Ref. Ares(2017)3802542 - 28/07/2017

TOWARDS SMART ZERO CO_2 CITIES ACROSS EUROPE

VITORIA-GASTEIZ ✣ TARTU ✣ SØNDERBORG

Deliverable 6.3: Data Model Architecture Implementation.

WP6, Task 6.3

Date of document

31/07/2017 (M18)

Deliverable Version:	D6.3, V1.0
Dissemination Level:	PU[1]
Author(s):	Alain Perez (MU), Enaitz Ezpeleta (MU), Felix Larrinaga (MU), Iñaki Arenaza (MU) Jose Luis Izkara (TEC), Alvaro Arroyo (GIS), Mauri Benedito (GIS), Jose Antonio Sanchez (GIS), Natividad Herrasti (ETIC), Aitor Akizu (ETIC), Josu Rollón (MTEL), Raquel Garcia (ACC), Alvaro Garcia (ACC), Urmo Lehtsalu (ET), Priit Kallas (ET), Jørgen Raun Petersen (VG), Patxi Sáez de Viteri (MON)

[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.	D6.3 Data Model Architecture Implementation.	
Diss. Level	Public	
Deliverable Lead	*MON*	
Status		Working
		Verified by other WPs
	X	Final version
Due date of deliverable	31/07/2017	
Actual submission date	28/07/2017	
Work Package	WP 6 - City Information Open Platform (CIOP)	
WP Lead	*MON*	
Contributing beneficiary(ies)	*TEC, MON, MTEL, ETIC, GIS, VG, ET*	

Date	Version	Person/Partner	Comments
21/03/2017	0.1	Mondragon Unibertsitatea (MGEP-MU)	First Draft for the ToC
07/04/2017	0.2	Alain Perez, Enaitz Ezpeleta, Iñaki Arenaza, Felix Larrinaga MGEP-MU Jose Luis Izkara/TEC Alvaro Arroyo/GIS	Contributions to 4.2.1, 4.2.2, 4.2.3, 4.2.4, 4.2.5 and 5.2.1
27/04/2017	0.3	Alain Perez, Iñaki Arenaza MGEP-MU Natividad Herrasti/ETIC José L. Hernández/CAR	Contributions to 4.2.2, 4.2.6, 4.2.8 (removed) and 5.2.1. Edited figures 3 & 6
29/06/2017	0.4	Felix Larrinaga, Enaitz Ezpeleta MGEP-MU, Mauri Benedito, Jose Antonio Sanchez /GIS	Contribution to 1.3, 2, 3 and 4 (section 4.1). Writing of 5.1, 5.2 (5.2.1 and parts of 5.2.2)
07/07/2017	0.5	Alain Perez, Iñaki Arenaza, Felix Larrinaga MGEP-MU, Raquel Garcia (ACC) Jose Luis Izkara/TEC Mauri Benedito, Jose Antonio Sanchez /GIS	Contribution to 1.2, changes to 4 (section 4.1). Writing of 5.2 (5.2.3 and parts of 5.2.2) and 6
13/07/2017	0.6	Alain Perez, Iñaki Arenaza, Felix Larrinaga MGEP-MU Natividad Herrasti, Aitor Akizu/ETIC Mauri Benedito, Jose Antonio Sanchez /GIS Urmo Lehtsalu, Priit Kallas (ET), Jørgen Raun Petersen (VG),	Contribution to 1.2, changes to 4 (section 4.1.8). Writing of 5.2 (5.2.3 and 5.2.4)
27/07/2017	1.0	Felix Larrinaga (MGEP-MU)	Reviews by TEC, VG and TE. Final version

Copyright notice

© 2016-2021SmartEnCity Consortium Partners. All rights reserved. All contents are reserved by default and may not be disclosed to third parties without the written consent of the SmartEnCity partners, except as mandated by the European Commission contract, for reviewing and dissemination purposes.
All trademarks and other rights on third party products mentioned in this document are acknowledged and owned by the respective holders. The information contained in this document represents the views of SmartEnCity members as of the date they are published. The SmartEnCity consortium does not guarantee that any information contained herein is error-free, or up to date, nor makes warranties, express, implied, or statutory, by publishing this document.

Table of content:

0 Publishable Summary 8
1 Introduction 9
1.1 Purpose and target group 9
1.2 Contributions of partners 10
1.3 Relation to other activities in the project 11
1.4 Reference Architecture and Demonstrator (data models) 11
2 Objectives 13
2.1 Objectives of WP 13
2.2 Objectives of Task 6.3 13
3 Overall Approach 15
4 SmartEnCity CIOP Reference Architecture Data Models 16
4.1 Reference Architecture in SmartEnCity 16
4.2 Data Models 18
4.2.1 Vertical Data Repository 18
4.2.2 KPI Repository 19
4.2.3 Historical Data Repository 19
4.2.4 Structural Data Repository 20
4.2.5 GIS Structural Data Repository 20
4.2.6 GIS Repository 21
4.2.7 Configuration Repository 21
4.2.8 Real Time Repository 21
5 SmartEnCity Demonstrator (Data Model) 23
5.1 Demonstrator design and development 23
5.2 Demonstrator Description 25
5.2.1 Demonstrator data flow 26
5.2.2 Data Models for demonstrator 27
5.2.3 User Guide 51
5.2.4 RA Demonstrator Functionality map 55
6 Conclusions, deviations and outputs for other WPs 56
7 References 57

Table of Tables:

Table 1: Abbreviations and Acronyms 7

Table 2: Contribution of partners 10

Table 3: Relation to other activities in the project 11

Table 4: RA functionality – Platform functionality matching.....55

Table of Figures:

Figure 1: Architecture building approach....15
Figure 2: Smart Cities General Architecture 16
Figure 3: Data Models for CIOP....18
Figure 4 Data Process lifecycle....24
Figure 5: Architecture of the demonstrator. 26
Figure 6: Related data models and data flow. 27
Figure 7: Vertical Data Model for the Demonstrator. 28
Figure 8: The KPI data model for the desmostrator. 30
Figure 9: Example of historical repository structure....32
Figure 10: Structural data model for the demonstrator....33
Figure 11 CityGML Modules 34
Figure 12 Simplified version of UML diagram of building model in CityGML. 35
Figure 13: Building database schema 39
Figure 14: Building example in database 40
Figure 15: Thematic surface example in database....41
Figure 16: Surface geometry example in database....41
Figure 17 Structure of GIS Repository....42
Figure 18: Real time repository data model for the demonstrator....48
Figure 19: Viewer elements: zoom tools (1), map layers (2) and reports window (3) 51
Figure 20: Map tip....54
Figure 21: Report Window 54

Abbreviations and Acronyms

Abbreviation/Acronym	Description
AENOR	Asociación Española de Normalización y Certificación
API	Application programming interface
CIOP	City Information Open Platform
CityGML	City Geography Mark-up Language
ESCO	Energy Savings Company
EC	European Commission
EV	Electric Vehicle
GIS	Geographic Information Systems
HDFS	Hadoop Distributed File System
HMI	Human Machine Interface
ICT	Information and Communication Technologies
IoT	Internet of Things
IUP	Integrated Urban Plans
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
RA	Reference Architecture
RDF	Resource Description Framework
REST	Representational State Transfer
SCADA	Supervisory Control And Data Acquisition
SmartEnCity	Towards Smart Zero CO2 Cities across Europe
SQL	Structured Query Language
UNE	Una Norma Española
WCS	Web Coverage Service
WFS	WORLDWIDE FLIGHT SERVICES
WP	Work Package
WMS	Web Map Service
EU	European Union

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 D6.3 presents the results of Task 6.3 “Data Model Architecture Implementation” within WP6 of the SmartEnCity project. The main objective for this task is to design and implement the data models that will accommodate data coming from the different data sources and demonstrators. The task included the following activities:

- Selection of data identifying the systems deployed in the demonstrators and considering the different stakeholders. The selection will be based on the ICT solution requirements and the Key Performance Indicators (KPI) to be measured in the project.
- Analysis, selection and design of the data models for the platform considering Smart City projects data models.
- Design and Deployment of the infrastructure to accommodate the data models.
- Test and validation of the data models.

Two are the results presented in this task. The first result is a demonstrator available online (see section 0). The demonstrator or prototype is a platform where the data models necessary for SmartEnCity are implemented. The deployed platform agrees with the Reference Architecture described in Task 6.2 (SmartEnCityD6.2, 2017). The demonstrator offers the data models necessary to build the CIOP in the lighthouses. The demonstrator has been constructed using different technologies (SQL, HDFS …).

The second result is this document. This document presents a description of the Reference Architecture (RA) data models proposed for the CIOP. The document includes a general description of the different data models that compose the architecture. The document also presents:

- The approach or methodology followed to obtain both results (RA and prototype)
- Access to the demonstrator users guide (online)

1 Introduction

1.1 Purpose and target group

This report constitutes part of the deliverable "D6.3 – Data Model Architecture Implementation". This report presents a demonstrator that implements data models for a specific data flow according to the Reference Architecture proposed for the SmartEnCity project. The report presents the description of the demonstrator and the process followed to construct that prototype, which is a specific instantiation of the Reference Architecture for a data flow. A data flow is the process data follows from the moment it enters the platform until it is consumed by an application. In that process, data can be transformed and stored in different data models depending on the requirements specified for that business case.

The main objective is to demonstrate the capability of building Data Models according to a Reference Architecture. Some of the data models constructed for this demonstrator will be shared by different data flows while others are specific to the data flow presented in this document. New data flows will be designed and build during pilot construction following their specific requirements. The demonstrator or prototype presented in this task has been deployed using standard technologies and tools that can be accessed through Internet and used in the context of a Smart City project.

The main activities carried out in this task are listed here:

- Selection of data. Identify which data are offered by the systems deployed in the demonstrators and select those required by the different stakeholders. The selection will be based on the ICT solution requirements and the Key Performance Indicators (KPI) to be measured in the project.
- Analysis, selection and design of the data models for the platform considering Smart City projects data models. Depending on the data selected and its behaviour different alternatives will be considered and selected.
- Design and Deployment of the infrastructure to accommodate the data models.
- Test and validation of the data models.

This report is structured in the following sections.

This section presents the purpose of the document, its relation with other WP and deliverables, and how the demonstrators are created from the SmartEnCity reference model.

Section 2 presents the objectives pursued in Task 6.3.

Section 3 presents the approach followed in constructing the architecture model and the data models including the methodology.

