>526429.995 4744340.146</gml:lowerCorner>` `<gml:upperCorner>526456.509 4744367.256</gml:upperCorner>` `</gml:Envelope>` `</gml:boundedBy>` `<app:id>2</app:id>` `<app:parent_id>1</app:parent_id>` `<app:root_id>1</app:root_id>` `<app:geometry>` `<gml:Polygon gml:id="2_APP_GEOMETRY" srsName="EPSG:25830">` `<gml:exterior>` `<gml:LinearRing>` `<gml:posList>526432.307 4744356.543 515.183 526429.995 4744354.210 515.496 526431.847 4744351.752 515.340 526435.507 4744346.894 515.667 526437.529 4744344.211 515.641 526440.593 4744340.146 515.877 526443.518 4744342.331 515.566 526447.297 4744345.153 515.392 526449.299 4744346.649 515.336 526456.509 4744352.035 515.086 526445.138 4744367.256 514.921 526437.928 4744361.869 515.321 526436.599 4744360.876 515.357 526432.307 4744356.543 515.183 526432.307 4744356.543 515.183</gml:posList>` `</gml:LinearRing>` `</gml:exterior>` `</gml:Polygon>` `</app:geometry>` `</app:surface_geometry_smartengasteizv1>`	Ok

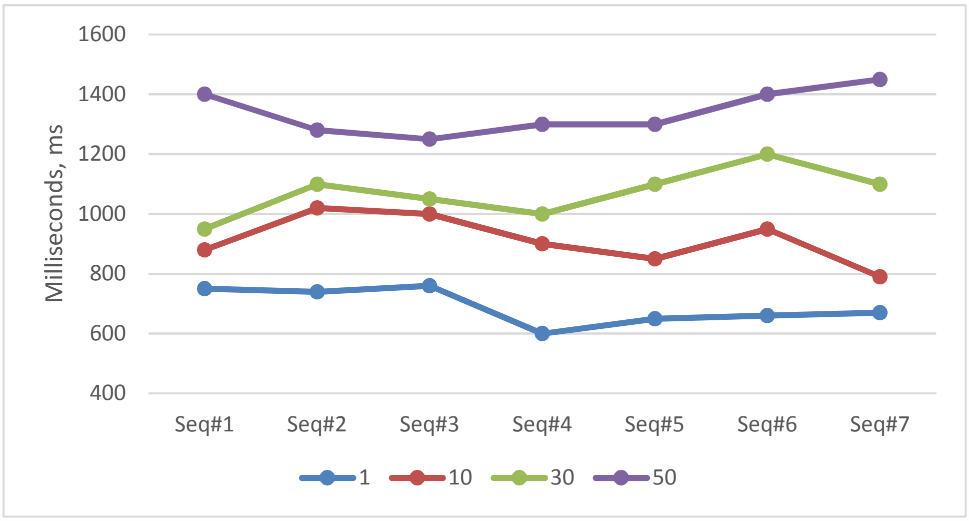
6.1.12 API for KPIs

6.1.12.1 Performance Test

Module Names	Get electricity consumption, Get electricity production, Get route
Test Type	**Response Time**
Testing Tool	Own developed
Acceptance Criteria	The average of the response time must be below 2 seconds for time intervals less than 12 months
Implementation	
Test is performed by a script executing this service 100 times in sequence. Input parameters are defined so requested time interval is less than 12 months.	
Result	
The average response time is less than 2 seconds.	

Test Type	**Data volume Load Tests**
Testing Tool	Own developed
Acceptance Criteria	Response time must be below 3 seconds when requesting data without defined time periods (all data) in sequence
Implementation	
Test is performed by a script executing this service 100 times in sequence. No input parameters defined for time interval – all data is returned. Maximum possible interval can be returned – 60 months.	
Result	
The average response time is less than 3 seconds.	
Test Type	**Concurrency Load Tests**
Testing Tool	Own developed
Acceptance Criteria	Response time must be within 2-4 seconds depending on request parameters for 50 concurrent users
Implementation	
Test is performed by a script executing this service 100 times in sequence. Mixed input parameters are used – with time interval defined and without. Maximum possible interval can be returned – 60 months.	
Result	
	

Module Names	Get last position, Count number of devices installed
Test Type	**Response Time**
Testing Tool	Own developed
Acceptance Criteria	The average of the response time must be below 1 second
Implementation	
Test is performed by a script executing this service 100 times in sequence.	

Result	
The average response time is less than 1 second.	
Test Type	**Data volume Load Tests**
Testing Tool	NA
Acceptance Criteria	NA
Implementation	
NA	
Result	
NA	
Test Type	**Concurrency Load Tests**
Testing Tool	Own developed
Acceptance Criteria	Response time must be below 1-2 second
Implementation	
Test is performed by a script executing this service 100 times in sequence for 50 concurrent users simultaneously.	
Result	
	

6.1.12.2 *Functional Test*

Module Name	Get electricity consumption			
Test Type	**Functional Test**			
Testing Tool	Own developed			
#	**Functionality**	**Input**	**Output**	**Result (Ok/NOk)**

1	Return electricity consumption from block level		Kwh consumed	Ok
2	Return electricity consumption from the building	{“address”:”Sonderborg, Test address 111, Denmark”}	Kwh consumed in the requested building	Ok

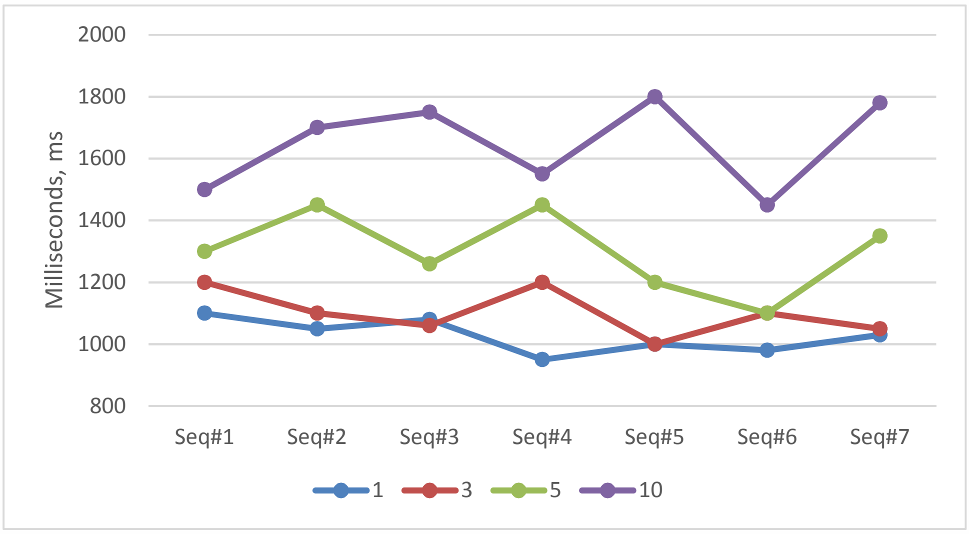
Module Name	Get electricity production			
Test Type	**Functional Test**			
Testing Tool	Own developed			
#	**Functionality**	**Input**	**Output**	**Result (Ok/NOk)**
1	Return electricity production from block level		Kwh produced (number)	Ok
2	Return electricity production from building	{“address”:”Sonderborg, Test address 111, Denmark”}	Kwh produced in requested building (number)	Ok

Module Name	Get route			
Test Type	**Functional Test**			
Testing Tool	Own developed			
#	**Functionality**	**Input**	**Output**	**Result (Ok/NOk)**
1	Return driven route from eCar	{“device”:11,”from”: 2017-04-23T18:25:43.511Z ,”to”: 2017-05-23T18:25:43.511Z }	Array of locations ids and timestamps (JSON)	Ok

6.1.13 Services for Integrating Open Data

6.1.13.1 Performance Test

Module Names	Get electricity prices, Get CO2 emission levels
Test Type	**Response Time**
Testing Tool	Own developed
Acceptance Criteria	The average of the response time must be below 2 seconds
Implementation	
Test is performed by a script executing this service 100 times in sequence.	

Result	
The average response time is less than 2 seconds.	
Test Type	**Data volume Load Tests**
Testing Tool	Own developed
Acceptance Criteria	Response time must be below 3-4 seconds.
Implementation	
Test is performed by a script executing this service 100 times in sequence.	
Result	
The average response time is less than 3-4 seconds.	
Test Type	**Concurrency Load Tests**
Testing Tool	Own developed
Acceptance Criteria	Response time must be within 2-4 seconds depending on request parameters for 10 concurrent users
Implementation	
Test is performed by a script executing this service 100 times in sequence for 10 concurrent users.	
Result	
	

6.1.13.2 Functional Test

Module Name	Get electricity prices
Test Type	**Functional Test**
Testing Tool	Own developed

#	Functionality	Input	Output	Result (Ok/NOk)

1	Get electricity prices forecast for next 24 hrs	N/A	Array of prices for next day (JSON)	Ok

Module Name	Get CO2 emission levels			
Test Type	**Functional Test**			
Testing Tool	Own developed			
#	**Functionality**	**Input**	**Output**	**Result (Ok/NOk)**
1	Get CO2 emission levels forecast for next 24 hrs	N/A	Array of CO2 emission levels for next day (JSON)	Ok

6.2 Integration Tests

6.2.1 Integration test 1: Energy Efficiency Test Scenario

Scenario Id:	ENERGYEFFICIENCY_01	
Scenario Name:	Energy efficiency demonstrator	
Scenario Description	The Energy Efficiency demonstrator implements the project Reference Architecture in the lighthouse of Vitoria-Gasteiz. The scenario consists in testing the (1) acquisition of building data coming from several sensors installed in a neighbourhood, (2) the transformation of this real-time data into a historical repository and GIS repository through ETL mechanisms, (3) the publication of APIs in the interoperability layer capable of querying data from the historical repository and (4) the visualization of the aggregated data from a dashboard that leverages the interoperability API.	
Test Environment		
Hardware	**Software**	**Network Config.**
Temperature sensors Humidity sensors Energy consumption meters	Chrome/Firefox web browser RESTful clients to test the endpoints	Demonstrator access via HTTP

Acceptance Criteria	
Criterion	**Value**
Response time	Below a given value
Failure response	Error notifications
Concurrency load response	Minimum number of queries/users

Awareness to input data changes		Smoothness / Page reload / Asynchronous
List of Modules Involved		
#	**Module**	**Technology**
1	Data acquisition about energy	WebService
2	ETL Processes (Energy Vertical Repo)	WebService Real-time repository
3	ETL Process (Historical repo)	Script RDBMS / NoSQL ETL mechanisms to leverage database search and query commands to transfer data from real-time to historical repositories.
4	ETL Processes (Structural Repo)	WebService
5	GIS Structural Repo Services	WebService
6	API for Energy Services	Secured API to provide data to the Intelligent Services layer Data processed by the Knowledge layer is offered by RESTful API
7	GIS Repo Services	WebService ETL mechanisms to feed GIS repository with average values of temperature and energy consumption.
8	Energy Application	Web application Leverages the Interoperability layer API to provide the user experience

The list of modules involved in the completion of this test scenario is depictured in Figure 11. The queries / transaction information related to these modules, as well as the test results, are detailed in the sections below. Although the modules are numbered consecutively, the processes are not necessarily sequentially