Section 4 identifies the data models or repositories necessary to build the Reference Architecture in SmartEnCity. The section presents a description of those data models indicating their characteristics (frequency, volume …) type of data expected, their relation with other repositories and the technologies more suitable for their implementation.

Section 5 includes a description of the demonstrator and its capabilities. The section presents the design and development process followed to build the demonstrator and its data flow. It also presents the data models implemented in the demonstrator and a user guide.

Section 6 presents the main conclusions.

Section 7 presents the references used in the document.

Main target group of the information, the demonstrator 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
MON/MGEP	Task Leader. Responsible of the demonstrator (deliverable). Responsible of the content in this document. Main contributor in Section 1 (Introduction), Section 2 (Objectives and Principles), Section 3 (Overall Approach), several subsection of Section 4 (4.2, 4.2.1, 4.2.2, 4.2.3, 4.2.4), several subsection of Section 0 (5.1, 0, 5.2.1, 5.2.2, 5.2.3, 5.2.4) and Section 6 (Conclusions). Has reviewed contributions to all the sections. Has contributed in the development of the demonstrator.
ET	Has reviewed contributions. Main contributor in Section 4 (4.2.8)
TEC	Main contributor in Section 4 (4.2.5, 4.2.6) and Section 0 (5.2.2, 5.2.4). Has reviewed contributions. Has contributed in the development of the demonstrator.
ETIC	Provider of infrastructure Main contributor in Section 4 (4.2.1, 4.2.7), and Section 0 (0, 5.2.1, 5.2.2, 5.2.3, 5.2.4). Has reviewed contributions. Has contributed in the development of the demonstrator.
CAR	Main contributor in Section 4 (4.2.2). Has reviewed contributions.
ACC	Main contributor in Section 5 (5.2.2, 5.2.3). Has reviewed contributions. Has contributed in the development of the demonstrator.
MTEL	Provider of infrastructure. Has contributed in the development of the demonstrator.
GIS	Provider of infrastructure Main contributor in Section 4 (4.2.5, 4.2.6), and Section 0 (5.1, 0, 5.2.1, 5.2.2, 5.2.3, 5.2.4). Has reviewed contributions. Has contributed in the development of the demonstrator.
VG	Has reviewed contributions. Main contributor in Section 4 (4.2.8)

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
D6.1	This deliverable provides the requirements identified for SmartEnCity
D6.2	This demonstrator presents the Reference Architecture for SmartEnCity
D6.4	This demonstrator extends D6.2 considering the interoperability needs for SmartEnCity.
WP3, WP4 and WP5	The implementation in each lighthouse will agree with the Reference Architecture and the layers and modules defined in it. Data models will be implemented there
WP7	KPIs are defined in that work package. Data Models for KPIs have been built according to D7.2 and data flow construction in SmartEnCity CIOP is outlined in D7.9 (Task 7.3)

Table 3: Relation to other activities in the project

1.4 Reference Architecture and Demonstrator (data models)

As it was done in (SmartEnCityD6.2, 2017) it is necessary to outline the differences between a Reference Architecture and a demonstrator. According to (Wikipedia, 2016) (based on ISO/IEC/IEEE 42010) reference architectures provide a template solution for the architecture (aka. architectural blueprint) for a **particular domain**. It also provides a **common vocabulary with which to discuss implementations**, often with the aim to stress **commonality**.

A reference architecture often consists of **a list of layers, modules and functions** and some indication of their **interfaces (or APIs) and interactions** with each other and with elements located outside of the scope of the reference architecture.

Reference architectures provide a template, often based on the **generalization of a set of solutions**. These solutions may have been generalized and structured for the depiction of one or more architecture structures based on the harvesting of a set of patterns that have been observed in a number of successful implementations. Further it shows how to compose these parts together into a solution. **Reference architectures will be instantiated for a particular domain or for specific projects.**

A demonstrator or prototype consists on a technological solution that fulfils the requirements of a Reference Architecture and provides the modules and functionality specific for the domain it represents. Several demonstrators build with different technologies and frameworks can agree with a common Reference Architecture and be consequently valid instantiations or implementations of that architecture.

In **(SmartEnCityD6.2, 2017)** the Reference Architecture proposed for SmartEnCity was presented. The SmartEnCity Reference Architecture is a layered model based on UNE 178104:2015 (AENOR CTN-178 group standard) **(AENOR CTN-178, 2015)**. The demonstrator presented in this document is an instantiation of that Reference Architecture

considering the **Data Models** to be constructed. Each demonstrator could use different technologies and frameworks to construct Data Models and at the same time agree with the Reference Architecture.

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…). (This Deliverable D 6.3)
- 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. (Deliverable 6.4)
- 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. (Deliverable 6.5)
- 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. (Deliverable 6.6)
- Integrate and validate the different modules of the ICT platform. (Deliverable 6.7)

The overall objective for this task (Task 6.3) and its deliverable (D6.3) is to design and implement the data models that will accommodate data coming from the different data sources and demonstrators. The expected deliverable is a demonstrator that presents the instantiation of those data models in a real platform.

2.2 Objectives of Task 6.3

The main objective for this task/deliverable is to design and implement the data models that will accommodate data coming from the different data sources and demonstrators. Data from the systems deployed in the lighthouse experiments must be analysed and selected. Thus system deployment tasks in work packages 3, 4 and 5 must be closely followed. The database model will also consider data consumer requirements; the added value services

and the HMI requirements from tasks 6.5 and 6.4 and the ICT solutions to be developed in work packages 3, 4 and 5. The model has to be flexible to accommodate heterogeneous data from present and future communities. The model will be conditioned by the volume, velocity and variety of data. Depending on those conditions and the nature of the information, the data models included might vary from a traditional relational database to a NoSQL database, a distributed unstructured model based on Big Data or even accommodate different models altogether.

At this moment in the project the detail for all ICT applications and systems to be implemented/installed in work packages 3, 4 and 5 was not fully available so the efforts have been put in demonstrating the capability of building all the data models necessary in the project for a specific data flow. All the repositories to be built in the final solution have been constructed although additional effort will be required in the pilot work packages to adapt those data models to the requirements of the applications and systems to be installed. This task has to closely work with work package 7 and as a result Key Performance Indicators (KPI) data model representation has been produced.

A specific implementation of the data models is presented as the main result of this task. The data models agree to the Reference Architecture specifications. That is to say, the deliverable will provide a demonstrator or prototype that accommodates all the data models identified in that Reference Architecture.

The main activities performed in this task are:

- Selection of data. Identify which data are offered by the systems deployed in the demonstrators and select those required by the different stakeholders. The selection will be based on the ICT solution requirements and the Key Performance Indicators (KPI) to be measured in the project.
- Analysis, selection and design of the data models for the platform considering Smart City projects data models. Depending on the data selected and its behaviour different alternatives will be considered and selected.
- Design and Deployment of the infrastructure to accommodate the data models. **Description of the data models and demonstrator included in this deliverable D6.3.**
- Test and validation of the data models.

3 Overall Approach

In this section, the steps followed to achieve the deliverable are outlined. The methodology and the Reference Architecture selected in (SmartEnCityD6.2, 2017) have been adopted and extended in this deliverable. The same approach followed in (SmartEnCityD6.2, 2017) has been used to define and later implement the data models. The approach is shown in Figure 1 and starts from analysing the requirements which consolidate in the Concept Architecture report. This report includes the data flows necessary to build the applications to be developed in the demonstrator. The Concept Architecture evolves into the Architecture Design stage. The result of this stage is the Reference Architecture which includes the data models for the solution. The results are presented as reports describing the Reference Architecture and as demonstrators. These demonstrators are instantiations of the Reference Architecture created as prototypes that can be easily shared and reused in the project.

In this task, two major approaches have been implemented:

1. Identify and describe the data models necessary for the SmartEnCity CIOP extending the Reference Architecture using the methodology proposed in (SmartEnCityD6.2, 2017). Data models are described including a general description of the model, the type of data expected, the relation to other repositories and the technologies proposed for the implementation. The results for this part are presented in Section 4.
2. Build a demonstrator that implements those data models for a specific data flow following the data flow construction process proposed in (SmartEnCityD7.9, 2017). The results from this part are presented in Section 0.

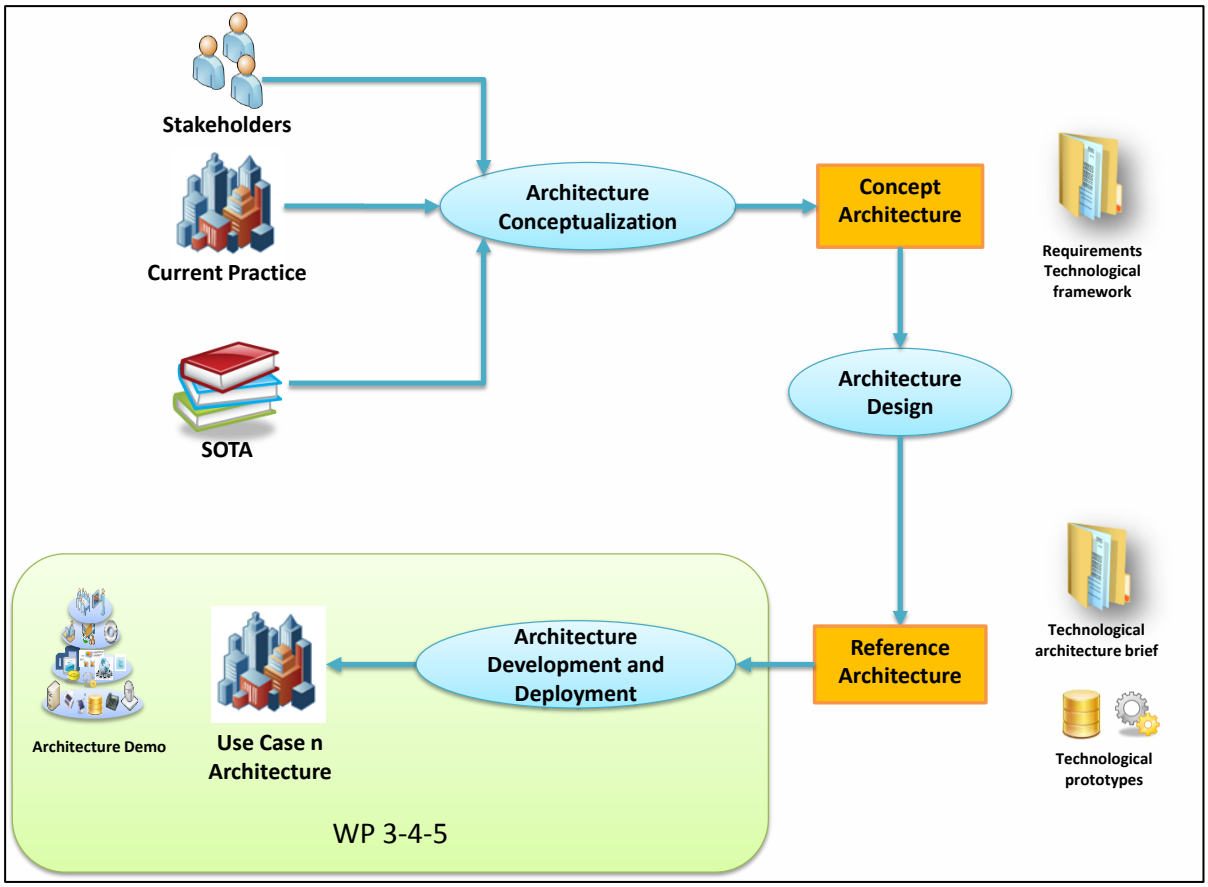


Figure 1: Architecture building approach

4 SmartEnCity CIOP Reference Architecture Data Models

4.1 Reference Architecture in SmartEnCity

In (SmartEnCityD6.2, 2017), the Reference Architecture proposed for SmartEnCity was presented. The Reference Architecture is a layered model based on UNE 178104:2015 (AENOR CTN-178 group standard). Figure 2 presents the layers and modules composing the reference architecture. It is worth outlining that the core of the Reference Architecture is an IoT platform. A reference architecture implementation gathers different types of data through the acquisition layer: Real time data (sensor data), Open Data (weather forecast), district heating, mobility, social networks, etc. Those data are stored and treated in the knowledge layer. The interoperability layer enables the consumption of those data through APIs. The intelligent service layer offers services and applications for the different vertical domains (energy, environment, mobility, etc.) that have been developed based on the Smart City infrastructure and available data sources.

The repositories necessary to build a Smart City platform are also outlined in the Reference Architecture. The repositories or data models necessary for its representation are generic (highlighted in orange). It should be noticed that the data models and the technologies provided to build high added value services are the main components that distinguish an IoT platform from a domain specific platform as the SmartEnCity CIOP platform. That is, what shapes a platform according to the domain (smart city) are the verticals and high added value services.

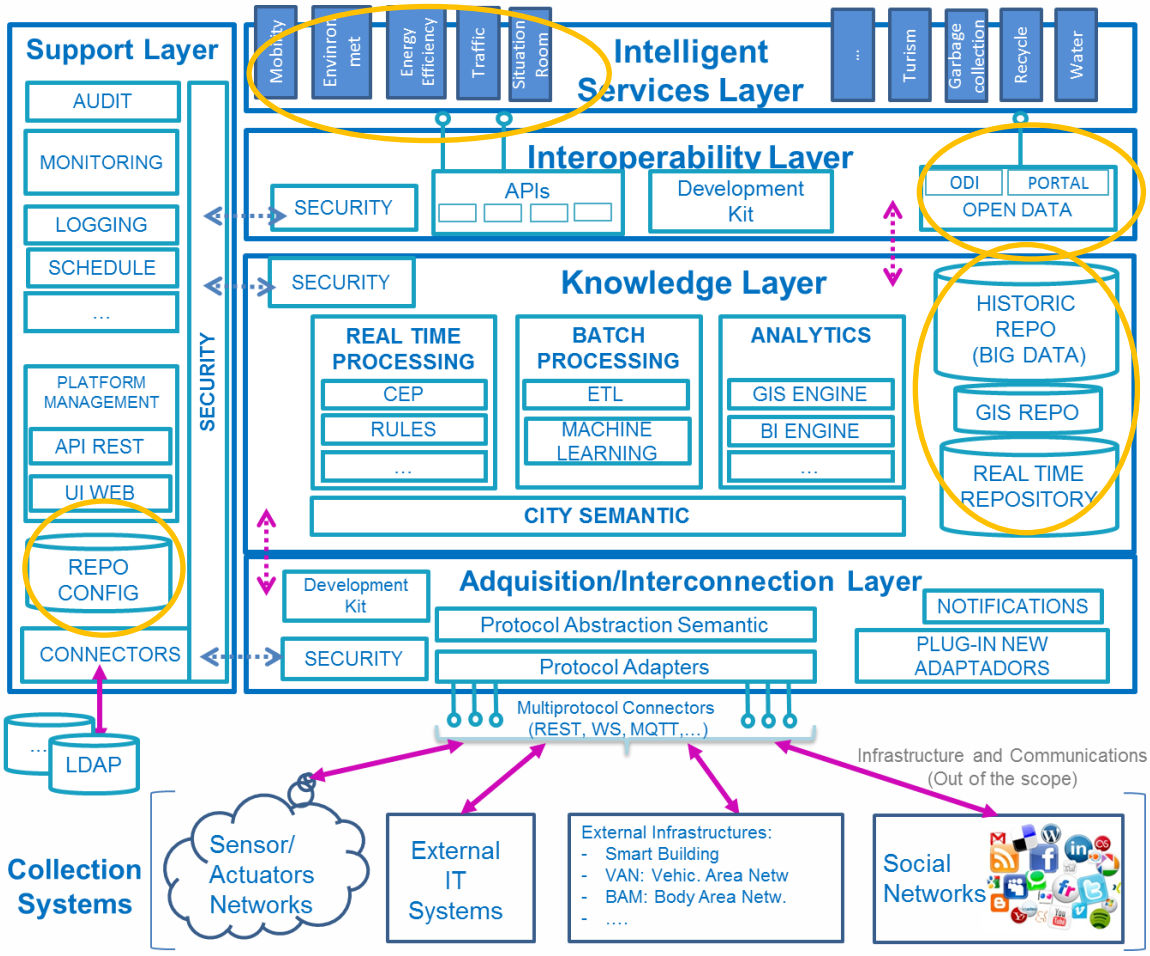


Figure 2: Smart Cities General Architecture

This section presents a description of the data models or repositories identified for the SmartEnCity CIOP. The data models are part of the Reference Architecture and are necessary to represent the different verticals of Smart Cities (energy, mobility, environment …), indicators (KPI), context information and infrastructure access and management.

The data models have been identified and defined following the methodology outlined in section 3. A detail description of the steps taken to define the reference architecture was presented in (SmartEnCityD6.2, 2017) (Section 4). During the workshops organised, the following issues related to the data models were discussed:

- Decide on data common to the three lighthouses and its requirements
- Identify verticals and possible stakeholders

Data models common to all the instantiations of the Reference Architecture were identified during those sessions (highlighted in orange above). The common repositories include the real time repository, the historical repository, the GIS repository, the structural repository, the configuration repository, the KPI repository and the verticals. A general description of those data models including the type of data expected, the relation with other repositories and the technologies more suitable to implement those data models was provided (see section 4.2). The implementation of those models implies the selection of technologies and the adaptation of existing data by means of data management (aggregation, extraction …). The data models implemented should be common but the level of completeness might vary from one lighthouse to the other. That is, some demonstrators might not implement certain repository or could only provide part of the data that the model is able to accommodate while others could implement the whole data structure.

The most important data repositories are KPIs and verticals. KPIs are valid to determine the behaviour of systems and measures. They are relevant enough to constitute an additional data model. KPIs are the main topic in work package 7. Thus, there has been closed collaboration with that work package in order to represent the KPI data model accordingly to the requirements and constraints identified there.

Another important issue addressed was the verticals or domains to be implemented in the project. In this stage of the project and from the contacts with the stakeholders the following verticals have been identified:

- Energy assessment
- Mobility
- Environment
- Social Acceptance

At this moment in the project the detail for all ICT applications and systems to be implemented/installed in the verticals of work packages 3, 4 and 5 is not fully available so the efforts have been put in demonstrate the capability of describing all the data models necessary for the Reference Architecture (this section) and building those data models for an specific data flow as a demonstrator (see Section 0).

Stakeholders interested in those verticals have been also addressed. The following stakeholders have been identified: resident, municipality, WP7 (KPIs) and Energy Service Companies (ESCO). Further work on the profiles and the needs required by each stakeholder is necessary in WP3, 4 and 5.

The following subsections present the data model description.

4.2 Data Models

While the IoT platform implements the functionality of the different parts of a Smart City project, the logic is represented/supported in the data models. The platform developed in Task 6.2 and presented as a demonstrator, is the base from which the demonstrator presented in Task 6.3 is developed. That is, Task 6.3 adds SmartEnCity data models to the demonstrator presented in D6.2.

Data models necessary to represent the Smart City domain, the different verticals of Smart Cities, common parts (such as indicators or KPIs), location and infrastructure are described in this section. Figure 3 presents the repositories identified for the Reference Architecture in SmartEnCity. Each repository is described in the following chapters including