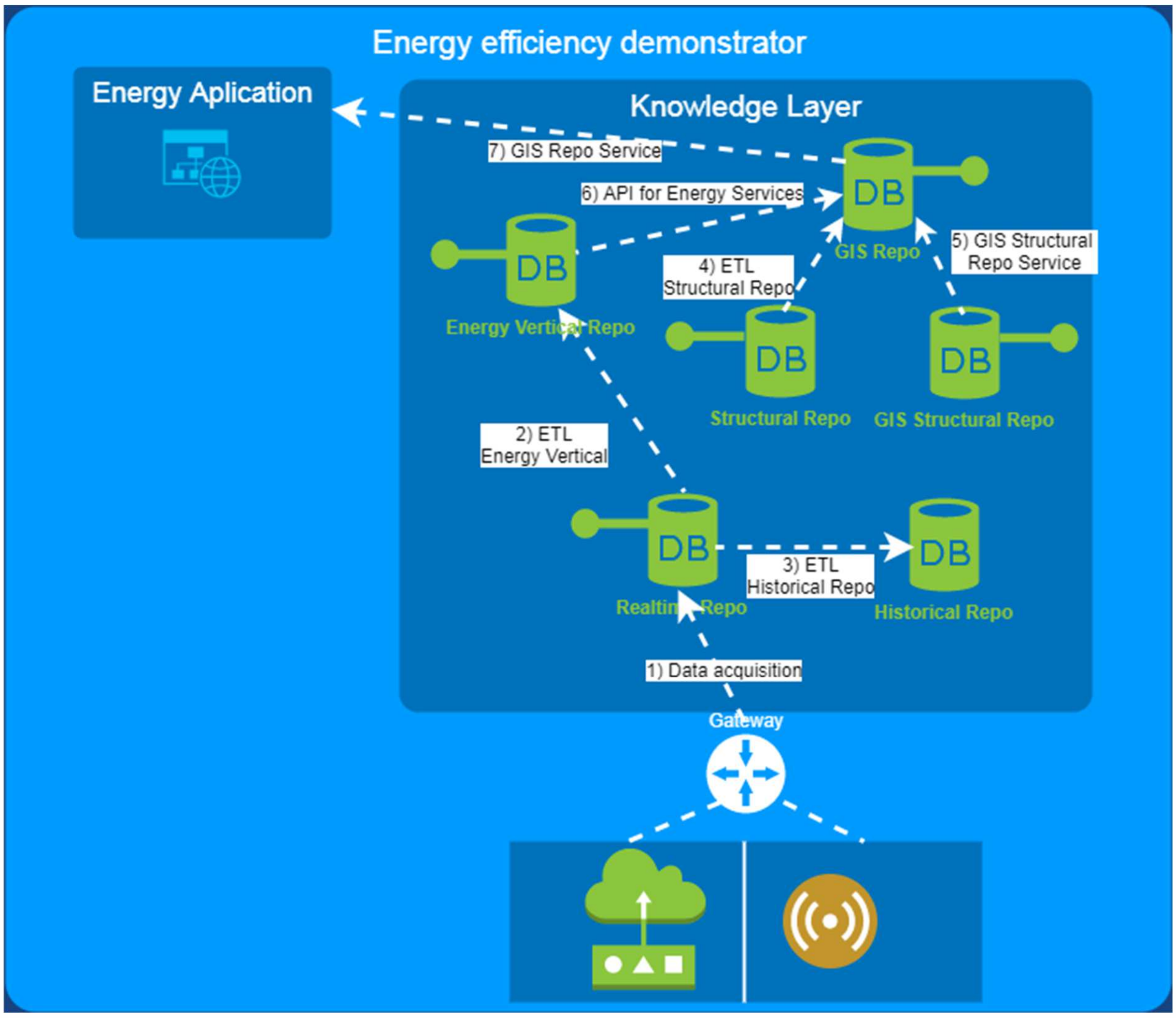


Figure 11 Energy Efficiency test scenario

6.2.1.1 Module 1: Data acquisition about energy

Simulated sensor data are stored on time series tables into the real time repository using the REST API provided by the Repo. For the test scenario two sensors are used: A temperature sensor and a consumption meter. Selected sensors are:

- Temperature Sensor: sensor9
- Consumption Meter: sensor0

The following table shows the data of the selected sensors stored in the real time repository for a selected timestamp (1 hour from '2017-12-21-11:00 to 2017-12-21-12:00'). Data are stored every 5 minutes.

Query / Transaction	**Result**
SELECT * FROM measurement WHERE timestamp>'2017-12-21T11' AND timestamp<'2017-12-21T12';	id | deviceid | timestamp | sensor0 | sensor9 -------+----------+--------------------------+---------+---------+---- 30301 | 1 | 2017-12-21T11:03:18+00:00 | 965 |…22 30302 | 1 | 2017-12-21T11:08:19+00:00 | 970 | 22

	30303 | 1 | 2017-12-21T11:13:19+00:00 | 962 | 22 30304 | 1 | 2017-12-21T11:18:19+00:00 | 975 | 22 30305 | 1 | 2017-12-21T11:23:20+00:00 | 985 | 23 30306 | 1 | 2017-12-21T11:28:20+00:00 | 986 | 23 30307 | 1 | 2017-12-21T11:33:21+00:00 | 995 | 23 30308 | 1 | 2017-12-21T11:38:21+00:00 | 990 | 23 30309 | 1 | 2017-12-21T11:43:21+00:00 | 984 | 23 30310 | 1 | 2017-12-21T11:48:22+00:00 | 983 | 23 30311 | 1 | 2017-12-21T11:53:22+00:00 | 988 | 23 30312 | 1 | 2017-12-21T11:58:23+00:00 | 979 | 23

6.2.1.2 Module 2: ETL Processes (Energy Vertical Repo)

Real Time data are aggregated and stored in the Energy Vertical Repository. This step exports a table from the time series tables (Real Time Repository) into the Energy Vertical Repo. The query is performed for the selected sensors and timestamp

The following table shows the data of the selected sensors stored in the Energy Vertical Repository for a selected timestamp. The data "Value" represents the mean value for the selected TimeStamp ('2017-12-21-11:00 to 2017-12-21-12:00').

- Temperature Sensor: sensor9 -> DeviceSetupID": 3
- Consumption Meter: sensor0 -> DeviceSetupID": 2

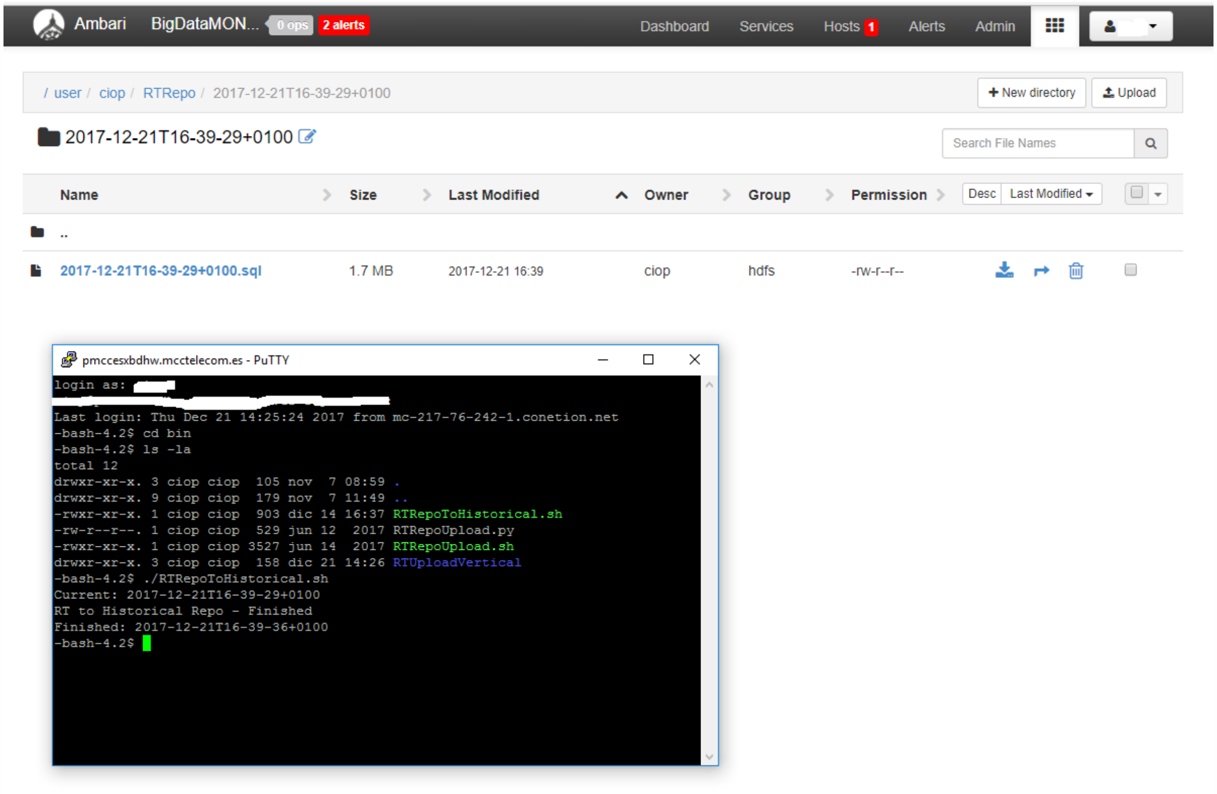
Query / Transaction	Result
http://energyrepo.azurewebsites.net/api/Measurements	... { "$id": "52060", "MeasurementID": 52088, **"Value": 980.166666666667,** "EstimatedCost": null, "StartDate": "2017-12-21T11:00:00", "EndDate": "2017-12-21T12:00:00", "StartDateTimestamp": 1513850400, "EndDateTimestamp": 1513854000, "**DeviceSetupID": 2** }, { "$id": "52061", "MeasurementID": 52089, **"Value": 22.6666666666667,** "EstimatedCost": null,

	"StartDate": "2017-12-21T11:00:00", "EndDate": "2017-12-21T12:00:00", "StartDateTimestamp": 1513850400, "EndDateTimestamp": 1513854000, **"DeviceSetupID": 3** }, …

6.2.1.3 *Module 3: ETL Process (Historical repo)*

Periodically, data from Real Time Repository (Postgre SQL) are backed up in a Hadoop file system (folder with timestamp name).

The following figure shows the Hadoop folder where the backup data are stored (see Figure 12).



/ user / ciop / RTRepo / 2017-12-21T16-39-29+0100

2017-12-21T16-39-29+0100

Name	Size	Last Modified	Owner	Group	Permission
..					
2017-12-21T16-39-29+0100.sql	1.7 MB	2017-12-21 16:39	ciop	hdfs	-rw-r--r--

pmccesxbdhw.mcctelecom.es - PuTTY

```
login as: [redacted]
[redacted]
Last login: Thu Dec 21 14:25:24 2017 from mc-217-76-242-1.conetion.net
-bash-4.2$ cd bin
-bash-4.2$ ls -la
total 12
drwxr-xr-x. 3 ciop ciop  105 nov  7 08:59 .
drwxr-xr-x. 9 ciop ciop  179 nov  7 11:49 ..
-rwxr-xr-x. 1 ciop ciop  903 dic 14 16:37 RTRepoToHistorical.sh
-rw-r--r--. 1 ciop ciop  529 jun 12  2017 RTRepoUpload.py
-rwxr-xr-x. 1 ciop ciop 3527 jun 14  2017 RTRepoUpload.sh
drwxr-xr-x. 3 ciop ciop  158 dic 21 14:26 RTUploadVertical
-bash-4.2$ ./RTRepoToHistorical.sh
Current: 2017-12-21T16-39-29+0100
RT to Historical Repo - Finished
Finished: 2017-12-21T16-39-36+0100
-bash-4.2$
```

Figure 12 Hadoop backup folder

6.2.1.4 *Module 4: ETL Processes (Structural Repo)*

It is used to associate real time data with elements in the environment (buildings and dwellings). First the measurement is linked to a specific Device (DeviceSetupID=2 and DeviceSetupID=3), then the Gateway related to the device (GatewayID), then the associated dweling (HouseholdID), and finally the Building (BuildingID) and the corresponding identifier in the GIS Repo (GisID).