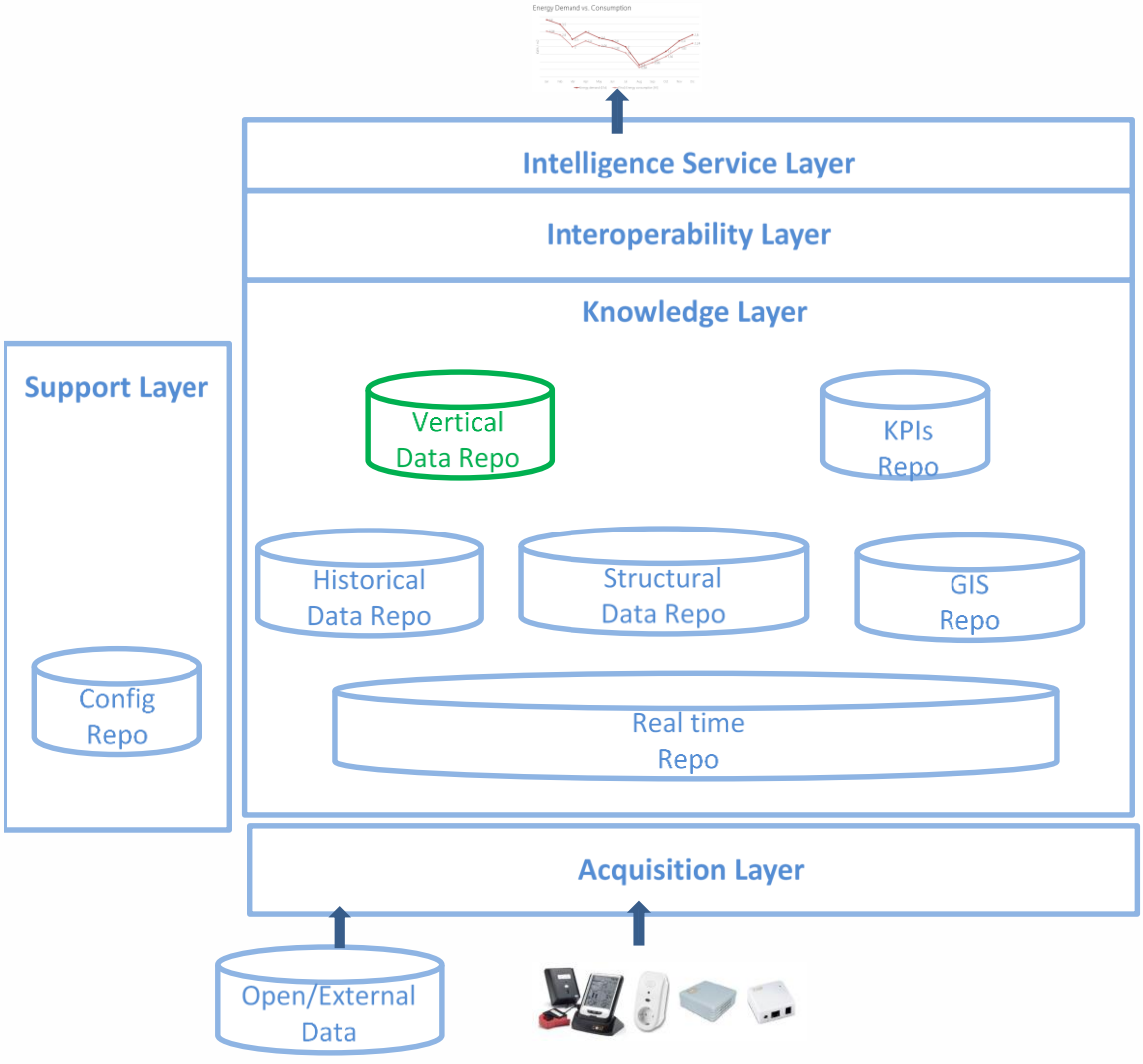


Figure 3: Data Models for CIOP

4.2.1 Vertical Data Repository

Description: Vertical data models are closely related to the different applications and data that will be used in the lighthouses. Therefore, the implementation of this repository depends on each lighthouse and the application to be developed in those. Nevertheless, the work explained in section 5.2.1 (Vertical Data Repository (Energy)) is an example of the type of

data model expected for this repository. The vertical for this demonstrator is focused in the electricity measurements collected at household level to monitor and present energy consumption data.

Type of Data: Vertical repositories might vary depending on the nature of data (volume, frequency…).

Relation to other Repositories: The relation with other repositories needs also to be addressed. Thus, we clearly detect the need to integrate this data models with the Structural Data Repository and the Real Time Repository. The Structural repository will provide information about location and context while the Real Time Repo will provide information collected in the city by sensor and systems. There will be relations with the Historical Data Repository since historical data from the verticals will be stored.

Technologies: Might vary depending on data nature. For the demonstrator since no Big Data conditions are expected in this repository, a relational database model has been designed. The measurements in this repository will consist of aggregated data from each sensor extracted from the Real Time repository.

4.2.2 KPI Repository

Description: The data model for the Key Performance Indicators (KPIs) stores the calculated/aggregated indicators. These indicators have been selected in WP7 and are calculated computing the values of different verticals and Real Time repositories. This repository will be common for all lighthouses in the project.

Type of Data: Aggregated data with low frequency of collection and reduce volume is expected. The number of indicators is not large either (around 50 indicators per lighthouse).

Relation to other Repositories: KPI Data Repository will have relations with the Real Time Repository and in some cases with the Vertical Data Repositories. It will also be related to the Structural Data Repository. For example, a KPI could be related to a district and could have aggregated data extracted from the electrical vertical and electrical real time repository.

Technologies: Since aggregated data is stored and not a real time data with high frequency and volume is expected a Relational Database technology has been identified as the best option for KPI repository.

In section 5.2.1, the design proposal for the demonstrator is presented.

4.2.3 Historical Data Repository

Description: The objective of the historical data repository is to store the historical data collected in the platform. Historical data is gathered from other repositories and stored permanently in a common space where historical data can be recovered in case it is necessary (data analysis, disaster recovery …). The swift of container is performed in order to free up space in the main repositories (verticals, real time, KPIs …).

Type of Data: The most common option is to store information in files. Files can be saved in their original data format or after performing some kind of transformation in plain text format, csv or similar (for example DB scripts). Storing information in files eases the reutilization of information with different technologies in the future.

Relation to other Repositories: There will be relations with the others Data Repositories in the knowledge layer since historical data from these repositories will be stored.

Technologies: To store the historical data, a scalable, flexible and fast technology is required. To meet these requirements Hadoop File System (HDFS) has been selected as the best option. In addition to these characteristics, Hadoop deploys a failure resilient distributed system that assures availability and consistency. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of a node failure, there is another copy available for use in another site. HDFS starts replicating the data to another existing node as soon as it detects new content. Scalability is also considered in Hadoop where new nodes can be added to address an increase in the storage requirements or in avoiding node failure.

4.2.4 Structural Data Repository

Description: The objective of the structural data repository is to store the structural data collected in the platform and the relations among them. This repository stores information from countries in the system to households. This information is then used to store data related to the structural elements (such as consumption per household, KPIs per district…)

Type of Data: The most common option is to store information in relational databases. This way, relations among the different elements in the repository are stored.

Relation to other Repositories: There will be relations with the others Data Repositories in the knowledge layer, such as verticals or KPIs. In the energy vertical repository, for example, each energy measurement in a household is stored. Therefore, a relation between that repository and the structural one is needed. KPI repository also stores specific KPIs grouped by different structural elements, so a relation is needed.

Technologies: Structural elements are closely related among them, and that relation is important to be stored. Therefore, relational data bases have been selected to store this kind of information.

4.2.5 GIS Structural Data Repository

Description: The data model for the Structural Data Repository stores the urban data model which sets the basis for the structural information of the city. The structural data repository is based on the standard CityGML, which is an open standardised data model and exchange format to store digital 3D models of cities and landscapes. It defines ways to describe most of the common 3D features and objects found in cities (such as buildings, roads, rivers, bridges, vegetation and city furniture) and the relationships between them. It also defines different standard levels of detail (LODs) for the 3D objects, which allows us to represent objects for different applications and purposes.[2]

Type of Data: The structural data repository will include geometry of the main city objects, as well as semantic properties of the different kinds of the 3D city object.

Relation to other Repositories: Structural Data Repository will provide to the GIS Repository with the most updated information about the structural data (geometry and static attributes) of the study area of the city. This repository will be used for the 3D visualization of

[2] CityGML homepage https://www.citygml.org/

the city and it requires access to other repositories in order to access information to be shown in the 3D model.

Technologies: A relational database will be used to store the information of the structural data repository. 3DCityDB will be used to efficiently store and quickly process information of the CityGML model. An implementation of the 3DCityDB in PostGIS will be used. WFS (Web Feature Services) will be used for accessing the information stored into the repository.

In section 0, the design proposal for the demonstrator is presented.

4.2.6 GIS Repository

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

GIS Repository is divided in four categories corresponding to: urban structures, street furniture, parks and gardens and urban mobility that are described in the next section along with the detailed description of the data for each field.

Type of data: This repository will contain the 2D geometry of the different city elements as well as the alphanumerical attributes related to each layer to complete the information.

Relation to other Repositories: This repository is closely related to the structural repository mentioned above and will take the data stored in it to complete the information.

Technologies: To store the model information it will be used a spatial database deployed in SQL Server. This information will be served through Geoserver using GIS OGC standard services like WCS, WFS, WMS. Geoserver will be deployed over a web application server.

4.2.7 Configuration Repository

Description: The Configuration Repository is the database of the needed information to manage the different users, profiles and security permissions. The Configuration Repository acts in a transversal way to the rest of the layers and components of the platform by offering services of users' management, security, access monitoring and others.

Type of Data: It consists in a structured database where the following data is stored: users' identification with passwords, permissions and allowed access; security rules and conditions; and the structure of the platform, layers and components.

Relation to other Repositories: As the Configuration Repository is a transversal layer it has relation with most of the repositories of the rest of the layers. The users´ management, access control and security are relative to all the databases and repositories of the rest of data.

Technologies: To access to the Configuration Repository a Web interface will be developed where the management of uses' access will be don bet REST services in each of the repositories. Each repository will have its own database of users and permissions.

4.2.8 Real Time Repository

Description: The Real Time Repository contains data received from the different sensors and external systems that act as external information sources for the different components of the platform.

Type of Data: The data in this repository is not structured, as the data sources can be so heterogeneous that it cannot be structured in a meaningful and useful way. Most of the time, this data will be time dependent (e.g., measurements from sensors, made at regular time intervals), and the processing of that data will also consider time intervals as the main selection and processing criteria (e.g, average energy consumption over 1 day, 1 week, etc.). It means the repository will be mainly dealing with so called “Time Series Data”. Also, the potential volume of data received and stored in this repository is very high (there can be thousands of different sensors, each sending several data points at potentially small time intervals).

Relation to other Repositories: As the Real Time Repository is the main external data input to the platform, especially data from external monitored or controlled systems, it has relationships with the repositories holding data about such external systems, such as the Historical Data repository, the KPI repository and the Vertical Data repositories.

Technologies: As explained in the “Type of Data” paragraph, the data stored in this repository will mainly consist of time series data, and the volume of such data will potentially by very high. A perfect fit for these two requirements is the usage of so-called Time Series Databases, or TSDBs. There are many TSDBs to choose from, both commercial and open-source. Also several traditional SQL and non-traditional noSQL database solutions can be extended or configured to be used as a TSDB.

5 SmartEnCity Demonstrator (Data Model)

This chapter presents the demonstrator developed for SmartEnCity in Task 6.3. First the process followed to design and develop the prototype is outlined. Secondly the actual demonstrator is presented.

5.1 Demonstrator design and development

The different applications and solutions to be provided in SmartEnCity are data centered. Thus, we will consider different flows of data depending on the process data will follow for a specific application or solution. A data flow is the process data follows from the moment it enters the platform until it is consumed by an application. The stages data passes in this process includes data uploading/collecting, transformation, storing, analysis, recovering and data downloading or consumption (visualization). See (SmartEnCityD7.9, 2017) for more detail on those stages.

Each application may depend on different data sources, may apply different processing algorithms or models for analysis and may be presented in a different manner. The storage of that data might be different as well. Some data will be stored locally in structured databases; other might come from other repositories (structure or no) or even from external sources (open data). Consequently, each application will build the data flows depending on those characteristics.

In order to have a common framework to design, construct, validate and commission the data flows, (SmartEnCityD7.9, 2017) proposes a framework based in (Ralph Kimball, 1998) data processing lifecycle. The framework is shown in Figure 4. The detail explanation of each stage is presented in (SmartEnCityD7.9, 2017). All data flows to be developed in SmartEnCity will follow this framework for construction.

The work conducted in this demonstrator has involved most of the stages in the framework but it has concentrated specially in the Data Modelling stage since the demonstrator focuses on building data models for SmartEnCity.

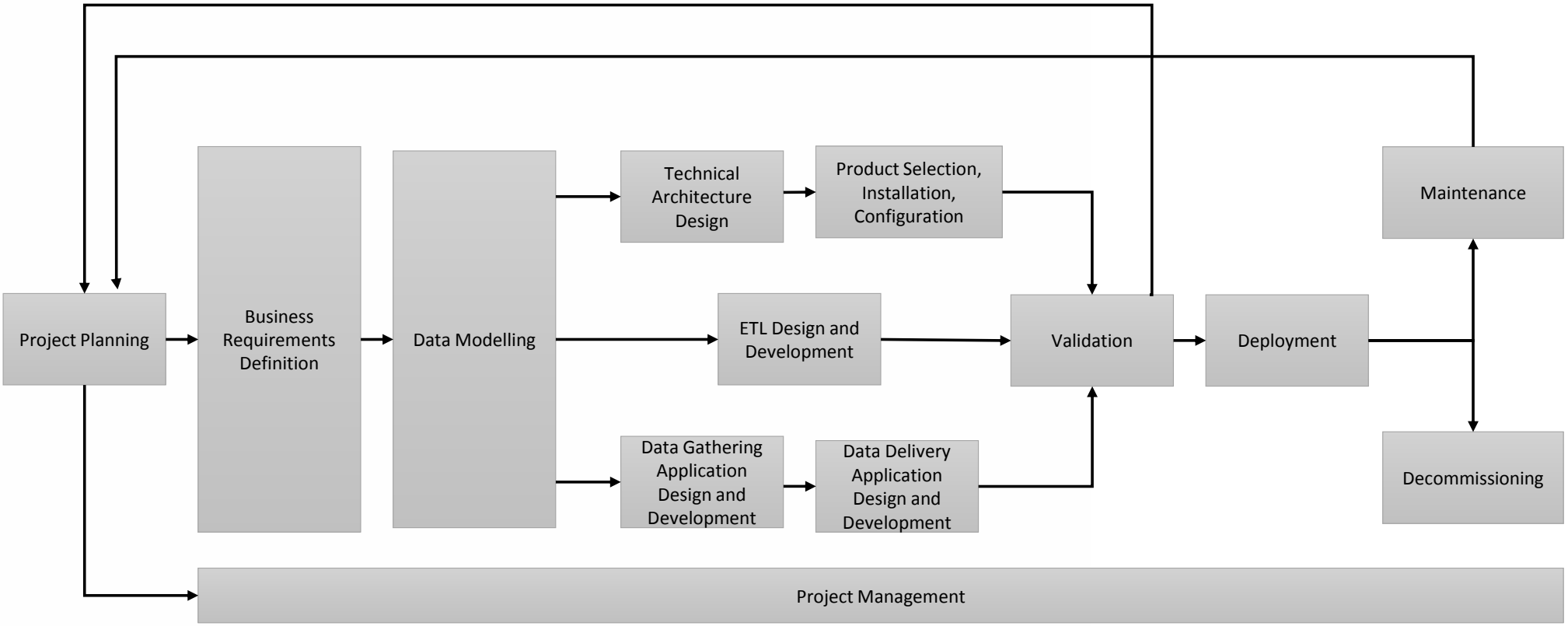


Figure 4 Data Process lifecycle

Data models are the real representation of a domain. It should be noted that data models necessary to represent the Smart City domain have been constructed with the requirements available at this stage. Demonstrator work packages (WP 3, 4 and 5) could use these data models to implement the solutions to be developed in their lighthouses. Different implementations for the data models are also valid. They have to agree with the Reference Architecture definitions given in Section 4. Data model implementations for each LH are highly dependent on technologies implemented and also on use cases. In case where off the shelf IoT platform is used, platform itself might dictate how data is collected, transformed and stored. In this case, additional processing services might be needed to provide data for KPIs or verticals. In addition to data models, IoT platform might have integrated asset and device management, GIS services etc.

5.2 Demonstrator Description

For a demonstration of results, we have created a basic CIOP that presents all the data models necessary for SmartEnCity. The demonstrator is a fully functional IoT platform that has all the generic layers and functionalities and data models, but only manages content for a specific data flow. Figure 5 presents the architecture of the solution with the repositories available. The idea behind the demonstrator is to collect data from sensors (energy monitoring system), store them in the CIOP platform, transform them to calculate KPIs and present those data in a context where data is associated to the city using GIS visualization solutions.

The data flow considered for the demonstrator is presented in section 5.2.1.The data models are detailed in section 5.2.2. A user guide where the reader can obtain information to access the demonstrator is available in section 5.2.3. Finally, a summary of the technologies, tools and mechanisms used in the demonstrator is presented in section 5.2.4.

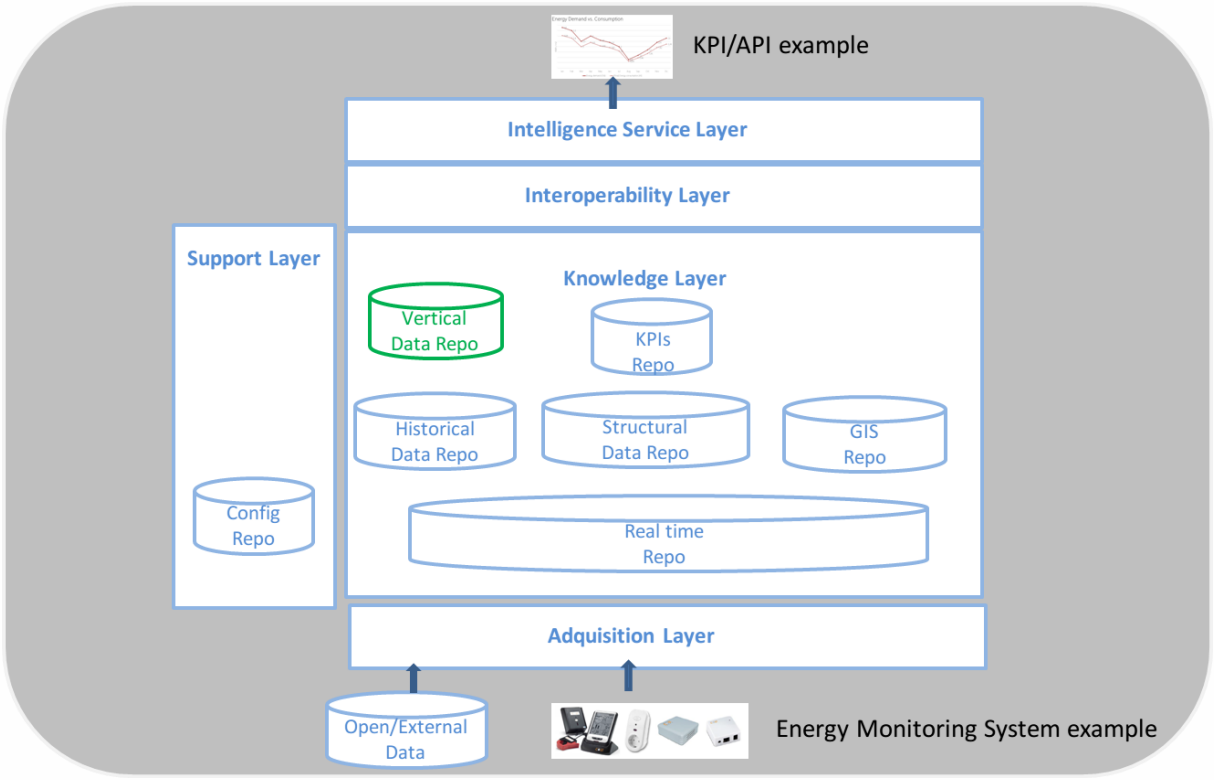


Figure 5: Architecture of the demonstrator.

5.2.1 Demonstrator data flow

As mentioned throughout this document, the different solutions to be developed in SmartEnCity will be constructed based in data flows. A data flow is the process data follows from the moment it enters the platform until it is consumed by an application. For the demonstrator and as an example, we have constructed a data flow that involves all the data models necessary for SmartEnCity. The main idea is to build a data flow that also goes through all the stages in the data process (see section 4 in (SmartEnCityD7.9, 2017)). Thus, data will be collected from sensors at the acquisition level and stored in the real time repository. The row data of the real time repository is extracted, summarized and stored in the vertical repository every hour. Moreover, once a week, a backup of the real time repository is also saved in the historical repository. Finally, every week, data from vertical repository is aggregated and stored in the KPI repository and GIS repository for its further visualization. All this flow is managed by different interoperability mechanisms explained in SmartEnCityD6.3, 2017. A visualization of the data flow is also visually represented in Figure 6 (blue arrows).

Figure 6 also shows which repositories are related among them. Vertical repository and KPI repository are related to Structural Data Repository. Using these relations, for example, we can know the measurements related to a household or the KPI value for a specific building. Finally, some structural elements are related to the GIS repository so they can be visualized.

That way, KPIs are finally visualized using a web application that presents data in a context (City graphical interface). See section 5.2.3 for more detail on the visualization add provided.