Query / Transaction	Result

http://energyrepo.azurewebsites.net/api/devicesetups/2	{
	"$id": "1",
	"DeviceSetupID": 2,
	"SetupDate": null,
	"MeasureTypeID": 1,
	"UnitID": 11,
	"ConstantConversionValue": 1,
	"LocationID": 7,
	"ManufacturingYear": 2017,
	"GatewayID": 2,
	"ChannelNumber": 0
	}

http://energyrepo.azurewebsites.net/api/DeviceSetups/3	{
	"$id": "3",
	"DeviceSetupID": 3,
	"SetupDate": null,
	"MeasureTypeID": 2,
	"UnitID": 1,
	"ConstantConversionValue": 1,
	"LocationID": 7,
	"ManufacturingYear": 2017,
	"GatewayID": 2,
	"ChannelNumber": 9
	}

http://energyrepo.azurewebsites.net/api/gateways/2	{
	"$id": "1",
	"GatewayID": 2,
	"SerialNumber": "c8bedffff7dd4ffdbfffdfff7ffdfefe",
	"ActivationCode": "ddd3ba03f1d0db52e6ed62911698e4e01487a738",
	"ApiKey": "04605149635c4ba167dc6d2357f9508e4428f07a",
	"Comments": "#DemoGateway",
	"HouseholdID": 33,
	"GatewayModelID": 1,
	"FeedID": 695318714
	}

http://structuralrepo.azurewebsites.net/api/Households/33	{ "$id": "1", "HouseholdID": 33, "Address": "1-A",

	"ResidentsNumber": 2, "SquareMeters": 85, **"BuildingID": 1,** "Exposure": "SE" }

http://structuralrepo.azurewebsites.net/api/Buildings/1	{
	"$id": "1",
	"BuildingID": 1,
	"Street": "Example street",
	"Number": "1",
	"DistrictID": 3,
	"GisID": "590472000001"
	}

6.2.1.5 *Module 5: GIS Structural Repo Services*

It allows the retrieval of the data about georeferenced buildings contained in the GIS structural repo. The services provide access to cartographic representation of all the buildings and other city elements in the study area. This information will set the cartographic information of the GIS repo.

The following table shows the retrieved data from the GIS Structural Repo for the building where the sensors of the scenario are located. In addition to the geometric representation of each building, information such as year of construction or address is obtained from this repo.

Query / Transaction	Result
Get Building Info: <wfs:GetFeature outputFormat='text/xml; subtype=gml/3.2.1' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xmlns:ogc='http://www.opengis.net/ogc' xmlns:wfs='http://www.opengis.net/wfs'> <wfs:Query typeName='app:building_smartengasteizv1'> <ogc:Filter> <ogc:PropertyIsLike wildCard="*" singleChar="#" escapeChar="!"> <ogc:PropertyName>app:id</ogc:PropertyName><ogc:Literal>239</ogc:Literal></ogc:PropertyIsLike> </ogc:Filter> </wfs:Query> </wfs:GetFeature>	<gml:FeatureCollection xsi:schemaLocation=">amp;http://www.opengis.net/gml/3.2 http://schemas.opengis.net/gml/3.2.1/deprecatedTypes.xsd http://www.deegree.org/app http://3dcity.tecnalia.com/ServiciosWeb/services/smartengasteizv1?SERVICE=WFS&amp;VERSION=1.1.0&amp;REQUEST=DescribeFeatureType&amp;OUTPUTFORMAT=text%2Fxml%3B+subtype%3Dgml%2F3.2.1&amp;TYPENAME=app:building_smartengasteizv1&amp;NAMESPACE=xmlns(app=http%3A%2F%2Fwww.deegree.org%2Fapp)" gml:id="WFS_RESPONSE" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:gml="http://www.opengis.net/gml/3.2"> <gml:featureMember> <app:building_smartengasteizv1 gml:id="APP_BUILDING_SMARTENGASTEIZV1_239" xmlns:app="http://www.deegree.org/app"> <app:id>239</app:id> <app:building_root_id>200</app:building_root_id> <app:building_parent_id>200</app:building_parent_id> <app:function>1000</app:function> <app:class>1000</app:class> <app:storeys_above_ground>6</app:storeys_above_ground> **<app:year_of_construction>1998**</app:year_of_construction> <app:measured_height>18.107929176153675</app:measured_height>

	<app:storeys_above_ground>6</app:storeys_above_ground> <app:lod2_solid_id>1720</app:lod2_solid_id> <app:viviendas>10</app:viviendas> </app:building_smartengasteizv1> </gml:featureMember> </gml:FeatureCollection>
Get Building Address: <wfs:GetFeature outputFormat='text/xml; subtype=gml/3.2.1' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xmlns:ogc='http://www.opengis.net/ogc' xmlns:wfs='http://www.opengis.net/wfs'> <wfs:Query typeName='app:address_smartengasteizv1'> <ogc:Filter> <ogc:PropertyIsLike wildCard="*" singleChar="#" escapeChar="!"> <ogc:PropertyName>app:id</ogc:PropertyName><ogc:Literal>239</ogc:Literal></ogc:PropertyIsLike> </ogc:Filter> </wfs:Query> </wfs:GetFeature>	<gml:FeatureCollection xsi:schemaLocation=">amp;http://www.opengis.net/gml/3.2 http://schemas.opengis.net/gml/3.2.1/deprecatedTypes.xsd http://www.deegree.org/app http://3dcity.tecnalia.com/ServiciosWeb/services/smartengasteizv1?SERVICE=WFS&amp;VERSION=1.1.0&amp;REQUEST=DescribeFeatureType&amp;OUTPUTFORMAT=text%2Fxml%3B+subtype%3Dgml%2F3.2.1&amp;TYPENAME=app:address_smartengasteizv1&amp;NAMESPACE=xmlns(app=http%3A%2F%2Fwww.deegree.org%2Fapp)" gml:id="WFS_RESPONSE" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:gml="http://www.opengis.net/gml/3.2"> <gml:featureMember> <app:address_smartengasteizv1 gml:id="APP_ADDRESS_smartengasteizv1_0" xmlns:app="http://www.deegree.org/app"> <app:id>0</app:id> **<app:street>PORTAL DE ARRIAGA**</app:street> **<app:house_number>1A**</app:house_number> <app:zip_code>01012</app:zip_code> <app:city>Vitoria</app:city> <app:xal_source> PORTAL DE ARRIAGA, 1A, 01012, Vitoria, Spain</app:xal_source> </app:address_smartengasteizv1> </gml:featureMember> </gml:FeatureCollection>

6.2.1.6 *Module 6: API for Energy Services*

The information on average temperature and energy consumption of each building is obtained from the Energy Vertical Repository. GIS repo gets access to this API to collect and georeference these data.

The following table shows the values of energy consumption and average temperature for the selected building and day (2017/12/21). The information is retrieved per hour and atypical values obtained from measurement errors are observed. These values are discarded to calculate the daily average values.

Query / Transaction	**Result**

Energy
 http://smartencityportal.azurewebsites.net/EnergyEfficiency/GetBuilding
	{
	 "BuildingID": "590472000001",
	 "TotalConsumption": 990.247,
	 "Unit": "kW",
	 }

Consumption?BuildingID=590472000001&StartDate=2017-12-20&EndDate=2017-12-21	
Values used to calculate the Aggregated value http://smartencityportal.azurewebsites.net/EnergyEfficiency/GetBuildingConsumption?BuildingID=590472000001&StartDate=2017-12-20&EndDate=2017-12-21	[{"MeasurementID":52064,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-20T23:00:00","EndDate":"2017-12-21T00:00:00","StartDateTimestamp":1513807200,"EndDateTimestamp":1513810800,"DeviceSetupID":2}, {"MeasurementID":52066,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T00:00:00","EndDate":"2017-12-21T01:00:00","StartDateTimestamp":1513810800,"EndDateTimestamp":1513814400,"DeviceSetupID":2}, {"MeasurementID":52068,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T01:00:00","EndDate":"2017-12-21T02:00:00","StartDateTimestamp":1513814400,"EndDateTimestamp":1513818000,"DeviceSetupID":2}, {"MeasurementID":52070,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T02:00:00","EndDate":"2017-12-21T03:00:00","StartDateTimestamp":1513818000,"EndDateTimestamp":1513821600,"DeviceSetupID":2}, {"MeasurementID":52072,"**Value":0.0,"**EstimatedCost":null,"StartDate":"2017-12-21T03:00:00","EndDate":"2017-12-21T04:00:00","StartDateTimestamp":1513821600,"EndDateTimestamp":1513825200,"DeviceSetupID":2}, {"MeasurementID":52074,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T04:00:00","EndDate":"2017-12-21T05:00:00","StartDateTimestamp":1513825200,"EndDateTimestamp":1513828800,"DeviceSetupID":2}, {"MeasurementID":52076,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T05:00:00","EndDate":"2017-12-21T06:00:00","StartDateTimestamp":1513828800,"EndDateTimestamp":1513832400,"DeviceSetupID":2}, {"MeasurementID":52078,"**Value":5285.0**,"EstimatedCost":null,"StartDate":"2017-12-21T06:00:00","EndDate":"2017-12-21T07:00:00","StartDateTimestamp":1513832400,"EndDateTimestamp":1513836000,"DeviceSetupID":2}, {"MeasurementID":52080,"**Value":12573.**0,"EstimatedCost":null,"StartDate":"2017-12-21T07:00:00","EndDate":"2017-12-21T08:00:00","StartDateTimestamp":1513836000,"EndDateTimestamp":1513839600,"DeviceSetupID":2}, {"MeasurementID":52082,"**Value":12239.0**,"EstimatedCost":null,"StartDate":"2017-12-21T08:00:00","EndDate":"2017-12-21T09:00:00","StartDateTimestamp":1513839600,"EndDateTimestamp":1513843200,"DeviceSetupID":2}, {"MeasurementID":52084,"**Value":12872.0**,"EstimatedCost":null,"StartDate":"2017-12-21T09:00:00","EndDate":"2017-12-21T10:00:00","StartDateTimestamp":1513843200,"EndDateTimestamp":1513846800,"DeviceSetupID":2}, {"MeasurementID":52086,"**Value":12657.0**,"EstimatedCost":null,"StartDate":"2017-12-21T10:00:00","EndDate":"2017-12-21T11:00:00","StartDateTimestamp":1513846800,"EndDateTimestamp":1513850400,"DeviceSetupID":2}, {"MeasurementID":52088,"**Value":980.166666666667**,"EstimatedCost":null,"StartDate":"2017-12-21T11:00:00","EndDate":"2017-12-21T12:00:00","StartDateTimestamp":1513850400,"EndDateTimestamp":1513854000,"DeviceSetupID":2}, {"MeasurementID":52090**,"Value":973.25**,"EstimatedCost":null,"StartDate":"2017-12-21T12:00:00","EndDate":"2017-12-21T13:00:00","StartDateTimestamp":1513854000,"EndDateTimestamp":1513857600,"DeviceSetupID":2}, {"MeasurementID":52092,"**Value":975.571428571429**,"EstimatedCost":null,"StartDate":"2017-12-21T13:00:00","EndDate":"2017-12-21T14:00:00","StartDateTimestamp":1513857600,"EndDateTimestamp":1513861200,"DeviceSetupID":2}]

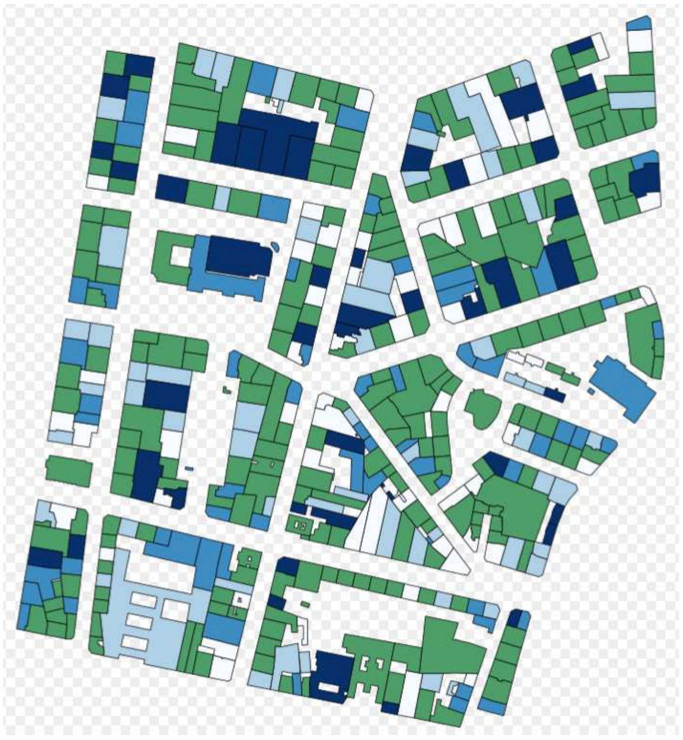
Average Temperature
 	
http://smartencityportal.azurewebsites.net/EnergyEfficiency/GetBuilding
	{
	
	"BuildingID": "590472000001",
	
	"AverageTemperature": 21.166,
	
	"Unit": "ºC"
	
	}

Temperature?BuildingID=590472000001&StartDate=2017-08-01&EndDate=2017-08-04	
Values used to calculate the Average Temperature http://smartencityportal.azurewebsites.net/EnergyEfficiency/GetBuildingTemperature?BuildingID=590472000001&StartDate=2017-12-20&EndDate=2017-12-21	[{"MeasurementID":52065,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-20T23:00:00","EndDate":"2017-12-21T00:00:00","StartDateTimestamp":1513807200,"EndDateTimestamp":1513810800,"DeviceSetupID":3}, {"MeasurementID":52067,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T00:00:00","EndDate":"2017-12-21T01:00:00","StartDateTimestamp":1513810800,"EndDateTimestamp":1513814400,"DeviceSetupID":3}, {"MeasurementID":52069,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T01:00:00","EndDate":"2017-12-21T02:00:00","StartDateTimestamp":1513814400,"EndDateTimestamp":1513818000,"DeviceSetupID":3}, {"MeasurementID":52071,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T02:00:00","EndDate":"2017-12-21T03:00:00","StartDateTimestamp":1513818000,"EndDateTimestamp":1513821600,"DeviceSetupID":3}, {"MeasurementID":52073,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T03:00:00","EndDate":"2017-12-21T04:00:00","StartDateTimestamp":1513821600,"EndDateTimestamp":1513825200,"DeviceSetupID":3}, {"MeasurementID":52075,"**Value":0.0**,"EstimatedCost":null,"StartDate":"2017-12-21T04:00:00","EndDate":"2017-12-21T05:00:00","StartDateTimestamp":1513825200,"EndDateTimestamp":1513828800,"DeviceSetupID":3}, {"MeasurementID":52077,"**Value":0.0,"**EstimatedCost":null,"StartDate":"2017-12-21T05:00:00","EndDate":"2017-12-21T06:00:00","StartDateTimestamp":1513828800,"EndDateTimestamp":1513832400,"DeviceSetupID":3}, {"MeasurementID":52079,"**Value":5285.0**,"EstimatedCost":null,"StartDate":"2017-12-21T06:00:00","EndDate":"2017-12-21T07:00:00","StartDateTimestamp":1513832400,"EndDateTimestamp":1513836000,"DeviceSetupID":3}, {"MeasurementID":52081,"**Value":12573.0**,"EstimatedCost":null,"StartDate":"2017-12-21T07:00:00","EndDate":"2017-12-21T08:00:00","StartDateTimestamp":1513836000,"EndDateTimestamp":1513839600,"DeviceSetupID":3}, {"MeasurementID":52083,"**Value":12239.0**,"EstimatedCost":null,"StartDate":"2017-12-21T08:00:00","EndDate":"2017-12-21T09:00:00","StartDateTimestamp":1513839600,"EndDateTimestamp":1513843200,"DeviceSetupID":3}, {"MeasurementID":52085,"**Value":12872.0**,"EstimatedCost":null,"StartDate":"2017-12-21T09:00:00","EndDate":"2017-12-21T10:00:00","StartDateTimestamp":1513843200,"EndDateTimestamp":1513846800,"DeviceSetupID":3}, {"MeasurementID":52087,"**Value":12657.0**,"EstimatedCost":null,"StartDate":"2017-12-21T10:00:00","EndDate":"2017-12-21T11:00:00","StartDateTimestamp":1513846800,"EndDateTimestamp":1513850400,"DeviceSetupID":3},{"MeasurementID":52089,"**Value":22.6666666666667**,"EstimatedCost":null,"StartDate":"2017-12-21T11:00:00","EndDate":"2017-12-21T12:00:00","StartDateTimestamp":1513850400,"EndDateTimestamp":1513854000,"DeviceSetupID":3}, {"MeasurementID":52091,"**Value":21.0833333333333**,"EstimatedCost":null,"StartDate":"2017-12-21T12:00:00","EndDate":"2017-12-21T13:00:00","StartDateTimestamp":1513854000,"EndDateTimestamp":1513857600,"DeviceSetupID":3}, {"MeasurementID":52093,"**Value":23.0**,"EstimatedCost":null,"StartDate":"2017-12-21T13:00:00","EndDate":"2017-12-21T14:00:00","StartDateTimestamp":1513857600,"EndDateTimestamp":1513861200,"DeviceSetupID":3}]

6.2.1.7 Module 7: GIS Repo Services

The GIS Repo provides the link between the alphanumeric data (consumption and temperature) with the geometric representation of the information. This repository provides access to both geometry (GetMap) and information (GetFeatureInfo) services.

The following table shows the answers to the queries that provide the geometry of the buildings of the study area (Coronation-Vitoria) and the alphanumeric information of the building used throughout this integration test.

Query / Transaction	Result
GetMap http://geoservergis.azurewebsites.net/geoserver/wms?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&FORMAT=image%2Fpng&TRANSPARENT=true&LAYERS=smartencity%3Acoronacionbld&STYLES=buildings_temperature&CRS=EPSG%3A3857&WIDTH=1904&HEIGHT=990&BBOX=-299169.75196034467%2C5288913.02456848%2C-296895.7503688607%2C5290095.409849768	
GetFeatureInfo http://geoservergis.azurewebsites.net/geoserver/wms?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetFeatureInfo&FORMAT=image%2Fpng&TRANSPARENT=true&QUERY_LAYERS=smartencity%3Acoronacionbld&LAYERS=smartencity%3Acoronacionbld&STYLES=buildings_temperature&INFO_FORMAT=application%2Fjson&I=50&J=50&CRS=EPSG%3A3857&WIDTH=101&HEIGHT=101&BBOX=-297701.92215155583%2C5289423.00286657%2C-297581.2949662933%2C5289543.630051833	{"type": "FeatureCollection","totalFeatures": "unknown","features": [{"type": "Feature","id": "coronacionbld.590472000001","geometry": {"type": "MultiPolygon","coordinates": [[[[526656.881,4744416.295],[526652.79,4744419.171],[526656.816,4744424.898],[526621.392,4744449.798],[526629.443,4744461.252],[526625.353,4744464.127],[526628.805,4744469.037],[526632.895,4744466.162],[526638.07,4744473.524],[526644.86,4744468.751],[526646.009,4744470.386],[526655.008,4744464.06],[526653.859,4744462.426],[526673.494,4744448.624],[526674.312,4744448.049],[526673.161,4744446.413],[526682.979,4744439.512],[526681.254,4744437.058],[526667.451,4744417.422],[526664.002,4744412.516],[526663.425,4744411.695],[526656.881,4744416.295]]]]},"geometry_name": "the_geom","properties": {"CODPOLI": 59,"CODPAR": 472,"CODEDI": 1,"CODIGO": "590472000001","NOMBRE": "PORTAL DE ARRIAGA","PL": "1A","BARRIOTEXT": "CORONACION","DISTRITO": 1,"VIVIENDAS": 0,"ALTURA": 6,"CONSTRU": 1998,"**T_AVG": 21.166**,"DATE_T_AVG": "2017-12-21T00:00:00Z","**ENERGY_AVG": 990.247**,"DATE_ENERGY_AVG": "2017-12-21T00:00:00Z"}}],"crs": {"type": "name","properties": {"name": "urn:ogc:def:crs:EPSG::25830"}}}

6.2.1.8 Module 8: Energy Application

The final application of this integration scenario is a GIS application that takes the data from the GIS repo. Results are shown to the user through a GIS application (see Figure 13) and a Building Report (see Figure 14).

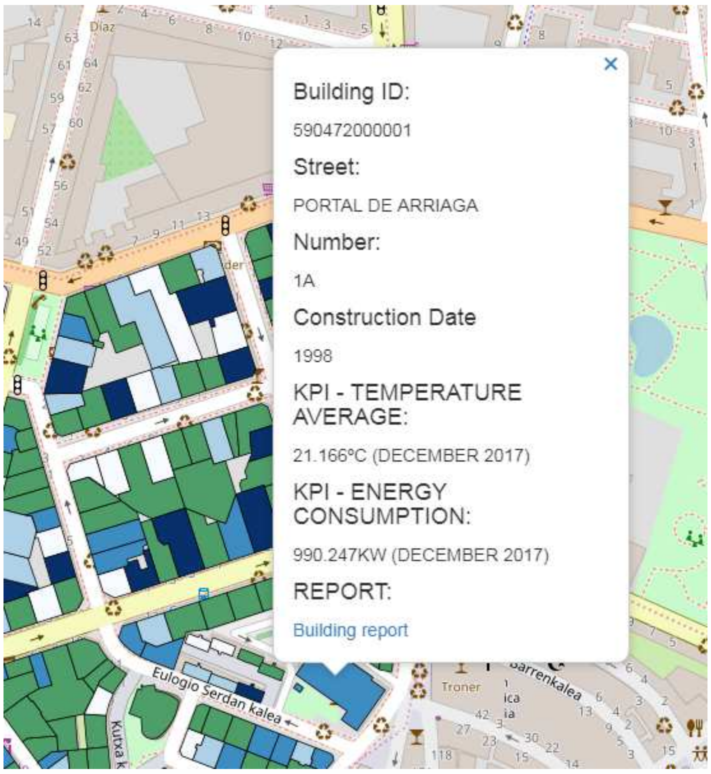
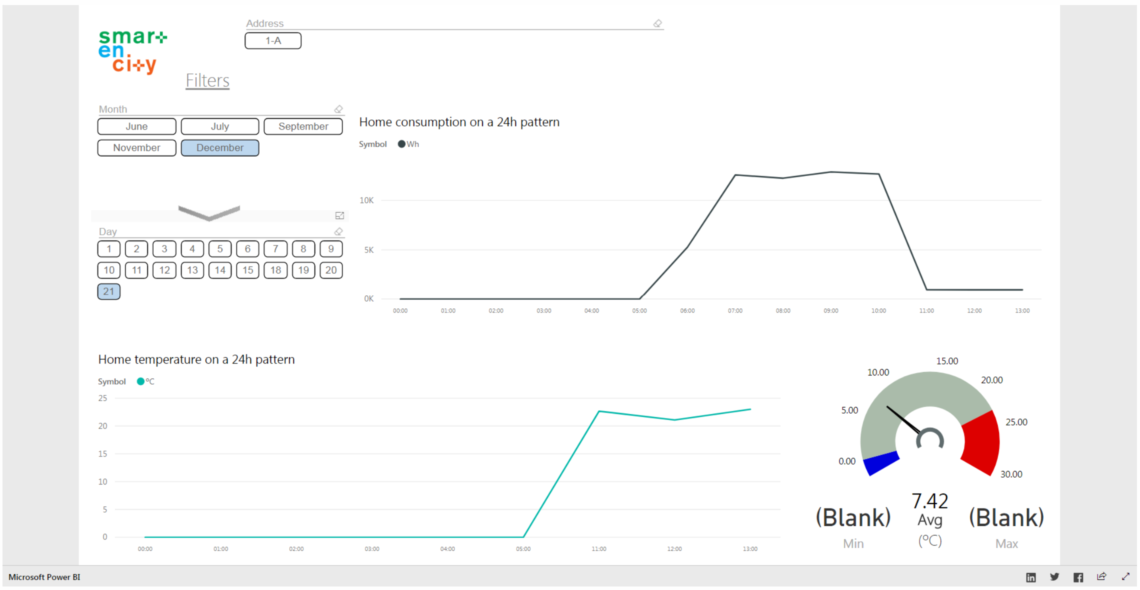

Building ID:
590472000001
Street:
PORTAL DE ARRIAGA
Number:
1A
Construction Date
1998
KPI - TEMPERATURE AVERAGE:
21.166°C (DECEMBER 2017)
KPI - ENERGY CONSUMPTION:
990.247KW (DECEMBER 2017)
REPORT:
Building report

Figure 13 GIS Interface of the Energy Application


Address: 1-A

Filters

Month: June, July, September, November, [✓ December]

Day: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 19, 20, [✓ 21]

Figure 14 Building Report in the Energy Application

6.2.2 Integration test 2: District Heating (Fortum)

As the purpose of integration testing is to validate that individual software modules work as a complete system. As Tartu's software and hardware vendors have not been selected we can only present a sample scenario in this section.