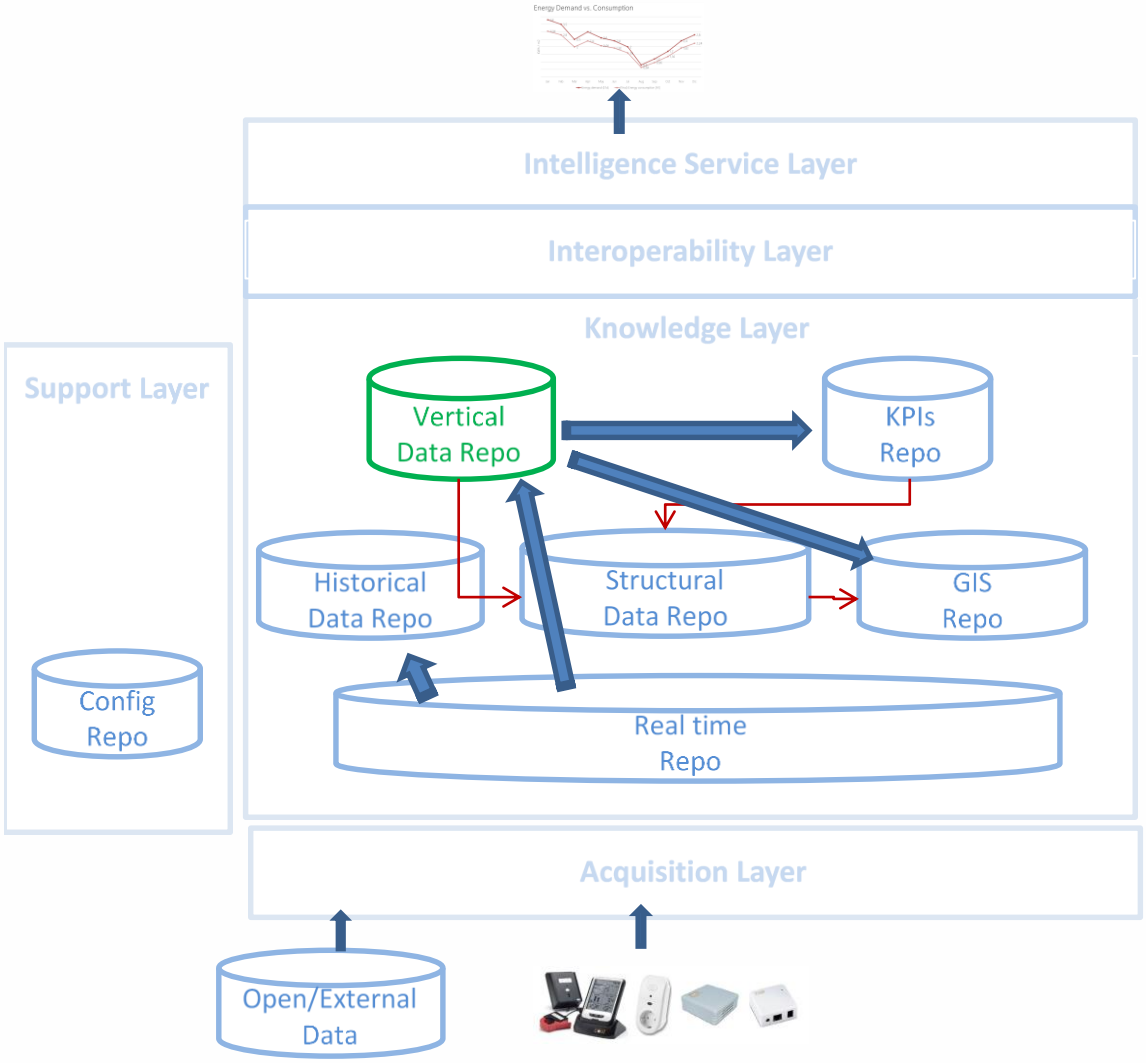


Figure 6: Related data models and data flow.

5.2.2 Data Models for demonstrator

This section presents the SmartEnCity data models used for the demonstrator. Data models necessary to represent the Smart City domain, the different verticals of Smart Cities, common parts (such as indicators or KPIs) and infrastructure are described next.

Vertical Data Repository (Energy)

The repository gathers the aggregated data of the measurements of sensors in the households. Therefore, it must be related to Real Time repository (sensor measurements) and Structural Repository (households, buildings, districts…).

Below, all tables of the data model are described, along with their columns and Data Types.

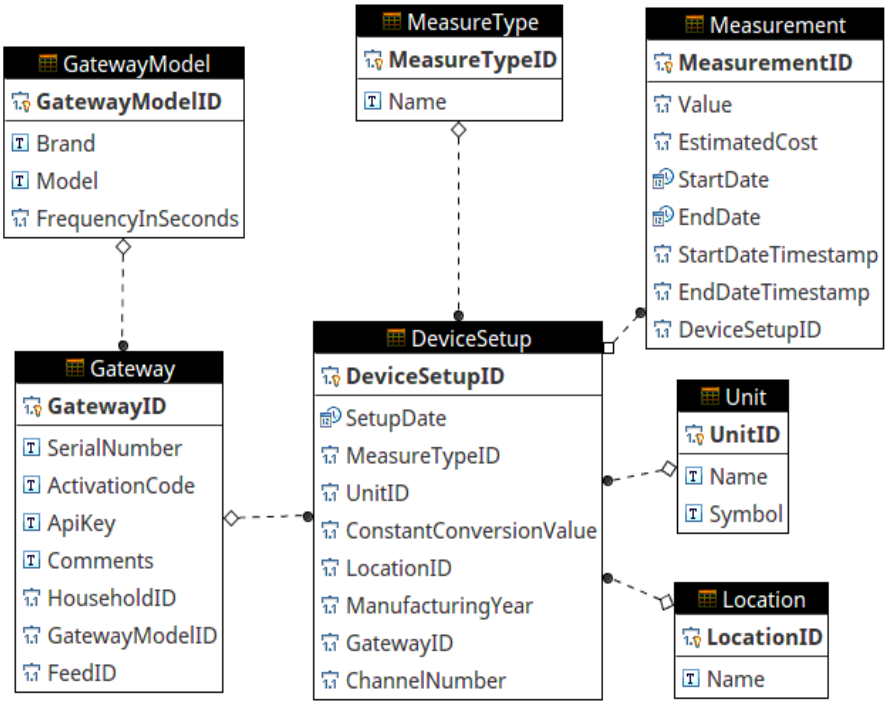


Figure 7: Vertical Data Model for the Demonstrator.

DeviceSetup

This table will store each configuration for each sensor in the system (the configuration for each channel of the gateway).

- **DeviceSetupID** INT: Device Setup auto numeric identifier.
- **SetupDate** DATETIME: Stores the date when the configuration has been created (usually, when the gateway has been installed).
- **MeasurementTypeID** INT: The reference to the type of measurement we will receive, such as consumption, humidity or temperature (foreign key).
- **UnitID**: the reference to the unit of the measurements we will be receiving (foreign key).
- **ConstantConventionValue** FLOAT: Constant to transform Device's energy measurements to a unified unit (Wh for example).
- **LocationID** INT: Reference to household room types (bedroom, living room, kitchen…).
- **GatewayID** INT: Reference to the Gateway that will send the measurements of the device.
- **ChannelNumber** INT: The Gateway's channel from where the data will be receiving.

Unit

This table will store the different units that will be used in the system (for measurements, KPIs…).

- **UnitID** INT: Unit's auto numeric identifier.
- **Name** VARCHAR(45): name of the unit for human readability ("Watt", "Degree Celsius", "Watt hour"...).
- **Symbol** VARCHAR(45): symbol of the unit (e.g. W, ºC, Wh…).

MeasureType

This table contains the three main energy measurement types managed in the project (Electrical, Thermal, Comfort [humidity]).

- **MeasureTypeID** INT: Measurement Type's auto numeric identifier
- **Name** VARCHAR: descriptive name of the measurement type (e.g. Electrical, Thermal, Comfort).

Measurement

This table will store each measurement from each device in the system. The measurement is usually send in a certain period of time.

- **MeasurementID** INT: Measurement auto numeric identifier.
- **Value** FLOAT: the value of the measurement (the unit of the measurement is stored in the devicesetup, we don't receive the unit from the gateway).
- **EstimatedCost** FLOAT: estimated cost of the measurement (if it is a consumption measurement).
- **StartDate** DATETIME: start date of the specific measurement.
- **EndDate** DATETIME: end date of the specific measurement.
- **StartDateTimestamp** TIMESTAMP: Calculated Unix Timestamp.
- **EndDateTimestamp** TIMESTAMP: Calculated Unix Timestamp.
- **DeviceSetupID** INT: the reference to the device setup id

Gateway

The gateway is the element that gathers the measurements of the devices of a household and sends them through different channels.

- **GatewayID** INT**:** Gateway's auto numeric identifier.
- **SerialNumber** VARCHAR: The serial number of the gateway given by the manufacturer.
- **ActivationCode** VARCHAR: calculated using serial number for device discovery (used only once).
- **ApiKey** VARCHAR: The API Key generated by the system that enables data gathering.
- **FeedID** INT[12]: the final id that will be used for communication purposes, given by the system to the gateway.
- **Comments** VARCHAR: Field to save other things as installer's name.
- **HouseholdID** INT: The reference to the household where the gateway is installed.
- **GatewayModelID** INT: The reference to the model of the gateway.

Gateway Model

This table stores the information about the different Gateway models.

- **GatewayModelID** INT: Gateway model's auto numeric identifier.
- **Brand** NVARCHAR: Brand of the gateway model.
- **Model** NVARCHAR: The reference or name that has given the brand's model of the gateway.
- **FrequencyInSeconds** INT: How often the gateway sends the data (in seconds).

Location

Location of the sensor in the household (e.g. kitchen, bedroom, living room…).

- **LocationID** INT: Location's auto numeric identifier.
- **Name** VARCHAR(45): Human readable name for the location.

KPI Repository

Figure 8 shows the design of the relational database for the KPI data model. It is inspired in the KPIs defined in D7.2 in order to be able to store the indicators from all the verticals in the same repository.

Below, all tables of the data model are described, along with their columns and Data Types.

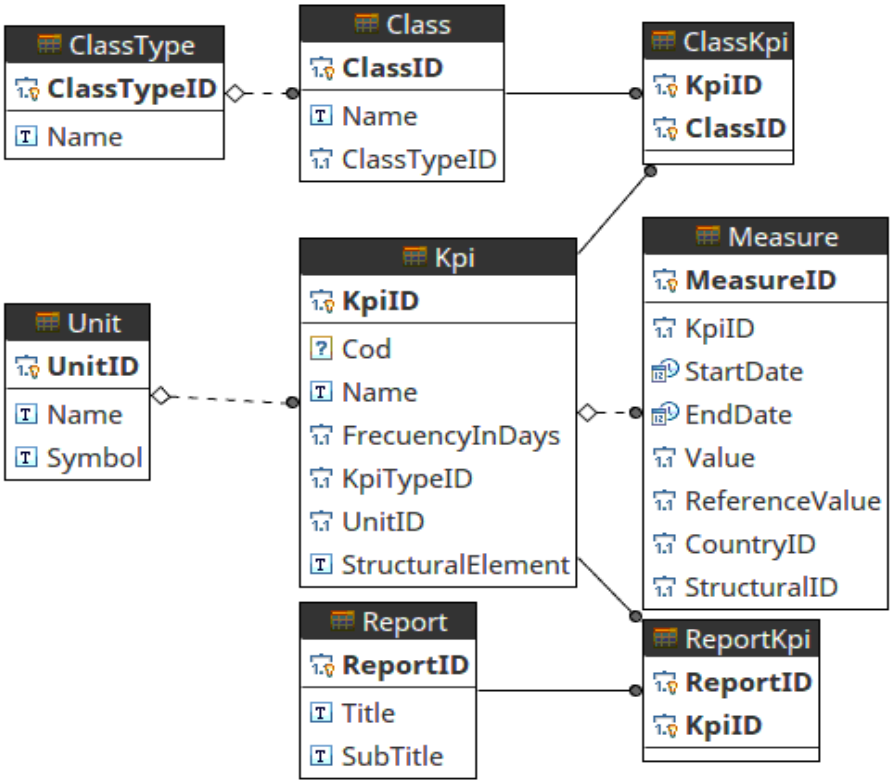


Figure 8: The KPI data model for the desmostrator.

Kpi

This table stores the general information for each KPI.

- **KpiID** INT: KPI's auto numeric identifier inside the system.
- **Cod** INT: Identifier of the KPI (outside the system, in a human readable form).
- **Name** NVARCHAR: Descriptive human readable name of the KPI.
- **FrequencyInDays** INT: How often the KPI is calculated (in days).
- **UnitID** INT: Reference to the unit of the measurements of the KPI (e.g. KWh/m²).
- **StructuralElement** NVARCHAR: The structural element that the KPI is calculated for (Building, District…)

ClassKpi

This table stores the relation among classes and KPIs. A class can have more than one KPI, this is a table created because of that *N to N* relation.

- **KpiID** INT: Reference to the Kpi.

- **Class** INT: Reference to the class.

Class

This table stores a first level classifications for the KPIs.

- **ClassID** INT: Class's auto numeric identifier.
- **Title** NVARCHAR: Descriptive title of the Kpi class.
- **ClassTypeID** INT: Reference to the Kpi class type.

ClassType

This table stores the second level classifications for the KPIs.

- **ClassTypeID** INT: Class's type auto numeric identifier.
- **Name** NVARCHAR: Descriptive name of the KPI class type.

ReportKpi

This table stores the relationships among reports and KPIs. A report can have more than one KPI, this is a table created because of that *N to N* relation.

- **ReportID** INT: Reference to the report.
- **KpiID** INT: Reference to the KPI.

Report

The report is a set of different KPIs to be shown.

- **ReportID** INT: Report' auto numeric identifier.
- **Title** NVARCHAR: Report's title.
- **SubTitle** NVARCHAR: Report's subtitle.

Unit

This table will store the different units for the Measures that will be used in the KPIs.

- **UnitID** INT: Unit's auto numeric identifier.
- **Name** VARCHAR(45): name of the unit for human readability ("Watt", "Degree Celsius", "Watt hour"...).
- **Symbol** VARCHAR(45): symbol of the unit (e.g. W, ºC, Wh…).

Measure

This table contains main KPI data. It is loaded with aggregated data from the different verticals.

- **MeasureID** INT: Measure's auto numeric identifier.
- **KpiID** INT: Reference to the KPI related to the measure.
- **StartDate** DATETIME: What date have we started calculating the measure from?
- **EndDate** DATETIME: Until what date have we calculated the measure?
- **Value** DECIMAL(9,2): The results of the aggregated data calculated from the StartDate to the EndDate, or the specific value of the KPI in that period of time.
- **ReferenceValue** DECIMAL(9,2): The estimated value this measure should have, according to some simulations (baseline value).
- **CountryID** INT: Reference to the country where the measure has been taken.
- **StructuralID** INT: The identification of the specific structural element the measure is related with (a specific building ID, or District ID…).

Historical Data Repository

Hadoop has been selected as the technological solution for the demonstrator. To store the historical data of the project and aiming at maintaining a logical structure in Hadoop, the following folder structure is proposed for this repository:

1. Data repo level: Different folders per each main data repository (verticals, GIS, KPIs …).
2. Time slots level (written using ISO 8601): Different time slots. In each data repository the best time slots will be defined depending on the characteristics of the technologies used, the data behaviour and the repositories. In some cases more than one time slots level will be created. For example, when the data is stored frequently in this repository.
3. Data level: Files are stored here as described in section 4.2.3 (files in original data format or files in plain text format or similar (store after transformation to make easier the use of information with different technologies in the future).

Example: Figure 9 presents the proposed structure considering an example (a relational database for the KPIs repository).

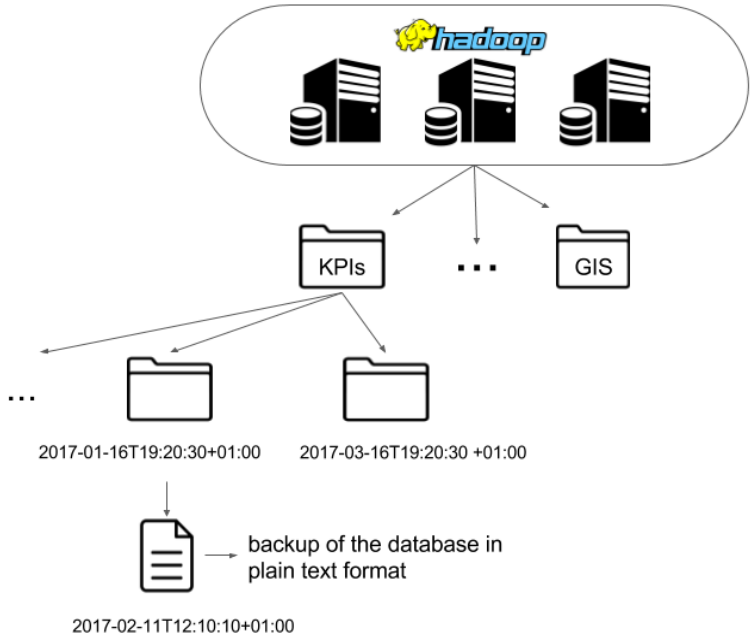


Figure 9: Example of historical repository structure

Structural Data Repository

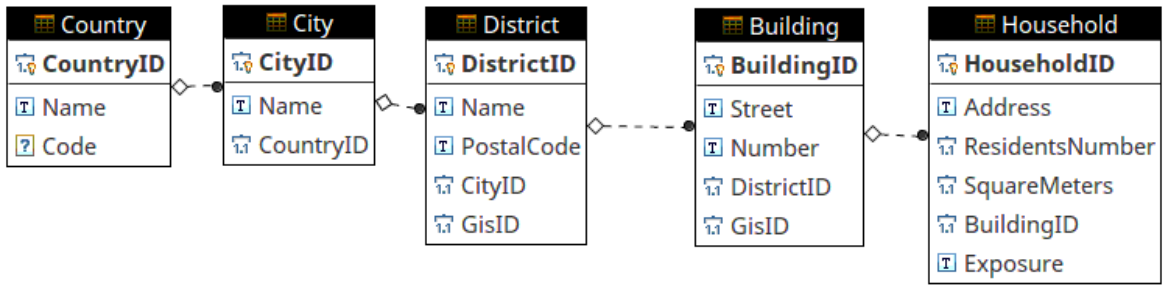


Figure 10: Structural data model for the demonstrator

Country

This table stores the countries that are in the system.

- **CountryID** INT: Country's auto numeric identifier inside the system.
- **Name** NVARCHAR: Descriptive human readable name of the Country.
- **Code** NVARCHAR: country code (e.g. *es* for *Spain*).

City

This table stores the cities that are in the system.

- **CityID** INT: City's auto numeric identifier inside the system.
- **Name** NVARCHAR: Descriptive human readable name of the City.
- **CountryID** INT: Reference to the country the city belongs to.

District

Districts are differentiated zones inside a city. This table stores the districts that are in the system.

- **DistrictID** INT: District's auto numeric identifier inside the system.
- **Name** NVARCHAR: Descriptive human readable name of the District.
- **PostalCode** NVARCHAR: a series of characters included in a postal address for the purpose of sorting mail.
- **CityID** INT: Reference to the city the district belongs to.
- **GisID** NVARCHAR: the identifier the district has in the GIS repository for interoperability purposes.
-

Building

This table stores the buildings that are in the system.

- **BuildingID** INT: Building's auto numeric identifier inside the system.
- **Street** NVARCHAR: Descriptive human readable name of the street where the building is placed.
- **Number** NVARCHAR: The number in the street (regarding to the address) where the building is placed.
- **DistrictID** INT: Reference to the district the buildings belongs to.
- **GisID** NVARCHAR: the identifier the building has in the GIS repository for interoperability purposes.

Household

The household are the specific homes that are inside a building. This table stores the households that are in the system.

- **HouseholdID** INT: Household's auto numeric identifier inside the system.
- **Address** VARCHAR: full address of the household.
- **ResidentsNumber** INT: amount of residents living in the household.
- **SquareMeters** INT: size of the household in m².
- **ConsumptionTarget** FLOAT: the amount of energy consumption the household is suposed to consume.
- **SelectedCostPerKw** FLOAT: Cost/KW of the household, established by the user.
- **Exposure** VARCHAR: The orientation/facing/exposure of the household (N for north, E for East, S for South, W for West, NW for Northwest, ESW for East-South-West, etc.).

GIS Structural Data Repository

Structural Data Repository in SmartEnCity will be based on the CityGML standard data model. This section describes the main elements of this data model that will be used in the project.

CityGML identifies several modules for the definition of the main city elements. Following figure (Figure 11) represents the modules defined in CityGML. The most representative modules for urban areas are rounded.