Scenario Id:	DISTRICT_HEATING_01
Scenario Name:	Heating data integration
Scenario Description	The District Heating demonstrator implements the project Reference Architecture in the lighthouse of Tartu. The scenario consists in testing the (1) acquisition of building heating data coming from district heating provider - Fortum, (2) the transformation of this real-time data into a historical repository, (3) the publication of APIs in the interoperability layer capable of querying data from the historical repository

Test Environment		
Hardware	**Software**	**Network Config.**
Smart IoT hub and temperature sensor	Fortum back-end database Heating data acquisition agent Cumulocity IoT platform	Not applicable

Acceptance Criteria	
Criterion	**Value**
Fetch Building data from Fortum backend and push it to Cumulocity Push temperature data from IoT hub	Building and temperature data available on Cumulocity platform

List of Modules Involved				
#	**Module**	**Technology**	**Query / Transaction**	**Result**
	Data acquisition agent	REST	GET https://fortum.server/xmlliveread/get-data.php?address=BuildingAddress POST https://cumulocity.server/measurement/measurements	HTTP/1.1 200 OK HTTP/1.1 201 Created
	Smart IoT hub	REST	POST https://cumulocity.server/measurement/measurements	HTTP/1.1 201 Created

6.2.3 Integration test 3: Electricity Production from Solar Panels

Scenario Id:	SOLAR_ELECTR_PROD_01
Scenario Name:	Solar Electricity Production demonstrator
Scenario Description	Electricity Production from solar panels demonstrator implements the Reference Architecture design defined in this project and makes tests on

	real-time data. Following scenario consists of several steps: 1) acquisition of electricity data from meter installed in the building (acquisition layer), 2) publishing this data to real-time repository and separate web application (knowledge layer), 3) aggregation of data 4) presentation of data in web application through REST API (interoperability layer).

Test Environment		
Hardware	**Software**	**Network Config.**
Energy consumption meter Datalogger	Chrome/Firefox web browser RESTful clients to test the endpoints	Demonstrator access via HTTP

Acceptance Criteria	
Criterion	**Value**
Response time	Below a given value
Failure response	Error notifications
Concurrency load response	Minimum number of queries/users
Awareness to input data changes	Smoothness / Page reload / Asynchronous

List of Modules Involved		
#	**Module**	**Technology**
1	Data acquisition of electricity production	GSM/GPRS (V/PL)
2	Publishing data	AMQP (RabbitMQ)
3	ETL process (store and transform data)	ETL mechanisms for aggregating data
4	API for Electricity Production Services	Secured API to provide data to the Intelligent Services layer Data processed by the Knowledge layer is offered by RESTful API
5	Presentation of data in web application	Web application Leverages the Interoperability layer API to provide the user experience

The list of modules involved in the completion of this test scenario are depictured in Figure 15

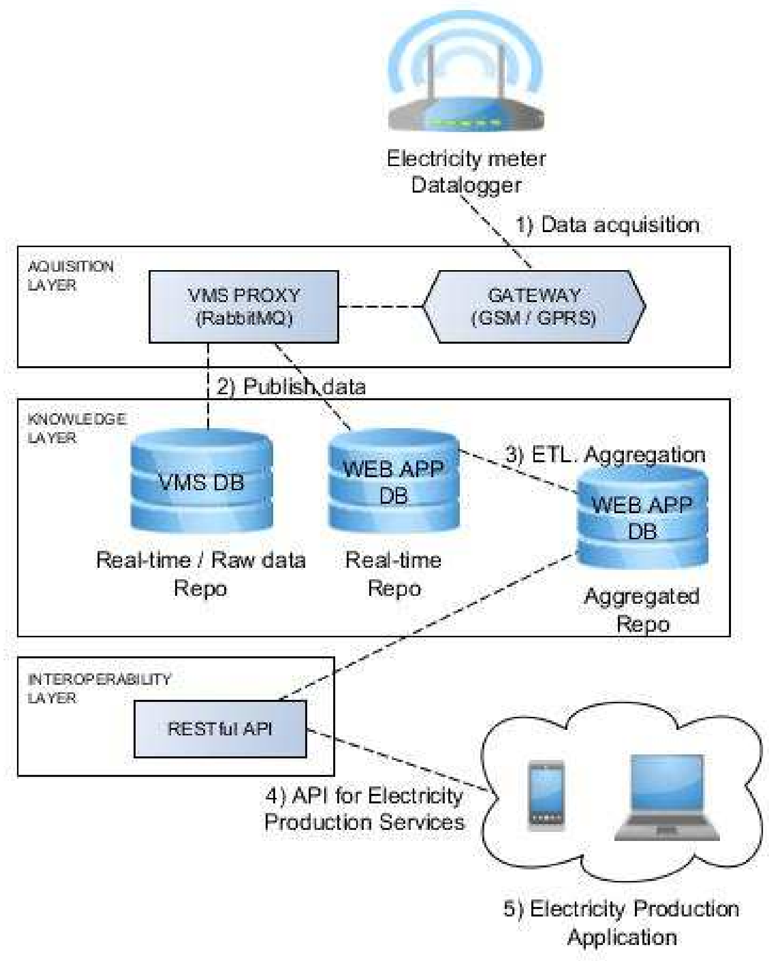
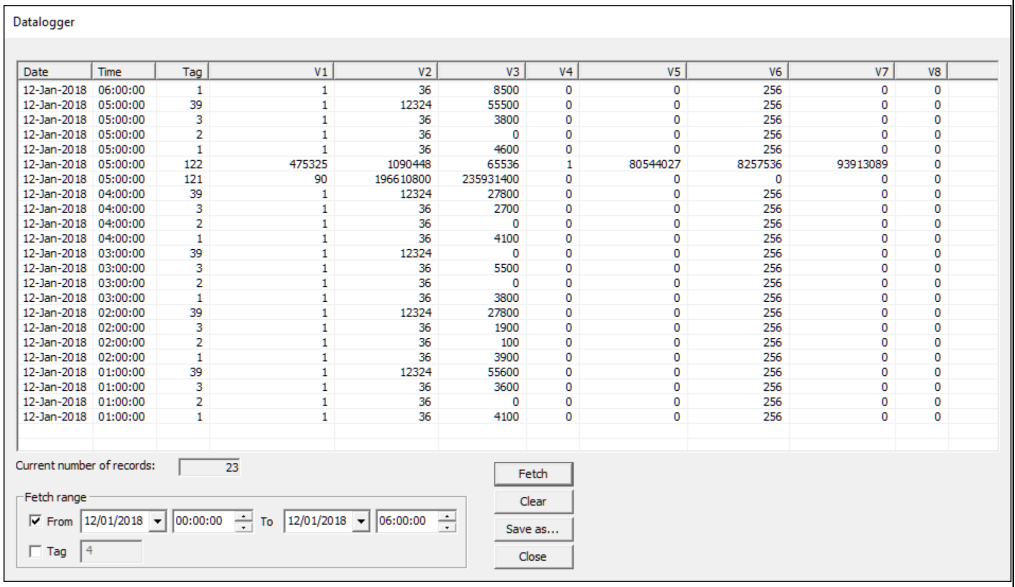


Figure 15 Electricity Production from Solar Panels test scenario

6.2.3.1 *Module 1: Data acquisition from device*

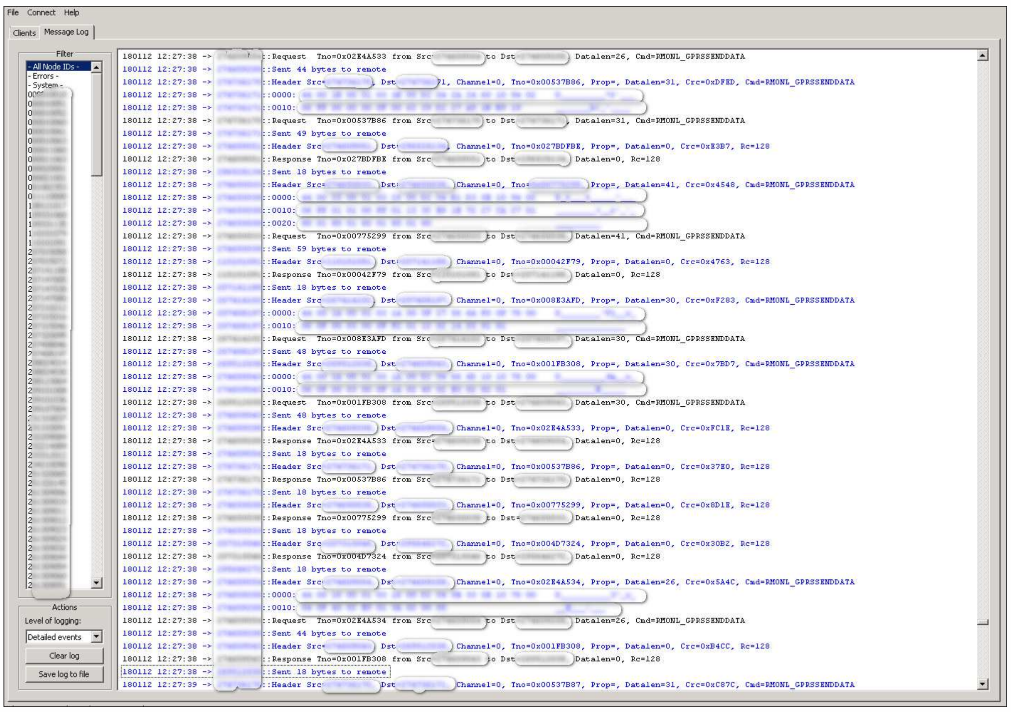
Configuration	Result
Device datalogger shows real data coming from metering device.	

Datalogger

Date	Time	Tag	V1	V2	V3	V4	V5	V6	V7	V8
12-Jan-2018	06:00:00	1	1	36	8500	0	0	256	0	0
12-Jan-2018	05:00:00	39	1	12324	55500	0	0	256	0	0
12-Jan-2018	05:00:00	3	1	36	3800	0	0	256	0	0
12-Jan-2018	05:00:00	2	1	36	0	0	0	256	0	0
12-Jan-2018	05:00:00	1	1	36	4600	0	0	256	0	0
12-Jan-2018	05:00:00	122	475325	1090448	65536	1	80544027	8257536	93913089	0
12-Jan-2018	05:00:00	121	90	196610800	235931400	0	0	0	0	0
12-Jan-2018	04:00:00	39	1	12324	27800	0	0	256	0	0
12-Jan-2018	04:00:00	3	1	36	2700	0	0	256	0	0
12-Jan-2018	04:00:00	2	1	36	0	0	0	256	0	0
12-Jan-2018	04:00:00	1	1	36	4100	0	0	256	0	0
12-Jan-2018	03:00:00	39	1	12324	0	0	0	256	0	0
12-Jan-2018	03:00:00	3	1	36	5500	0	0	256	0	0
12-Jan-2018	03:00:00	2	1	36	0	0	0	256	0	0
12-Jan-2018	03:00:00	1	1	36	3800	0	0	256	0	0
12-Jan-2018	02:00:00	39	1	12324	27800	0	0	256	0	0
12-Jan-2018	02:00:00	3	1	36	1900	0	0	256	0	0
12-Jan-2018	02:00:00	2	1	36	100	0	0	256	0	0
12-Jan-2018	02:00:00	1	1	36	3900	0	0	256	0	0
12-Jan-2018	01:00:00	39	1	12324	55600	0	0	256	0	0
12-Jan-2018	01:00:00	3	1	36	3600	0	0	256	0	0
12-Jan-2018	01:00:00	2	1	36	0	0	0	256	0	0
12-Jan-2018	01:00:00	1	1	36	4100	0	0	256	0	0

Current number of records: 23

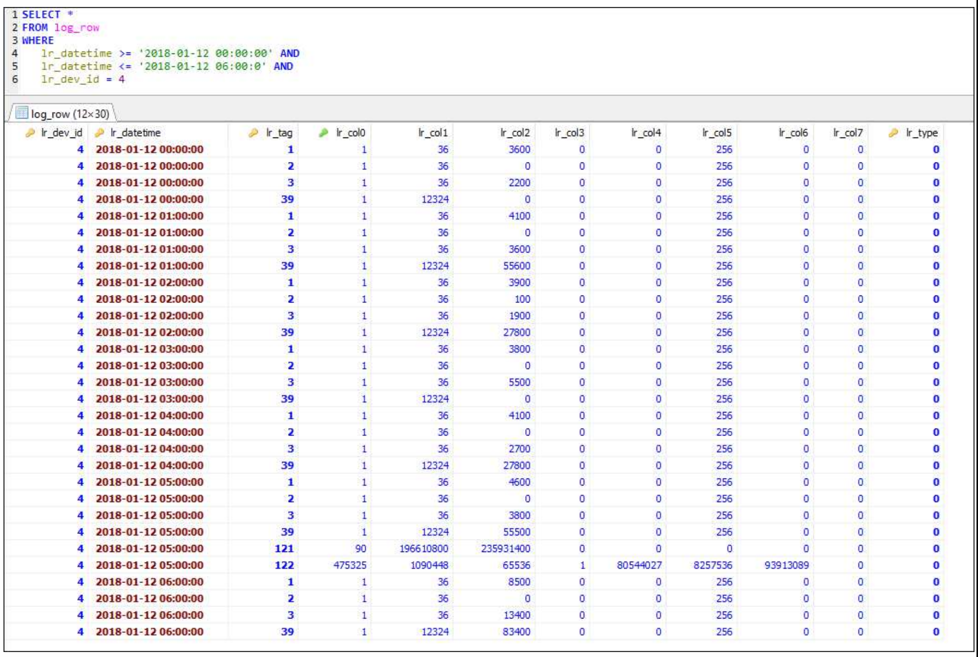
Fetch range
[✓] From 12/01/2018 00:00:00 To 12/01/2018 06:00:00
[ ] Tag 4

Gateway log message window shows data coming from this device (GSM/GPRS).	

Level of logging: Detailed events

```
180112 12:27:38 -> [redacted]::Request  Tno=0x02E4A533 from Src[redacted] to Dst[redacted] Datalen=26, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::Sent 44 bytes to remote
180112 12:27:38 -> [redacted]::Header Src=[redacted], Dst[redacted], Channel=0, Tno=0x00537B86, Prop=, Datalen=31, Crc=0xDFED, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::0000:[redacted]
180112 12:27:38 -> [redacted]::0010:[redacted]
180112 12:27:38 -> [redacted]::Request  Tno=0x00537B86 from Src[redacted] to Dst[redacted], Datalen=31, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::Sent 49 bytes to remote
180112 12:27:38 -> [redacted]::Header Src[redacted] Dst[redacted], Channel=0, Tno=0x027BDFBE, Prop=, Datalen=0, Crc=0xE3B7, Rc=128
180112 12:27:38 -> [redacted]::Response Tno=0x027BDFBE from Src[redacted] to Dst[redacted] Datalen=0, Rc=128
180112 12:27:38 -> [redacted]::Sent 18 bytes to remote
180112 12:27:38 -> [redacted]::Header Src=[redacted] Dst:[redacted] Channel=0, Tno=[redacted] Prop=, Datalen=41, Crc=0x4548, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::0000:[redacted]
180112 12:27:38 -> [redacted]::0010:[redacted]
180112 12:27:38 -> [redacted]::0020:[redacted]
180112 12:27:38 -> [redacted]::Request  Tno=0x00775299 from Src[redacted] to Dst[redacted] Datalen=41, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::Sent 59 bytes to remote
180112 12:27:38 -> [redacted]::Header Src[redacted] Dst[redacted] Channel=0, Tno=0x00042F79, Prop=, Datalen=0, Crc=0x4763, Rc=128
180112 12:27:38 -> [redacted]::Response Tno=0x00042F79 from Src[redacted] to Dst[redacted] Datalen=0, Rc=128
180112 12:27:38 -> [redacted]::Sent 18 bytes to remote
180112 12:27:38 -> [redacted]::Header Src[redacted], Dst[redacted] Channel=0, Tno=0x008E3AFD, Prop=, Datalen=30, Crc=0xF283, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::0000:[redacted]
180112 12:27:38 -> [redacted]::0010:[redacted]
180112 12:27:38 -> [redacted]::Request  Tno=0x008E3AFD from Src[redacted] to Dst[redacted] Datalen=30, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::Sent 48 bytes to remote
180112 12:27:38 -> [redacted]::Header Src[redacted] Dst[redacted] Channel=0, Tno=0x001FB308, Prop=, Datalen=30, Crc=0x7BD7, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::0000:[redacted]
180112 12:27:38 -> [redacted]::0010:[redacted]
180112 12:27:38 -> [redacted]::Request  Tno=0x001FB308 from Src[redacted] to Dst[redacted] Datalen=30, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::Sent 48 bytes to remote
180112 12:27:38 -> [redacted]::Header Src[redacted] Dst[redacted] Channel=0, Tno=0x02E4A533, Prop=, Datalen=0, Crc=0xFC1E, Rc=128
180112 12:27:38 -> [redacted]::Response Tno=0x02E4A533 from Src=[redacted] to Dst[redacted] Datalen=0, Rc=128
180112 12:27:38 -> [redacted]::Sent 18 bytes to remote
180112 12:27:38 -> [redacted]::Header Src[redacted] Dst[redacted] Channel=0, Tno=0x00537B86, Prop=, Datalen=0, Crc=0x37E0, Rc=128
180112 12:27:38 -> [redacted]::Response Tno=0x00537B86 from Src[redacted] to Dst[redacted] Datalen=0, Rc=128
180112 12:27:38 -> [redacted]::Sent 18 bytes to remote
180112 12:27:38 -> [redacted]::Header Src[redacted] Dst[redacted] Channel=0, Tno=0x00775299, Prop=, Datalen=0, Crc=0x8D1E, Rc=128
180112 12:27:38 -> [redacted]::Response Tno=0x00775299 from Src[redacted] to Dst=[redacted] Datalen=0, Rc=128
180112 12:27:38 -> [redacted]::Sent 18 bytes to remote
180112 12:27:38 -> [redacted]::Header Src[redacted] Dst[redacted] Channel=0, Tno=0x004D7324, Prop=, Datalen=0, Crc=0x30B2, Rc=128
180112 12:27:38 -> [redacted]::Response Tno=0x004D7324 from Src[redacted] to Dst[redacted] Datalen=0, Rc=128
180112 12:27:38 -> [redacted]::Sent 18 bytes to remote
180112 12:27:38 -> [redacted]::Header Src[redacted] Ds[redacted] Channel=0, Tno=0x02E4A534, Prop=, Datalen=26, Crc=0x5A4C, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::0000:[redacted]
180112 12:27:38 -> [redacted]::0010:[redacted]
180112 12:27:38 -> [redacted]::Request  Tno=0x02E4A534 from Src[redacted] to Dst[redacted] Datalen=26, Cmd=RMONL_GPRSSENDDATA
180112 12:27:38 -> [redacted]::Sent 44 bytes to remote
180112 12:27:38 -> [redacted]::Header Src=[redacted] Dst[redacted] Channel=0, Tno=0x001FB308, Prop=, Datalen=0, Crc=0xB4CC, Rc=128
180112 12:27:38 -> [redacted]::Response Tno=0x001FB308 from Src[redacted] to Dst[redacted] Datalen=0, Rc=128
180112 12:27:38 -> [redacted]::Sent 18 bytes to remote
180112 12:27:39 -> [redacted]::Header Src[redacted] Dst[redacted] Channel=0, Tno=0x00537B87, Prop=, Datalen=31, Crc=0xC87C, Cmd=RMONL_GPRSSENDDATA
```

6.2.3.2 Module 2: Publishing Data

Query / Transaction	**Result**
VMS Proxy publishes data to VMS, data is saved in database. Select * from log_row where lr_datetime >= '2018-01-12 00:00:00' and lr_datetime <= '2018-01-12 06:00:00' and lr_dev_id = 4	

```
SELECT *
FROM log_row
WHERE
    lr_datetime >= '2018-01-12 00:00:00' AND
    lr_datetime <= '2018-01-12 06:00:0' AND
    lr_dev_id = 4
```

log_row (12×30)

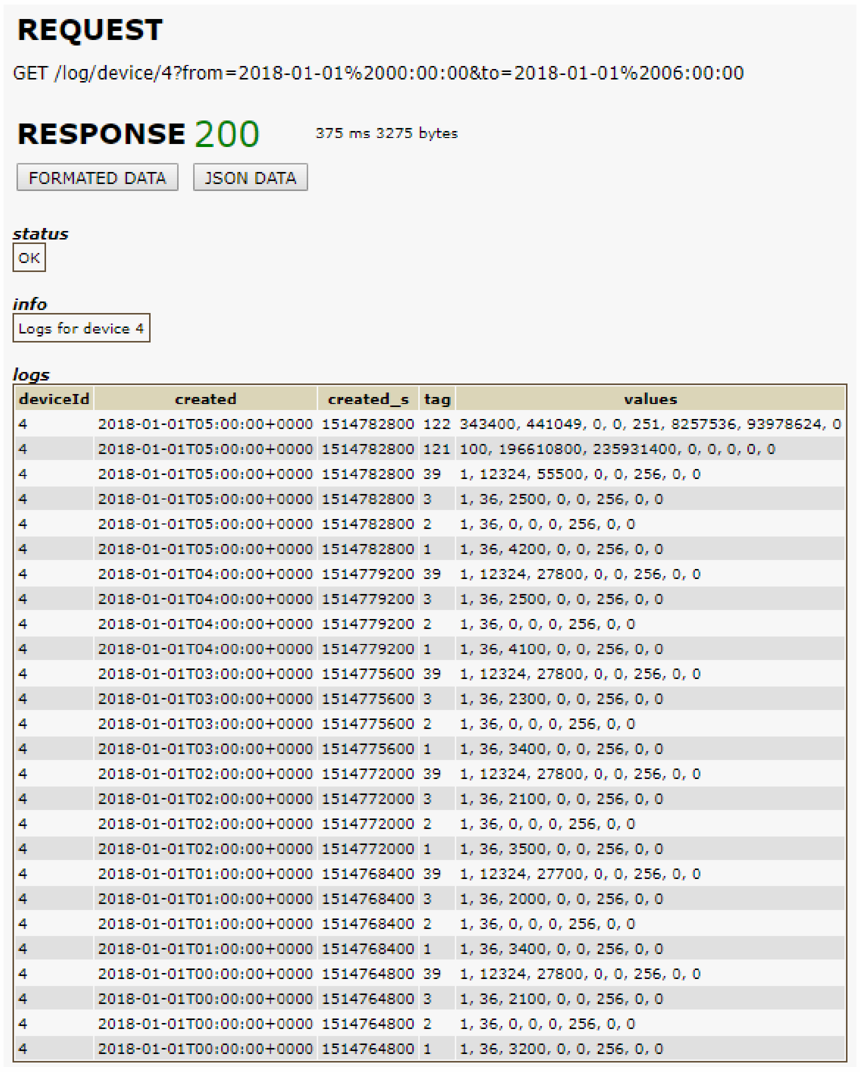
lr_dev_id	lr_datetime	lr_tag	lr_col0	lr_col1	lr_col2	lr_col3	lr_col4	lr_col5	lr_col6	lr_col7	lr_type
4	2018-01-12 00:00:00	1	1	36	3600	0	0	256	0	0	0
4	2018-01-12 00:00:00	2	1	36	0	0	0	256	0	0	0
4	2018-01-12 00:00:00	3	1	36	2200	0	0	256	0	0	0
4	2018-01-12 00:00:00	39	1	12324	0	0	0	256	0	0	0
4	2018-01-12 01:00:00	1	1	36	4100	0	0	256	0	0	0
4	2018-01-12 01:00:00	2	1	36	0	0	0	256	0	0	0
4	2018-01-12 01:00:00	3	1	36	3600	0	0	256	0	0	0
4	2018-01-12 01:00:00	39	1	12324	55600	0	0	256	0	0	0
4	2018-01-12 02:00:00	1	1	36	3900	0	0	256	0	0	0
4	2018-01-12 02:00:00	2	1	36	100	0	0	256	0	0	0
4	2018-01-12 02:00:00	3	1	36	1900	0	0	256	0	0	0
4	2018-01-12 02:00:00	39	1	12324	27800	0	0	256	0	0	0
4	2018-01-12 03:00:00	1	1	36	3800	0	0	256	0	0	0
4	2018-01-12 03:00:00	2	1	36	0	0	0	256	0	0	0
4	2018-01-12 03:00:00	3	1	36	5500	0	0	256	0	0	0
4	2018-01-12 03:00:00	39	1	12324	0	0	0	256	0	0	0
4	2018-01-12 04:00:00	1	1	36	4100	0	0	256	0	0	0
4	2018-01-12 04:00:00	2	1	36	0	0	0	256	0	0	0
4	2018-01-12 04:00:00	3	1	36	2700	0	0	256	0	0	0
4	2018-01-12 04:00:00	39	1	12324	27800	0	0	256	0	0	0
4	2018-01-12 05:00:00	1	1	36	4600	0	0	256	0	0	0
4	2018-01-12 05:00:00	2	1	36	0	0	0	256	0	0	0
4	2018-01-12 05:00:00	3	1	36	3800	0	0	256	0	0	0
4	2018-01-12 05:00:00	39	1	12324	55500	0	0	256	0	0	0
4	2018-01-12 05:00:00	121	90	196610800	235931400	0	0	0	0	0	0
4	2018-01-12 05:00:00	122	475325	1090448	65536	1	80544027	8257536	93913089	0	0
4	2018-01-12 06:00:00	1	1	36	8500	0	0	256	0	0	0
4	2018-01-12 06:00:00	2	1	36	0	0	0	256	0	0	0
4	2018-01-12 06:00:00	3	1	36	13400	0	0	256	0	0	0
4	2018-01-12 06:00:00	39	1	12324	83400	0	0	256	0	0	0

6.2.3.3 Module 3: ETL process (store and transform data)

Query / Transaction	**Result**
insert into log_row_week select lr_dev_id,WEEK(lr_datetime),lr_tag,avg(lr_col0),avg (lr_col1),avg(lr_col2),avg(lr_col3),avg(lr_col4),avg(lr	Status OK. 168 rows inserted.

_col5),avg(lr_col6),avg(lr_col7),lr_type from log_row where lr_datetime >= '2018-01-01 00:00:00' and lr_datetime <= '2018-01-07 23:59:59' group by lr_dev_id,WEEK(lr_datetime);	

6.2.3.4 Module 4: API for Electricity Production Services

Query / Transaction	Result
GET/log/device/4?from=2018-01-01%2000:00:00&to=2018-01-01%2006:00:00	

REQUEST

GET /log/device/4?from=2018-01-01%2000:00:00&to=2018-01-01%2006:00:00

RESPONSE 200 375 ms 3275 bytes

status

OK

info

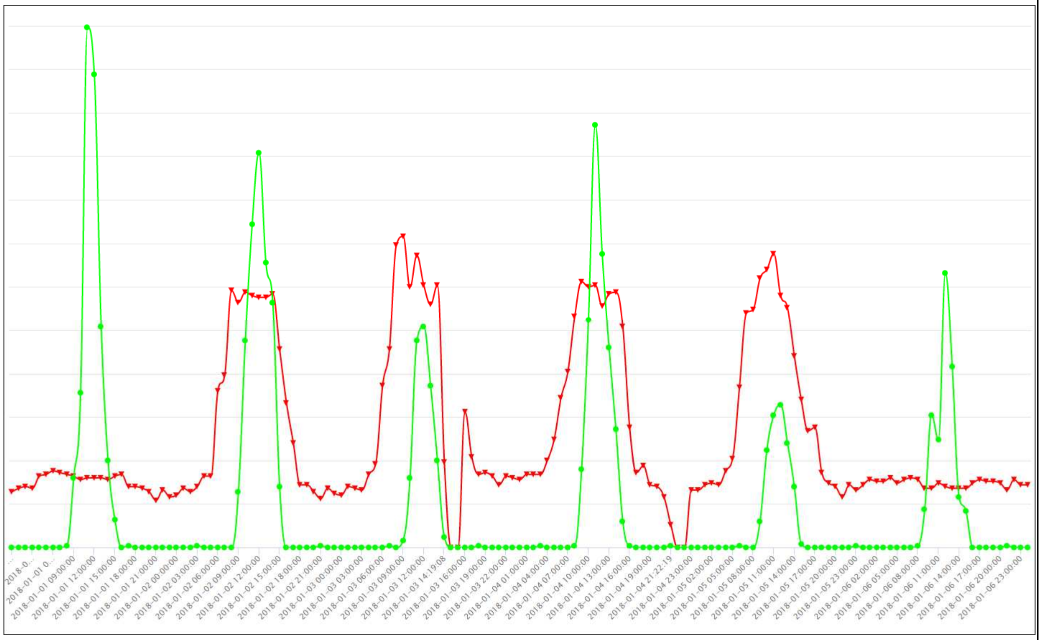
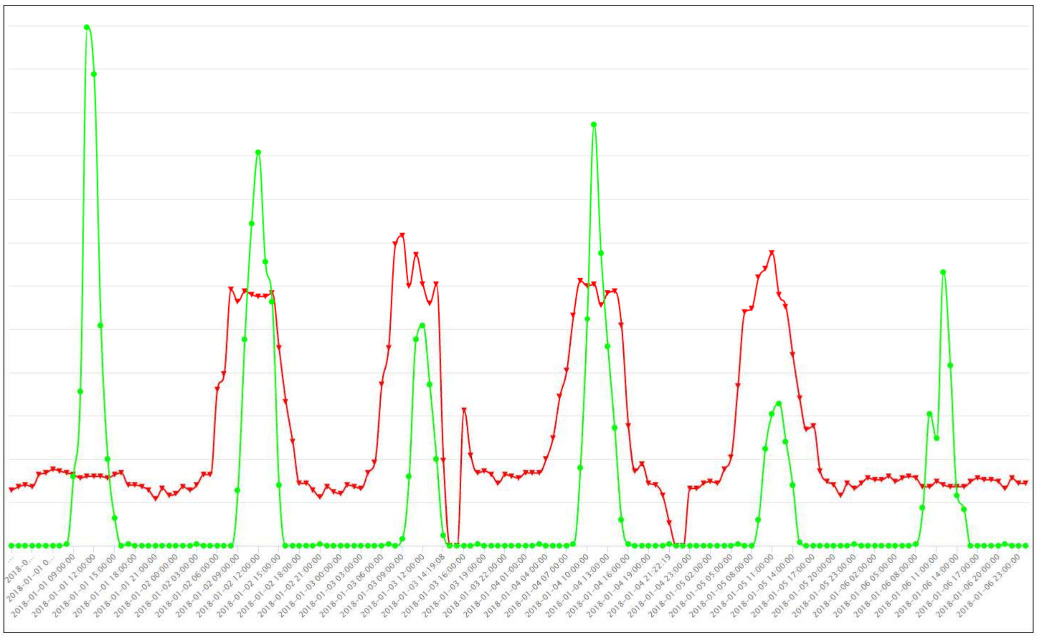
Logs for device 4

logs

deviceId	created	created_s	tag	values
4	2018-01-01T05:00:00+0000	1514782800	122	343400, 441049, 0, 0, 251, 8257536, 93978624, 0
4	2018-01-01T05:00:00+0000	1514782800	121	100, 196610800, 235931400, 0, 0, 0, 0, 0
4	2018-01-01T05:00:00+0000	1514782800	39	1, 12324, 55500, 0, 0, 256, 0, 0
4	2018-01-01T05:00:00+0000	1514782800	3	1, 36, 2500, 0, 0, 256, 0, 0
4	2018-01-01T05:00:00+0000	1514782800	2	1, 36, 0, 0, 0, 256, 0, 0
4	2018-01-01T05:00:00+0000	1514782800	1	1, 36, 4200, 0, 0, 256, 0, 0
4	2018-01-01T04:00:00+0000	1514779200	39	1, 12324, 27800, 0, 0, 256, 0, 0
4	2018-01-01T04:00:00+0000	1514779200	3	1, 36, 2500, 0, 0, 256, 0, 0
4	2018-01-01T04:00:00+0000	1514779200	2	1, 36, 0, 0, 0, 256, 0, 0
4	2018-01-01T04:00:00+0000	1514779200	1	1, 36, 4100, 0, 0, 256, 0, 0
4	2018-01-01T03:00:00+0000	1514775600	39	1, 12324, 27800, 0, 0, 256, 0, 0
4	2018-01-01T03:00:00+0000	1514775600	3	1, 36, 2300, 0, 0, 256, 0, 0
4	2018-01-01T03:00:00+0000	1514775600	2	1, 36, 0, 0, 0, 256, 0, 0
4	2018-01-01T03:00:00+0000	1514775600	1	1, 36, 3400, 0, 0, 256, 0, 0
4	2018-01-01T02:00:00+0000	1514772000	39	1, 12324, 27800, 0, 0, 256, 0, 0
4	2018-01-01T02:00:00+0000	1514772000	3	1, 36, 2100, 0, 0, 256, 0, 0
4	2018-01-01T02:00:00+0000	1514772000	2	1, 36, 0, 0, 0, 256, 0, 0
4	2018-01-01T02:00:00+0000	1514772000	1	1, 36, 3500, 0, 0, 256, 0, 0
4	2018-01-01T01:00:00+0000	1514768400	39	1, 12324, 27700, 0, 0, 256, 0, 0
4	2018-01-01T01:00:00+0000	1514768400	3	1, 36, 2000, 0, 0, 256, 0, 0
4	2018-01-01T01:00:00+0000	1514768400	2	1, 36, 0, 0, 0, 256, 0, 0
4	2018-01-01T01:00:00+0000	1514768400	1	1, 36, 3400, 0, 0, 256, 0, 0
4	2018-01-01T00:00:00+0000	1514764800	39	1, 12324, 27800, 0, 0, 256, 0, 0
4	2018-01-01T00:00:00+0000	1514764800	3	1, 36, 2100, 0, 0, 256, 0, 0
4	2018-01-01T00:00:00+0000	1514764800	2	1, 36, 0, 0, 0, 256, 0, 0
4	2018-01-01T00:00:00+0000	1514764800	1	1, 36, 3200, 0, 0, 256, 0, 0

6.2.3.5 Module 5: Presentation of data in web application

User view	Result

User graph showing weekly data from period 2018-01-01 00:00:00-2018-01-01 06:00:00	
User graph showing daily data from period 2018-01-01 00:00:00-2018-01-01 06:00:00	

7 Report on tests & Correction measures

This section provides the software test reports for the integration and validation task (T6.7) of the SmartEnCity ICT platform modules. It contains the results of the unit tests (section 6.1) and integration tests (section 6.2), which were executed during the testing phase. The information provided in the reports comprises a qualitative assessment of each test and related correction measures, which deliver further details on the assessment, e.g. discussions on performance values, along with the identification of issues and plans for their resolution before the production or operation phase starts in each lighthouse implementation.

7.1 Unit Tests

The table below provides the test summary report for the unit tests deployed in section 6.1.

Test Summary Report		
Module Name	**Qualitative Assessment**	**Correction measures**
Data Acquisition about Energy	The performance tests in terms of time response and concurrency passed according to the acceptance criteria defined. The functional tests in data acquisition about energy showed a consistent output for sensor activation, sensor provisioning and sensor measurement. In this case, and applicable for all other tests, the response time depends on the bandwidth and the speed of the connections.	The data acquisition system offers a response time of 110 ms. per measurement, which means it is proportional to the number of measurements. On the other hand, the response time under concurrency conditions is well handled by the module, since it stays below 20 seconds for storing 1000 sensor measurements read concurrently. According to these results, it can be concluded that scalability issues will not be anticipated at production stage.
ETL Processes (Energy Vertical Repository)	The ETL process, which aggregates the real-time data and stores it in the vertical repository, was tested to verify that the data transformation was correctly performed. The ETL scripts showed that the response time and the data load tests passed according to the acceptance criteria defined.	This process aggregates the real-time data in the Energy Vertical Repo, allowing the energy applications to easily display different data perspectives or online analytical processing (OLAP) cubes to the user, delegating the processing and data transformation to the ETL process. In order to obtain a good throughput, the ETL process performance, in terms of response time, should be comparable to the data acquisition process to assure a smooth data transfer to the Energy Vertical Repo. In this regard, response time during the tests is 241 milliseconds for processing and storing 1000 sensor measurements into the Energy Vertical Repo. These results can be considered acceptable for assuring a smooth visualization of data from the energy applications
API for Energy Services	The response time and the concurrency load tests in the performance tests fall within the specified acceptance values.	This process lets the energy applications leverage the data stored in the Energy Vertical Repo. The response time of the interface shows that the response time does

	The functional tests proved that the data delivered from the Energy Vertical Repo is consistent and well formed.	not anticipate scalability problems when the number of users increases in production stage.
Energy Application	All GUI tests passed. The navigation is smooth and the look & feel is consistent.	The GUI is not optimized for mobile devices. A responsive web design approach should be adopted in the transition from application prototype (demonstrator) to a full featured product.
API Mobility Services	The functional tests evidenced that the data delivered from the Mobility Vertical Repo is consistent and well formed.	This process lets the mobility applications leverage the data stored in the Mobility Vertical Repo. The response time values of the interface show that scalability problems are not anticipated when the number of users increases in production stage.
API Citizen Engagement Services	The response time and the concurrency load tests in the performance tests fall within the specified acceptance values. The functional tests proved that the data delivered from the Citizen Engagement Vertical Repo is consistent with the repo content and valid.	The response time of the interface shows that it does not anticipate scalability problems when the number of users increases in production stage.
ETL Processes (Historical Repository)	The ETL process, which regularly dumps data from the Realtime Repo to a distributed storage system was tested and verified that the data storage was correctly performed.	The ETL script showed that the response time is below the acceptance criteria defined. However, in this case, differently from the ETL Energy Vertical, a good throughput is not critical, since this process aims to update an analytical database (Historical Repo) and not an operational one, like the Energy Vertical Repo is. While the Realtime Repo is limited in space and the oldest registries are deleted to remain it manageable and fast, the Historical Repo contains the whole data history. This feature allows auditing the sensor data in case there is any disagreement derived from the energy application use and the data is not available in the Realtime Repo.
ETL Processes (Structural Repository)	The ETL process was tested and provided consistent and well-formed results	This ETL extracts data from the dwellings that is to be combined with GIS data and sensor data from the Energy Vertical Repo. The response time (216ms) is similar to the response time under load conditions of the Energy Vertical Repo (216ms), assuring that bottlenecks in the data flow are not anticipated in production mode.
GIS Repo services	The GIS Repo services were tested in response time and data volume load test. While the response time for rendering a unique map data layer falls below the acceptance	This service obtains the building data contained in the GIS repo in terms of geometry associated to the structures and layers. If more than two layers are requested as options for the map display, the rendering

	criteria, the response time for rendering several layers at the same time exceeds the limit under some circumstances that do not affect usability.	time goes above the specification acceptance criteria of 2 seconds. However, the common use cases require one layer each time, which falls below the limits for the acceptance criteria.
GIS Structural Repo services	The GIS Structural Repo services were tested in response time, data volume and concurrency. All the tests results comply with the defined acceptance criteria. The services were also tested according to the functional test definitions. All the outputs from the tests comply with validation and consistency requirements.	This service allows the retrieval of building data and geometry, which is linked to the Structural Repo and Energy Vertical Repo. In this case, a good throughput is not critical, since this process will be updated periodically only when relevant updates in the structural representation of the environment are performed.
API for KPIs	The KPI APIs were tested in terms of response time, data volume load tests and concurrency load tests and fulfill the acceptance criteria. On the other hand, some functional tests were performed to get electricity consumption, production and mobility routes. All the outputs were verified and comply with validation and source consistency requirements.	These APIs are used by the vertical applications to retrieve aggregated and processed data. Time responses assure that data is retrieved at reasonable intervals, creating a good user experience in data monitoring in the final application.
Services for Integrating Open Data	The services for integration open data were tested in terms of response time, data volume load tests and concurrency load tests and fulfill the acceptance criteria. Some functional tests were performed to get electricity prices and CO2 emission levels. All the outputs were verified and comply with validation and source consistency requirements.	These services are used by the vertical applications to process or complete sensor data with additional information, such as energy prices and sustainability indicators. Time responses achieved, like those obtained in the API for KPIs, assure that open data and the KPIs are in synch and refreshed at reasonable time intervals.

Table 31 Unit tests summary report

7.2 Integration Tests

The table below provides the test summary report for the integration tests deployed in section 6.2**¡Error! No se encuentra el origen de la referencia.**.

Test Summary Report		
Integration Test Name	**Qualitative Assessment**	**Correction measures**
Energy Efficiency Test Scenario	All API tests passed in terms of performance and	CIOP in Vitoria-Gasteiz lighthouse is based on Azure technologies. Concerning the API testing, the services that extract temperature and consumption sensor data and render this data into an energy application were leveraged.

	functionality. The GUIs tests offered an average user experience.	In this sense, specific sensor values were tracked from the data acquisition source. This sensor data is stored in the Realtime Repo. From here, an ETL process regularly dumps the content of the Realtime Repo into a distributed database (Historical Repo). In parallel, another ETL process aggregates the real-time data and stores the mean values into the Energy Vertical Repo. On the other hand, the ETL Structural Repo associates the Energy Vertical Repo sensor measurements with the associated dwelling, building and GIS Repo identifier. Furthermore, the GIS Structural Repo Service links the Structural Repo data with cartographic and building information, such as the address, zip code, house number, etc. Finally, the API for Energy Services and the GIS Repo Service provide a service façade for the Energy Application to render the data. Summarizing, the tracking of the sensor data from the physical devices through the structural and GIS processes, and its logical representation in the Energy Application was tested. As a result, data consistency was validated. The test demonstrates the integration and correct flow of data between the modules used for the execution of this test. Nevertheless, errors and atypical data have been identified during the implementation of the integration test. Errors in aggregated data have been identified during the testing process. These errors are at different levels in timestamp and element. Regarding the GUI testing and assessment, sensor input data changes were promptly managed by the Energy Application, leading to a smooth web user experience. However, this test scenario shown issues with the legibility of the pop up reports in the main screen and in the rendering of the report visual elements, such as the widgets for selecting the Building ID and the address.
District Heating (Fortum)	As the purpose of integration testing is to validate that individual software modules work as a complete system and at this time Tartu's software and hardware vendors have not been selected – we can only present a sample scenario in this section.	Tartus' base technology for CIOP is Cumulocity. This commercial product provides most of the necessary testing and monitoring capabilities to all of the hardware and software modules that are connected from sensing and/or Intelligent Service layers – e.g. providing continuous point-to-point testing, which limits the need for system wide integration testing. In Cumulocity it is important just once to set correct settings of how frequent check should be performed and testing and monitoring will be executed automatically.
Electricity Production from Solar Panels	All tests passed in terms of performance and functionality.	Sønderborg CIOP solution is based on VMS commercial platform. After setting correctly configuration parameters of connected hardware, the monitoring capabilities and logging information is available which allows to check overall system performance at any given time. Data acquisition from connected dataloggers by Gateway was tested resulting in no unexpected performance. The publishing and storing of collected data from Gateway to

		database(s) were tested which performed as expected. The presentation (plain table and charts) to end user was generated from the collected data resulting in good user experience. No correction measures were required.

Table 32 Integration tests summary report

8 Conclusions, deviations and outputs for other WPs

Deliverable D6.7 presents the results of integration and validation of the different modules of the ICT platform developed in the previous task of WP6.

The V-test model is the reference approach chosen for testing and validation of the CIOP platform. This model is taken as a reference base for testing and quality assurance of software development. The model consists of several steps and tests of which the most representative ones for the type of development carried out in WP6 have been selected. Several standards, tools and templates have been defined and are available for the implementation of the selected approach for the SmartEnCity project.

The adaptation of the selected methodology to the SmartEnCity project considers three types of relevant tests: unit tests, integration test and monitoring test. The first ones are made for each of the modules that make up the software platform separately during the software development phase. The integration tests represent the test of the functionalities of the system and the connection between modules in an integrated manner. The monitoring tests contemplate those aspects to be considered during the operation of the platform that allow controlling the correct operation beyond the development phase. In addition, the components of the platform are heterogeneous in type and form (e.g. repository, process, or service), and each one requires a test plan, although for all of them two basic test types have been defined: Performance test and Functional test. The reference templates have been adapted to the characteristics of the developed ICT platform. The adaptation of the methodology also includes the processes and templates for testing the added-value services identified to be developed in each of the project lighthouses.

The integration of the results of the different demonstrators to be developed in the lighthouses and followers of the project will be carried out through a global access portal that controls the access to the users of the platform and facilitates the visualization and comparison of indicators of the different cities. This system will facilitate also the monitoring of the platform operation once deployed.

Unitary tests have been carried out for some examples of modules or end-points developed in the previous activities of WP6. These end-points represent a representative sample of the modules that make up a SmartEnCity platform. The identified modules are of heterogeneous types and belong to one of the layers of the reference architecture defined for the SmartEnCity project. The sample collects modules of all layers and represents different verticals of application. The results of these validations test the correct functioning and development of the identified modules and set the basis for the testing of new modules to be developed in the project based on the defined reference architecture.

The defined integration tests are complementary in scope, city of reference and base platform. All of them are based on the reference architecture and the layers and modules described in that reference. The integration tests show the integrated functioning of the developed modules and the correct flow of data through prototype applications that include from the capture of information to the presentation of information to the user in a final application.

Concerning the acquisition of data from the physical energy sensors, mobility sensors and citizen mobile devices, the unit test summary reports show conclusively that the performance in terms of time response, data volume and concurrency comply with the acceptance criteria defined. On the other hand, the functional tests in data acquisition showed a consistent data output for sensor activation and data provisioning.

In regard to the ETL processes involving data transformation between databases, the test scripts demonstrated that performance falls within the acceptance criteria limits defined. In terms of data transformation correctness, the test validated the data consistency comparing the output data with the data sources.

The API Services, which leverage the acquisition and data transformation (e.g. aggregation, GIS and structural linkage) were also tested from the performance and functional perspective. These API tests proved high service throughput, data consistency and no scalability problems are foreseen when the number of users increase in production stage.

Three integration scenarios, complementary with regard to SmartEnCity domain and lighthouse implementation, and including some of the end-points mentioned above, were deployed and tested following the integration test requirements. In conclusion, all data acquisition, ETL and database API tests passed in terms of performance and functionality and the GUIs tests offered an average user experience concerning usability, look and feel, smoothness and data flow.

No deviations have been produced according to the dates and content of the deliverable with respect to the proposed plan.

The outputs produced in this deliverable will have effects on activities related with the deployment, integration and validation of the CIOP platform in the three lighthouse cities (Vitoria-Gasteiz WP3, Tartu WP4 and Sonderborg WP5).

9 References

SmartEnCityD6.1. (2016). SmartEnCity Deliverable 6.1: CIOP Functional and Non-Functional Specifications.

SmartEnCityD6.2. (2017). SmartEnCity D6.2 "CIOP architecture generic implementation".

SmartEnCity D6.3 (2017). SmartEnCity D6.3. "Data Model Architecture Implementation".

SmartEnCity D6.4 (2017). SmartEnCity D6.4. "Interoperability mechanisms Implementation".

SmartEnCity D6.5 (2017). SmartEnCity D6.5. "Design guide and tool catalogue".

SmartEnCity D6.6 (2017). SmartEnCity D6.6. "Strategies for added-value services and tool catalogue".