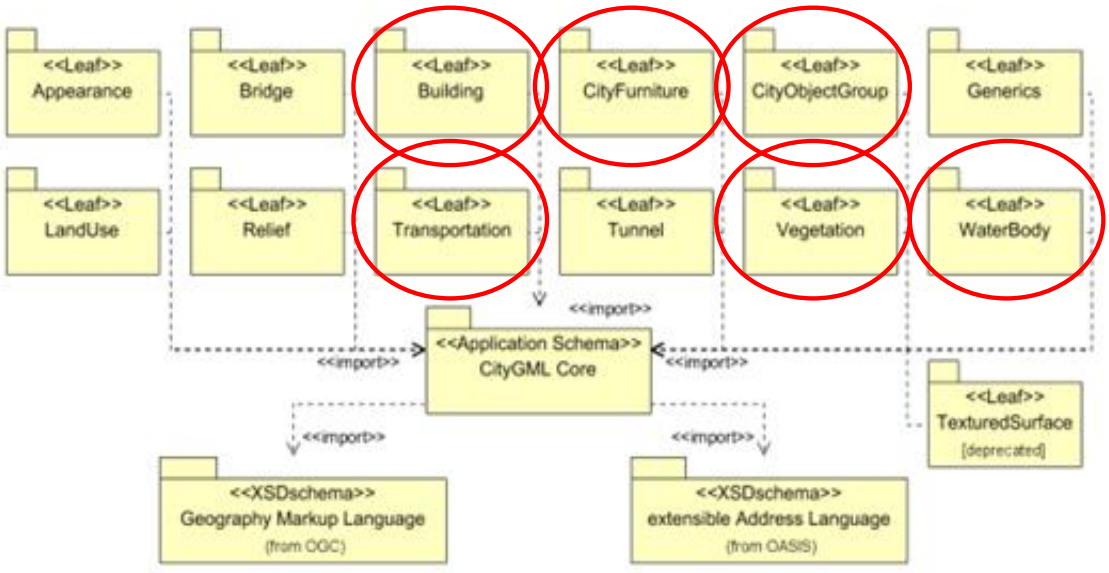


Figure 11 CityGML Modules

Building is the main element of a City Model, a city is mainly composed of buildings. Transportation module represents urban elements such as, roads, parking areas and rail networks. Vegetation module represents green spaces, from solitary trees to gardens and forest areas. WaterBody module is used to represent rivers, lakes or ponds. Other elements located in the city such as bus stop, railway station or traffic lights are represented as

CityFurniture. Different elements from the previous modules can be grouped into a CityObjectGroup in order to represent an area, district or city.

Building

The main element of the city model is the building. A simplified version of the UML diagram of the building model in CityGML is presented in the following figure (Figure 12). Building model in CityGML includes building parts and elements such as rooms, installations or furniture. However for the level of definition required in the project those elements will not be represented in the city model. Bellow the main data which represents the building are listed and detailed.

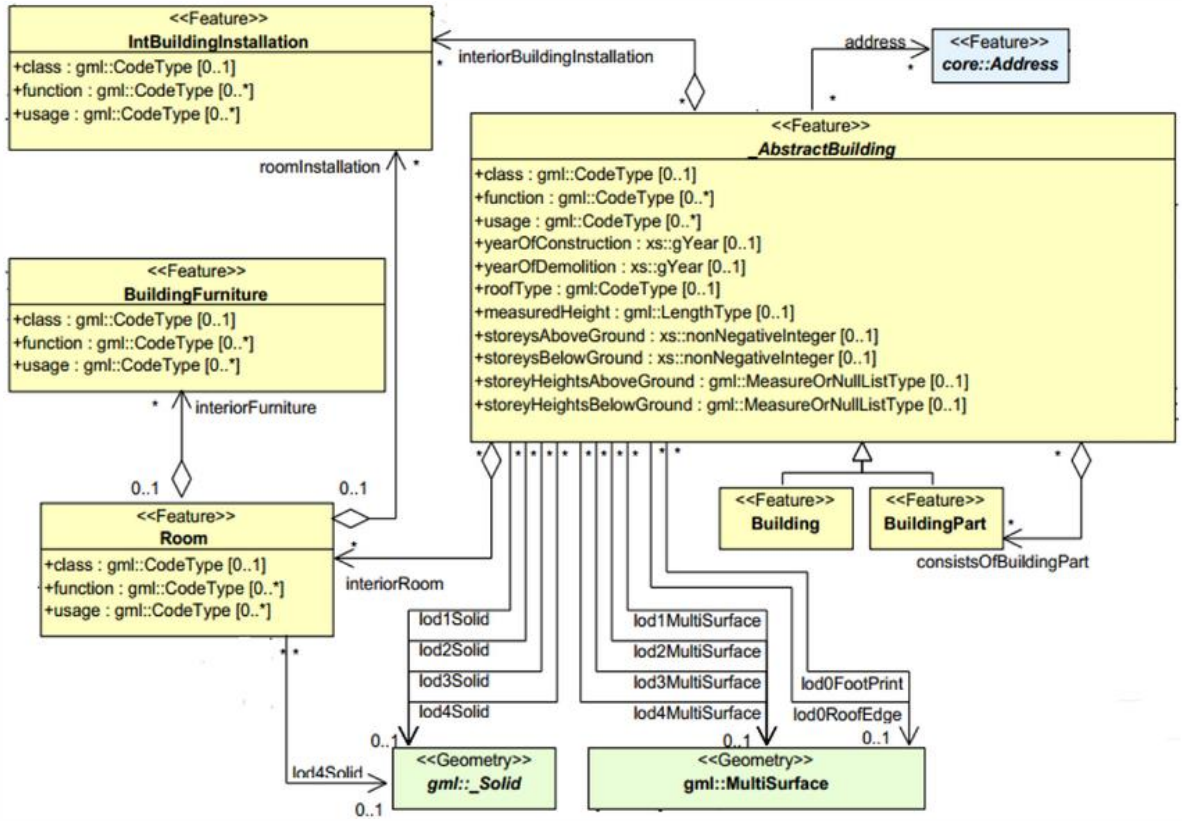


Figure 12 Simplified version of UML diagram of building model in CityGML.

Property name	Type	Description
Name	String	Name of the Building, inherited from CityObject
Description	String	Description of the Building, inherited from CityObject
Class	gml:CodeType	Class of the building (e.g. School, Church, Business or Industry)
Function	gml:CodeType	Function of the building (e.g. Residential, Hotel, Public building or Museum)

Usage	gml:CodeType	Current use of the building, it could be different from the purpose for which it was built
YearOfConstruction	Year	Year in which the building was constructed
YearOfDemolition	Year	Year in which the building was demolished, if applicable.
RoofType	gml:CodeType	Type of the roof of the building (e.g. Flat, Gabled or Hipped)
MeassuredHeight	Numeric	Size of the building from the lowest element to the highest element.
StoreysAboveGround	Numeric	Number of storeys above the ground level
StoreysBelowGround	Numeric	Number of storeys under the ground level
StoreyHeightsAboveGround	Numeric	List of storey heights above the ground level (from nearest to farthest to ground floor)
StoreyHeightsBelowGround	Numeric	List of storey heights below the ground level (from nearest to farthest to ground floor)

Other information stored related to the buildings is the **address** of the building, including Country, City, Postal code and street name and number.

Geometry of the building is represented by the footprint in 2D for Level of Detail 0 (LoD0) or a georeferenced 3D geometric representation of the building envelope for higher levels (from LoD1 to LoD4). It can be represented by a Solid or a Multisurface.

Transportation

Transportation in CityGML represents roads, tracks, rails, and squares. Main data representing a transportation object in CityGML are described in the following table:

Property name	Type	Description
Name	String	Name of the Transportation object, inherited from CityObject
Description	String	Description of the Transportation object, inherited from CityObject
Class	gml:CodeType	Class of the transportation object
Function	gml:CodeType	Function of the Transportation object
Usage	gml:CodeType	Current use of the Transportation object

The geometry of the transportation objects are represented by 2D lines in (LoD0) establishing a linear network. For higher levels of details an explicit surface geometry is used represented by a Multisurface.

Vegetation

Main data representing a vegetation area in CityGML are described in the following table:

Property name	Type	Description
Name	String	Name of the vegetation area, inherited from CityObject
Description	String	Description of the vegetation area, inherited from CityObject
Class	gml:CodeType	Plant community of the vegetation area
Function	gml:CodeType	Intended purpose of the vegetation area
Usage	gml:CodeType	Real purpose of the vegetation area
AverageHeight	Numeric	Average relative vegetation height

The geometry of the vegetation areas are represented by Multisolid or Multisurface geometries for LoD1 to LoD4. LoD0 is not defined for vegetation objects.

WaterBody

WaterBody in CityGML represents rivers, canals, lakes, and basins. Main data representing a water body in CityGML are described in the following table:

Property name	Type	Description
Name	String	Name of the waterbody element, inherited from CityObject
Description	String	Description of the waterbody object, inherited from CityObject
Class	gml:CodeType	Classification of the object, e.g. lake, river, or fountain
Function	gml:CodeType	Purpose of the waterbody object like, for example national waterway or public swimming
Usage	gml:CodeType	Current use of the waterbody object

The geometry of the waterbody objects is represented by 2D surfaces (MultiCurve or MultiSurface). From LoD1 to higher levels water bodies may also be modelled as water filled volumes represented by Solids.

CityFurniture

City furniture objects are immovable objects like lanterns, traffic lights, traffic signs, flower buckets, advertising columns, benches, delimitation stakes, or bus stops. Main data representing a cityfurniture object in CityGML are described in the following table:

Property name	Type	Description
Name	String	Name of the object, inherited from CityObject

Description	String	Description of the object, inherited from CityObject
Class	gml:CodeType	Classification of the object like traffic light, traffic sign, delimitation stake, or garbage can,
Function	gml:CodeType	Determines to which thematic area the city furniture object belongs
Usage	gml:CodeType	Real purpose of the object

City furniture objects can be represented in city models with their specific geometry.

Building

The building model is described in the table shown in Figure 13. The three CityGML classes AbstractBuilding, Building and BuildingPart are merged into the single table Building. The hierarchy within a building is realized by the foreign key building_parent_id, it refers to the superordinate building and contains null if it does not exist. This allows creating a tree structure within a building. In the same way, the building_root_id indicates the top level of a building, the root.

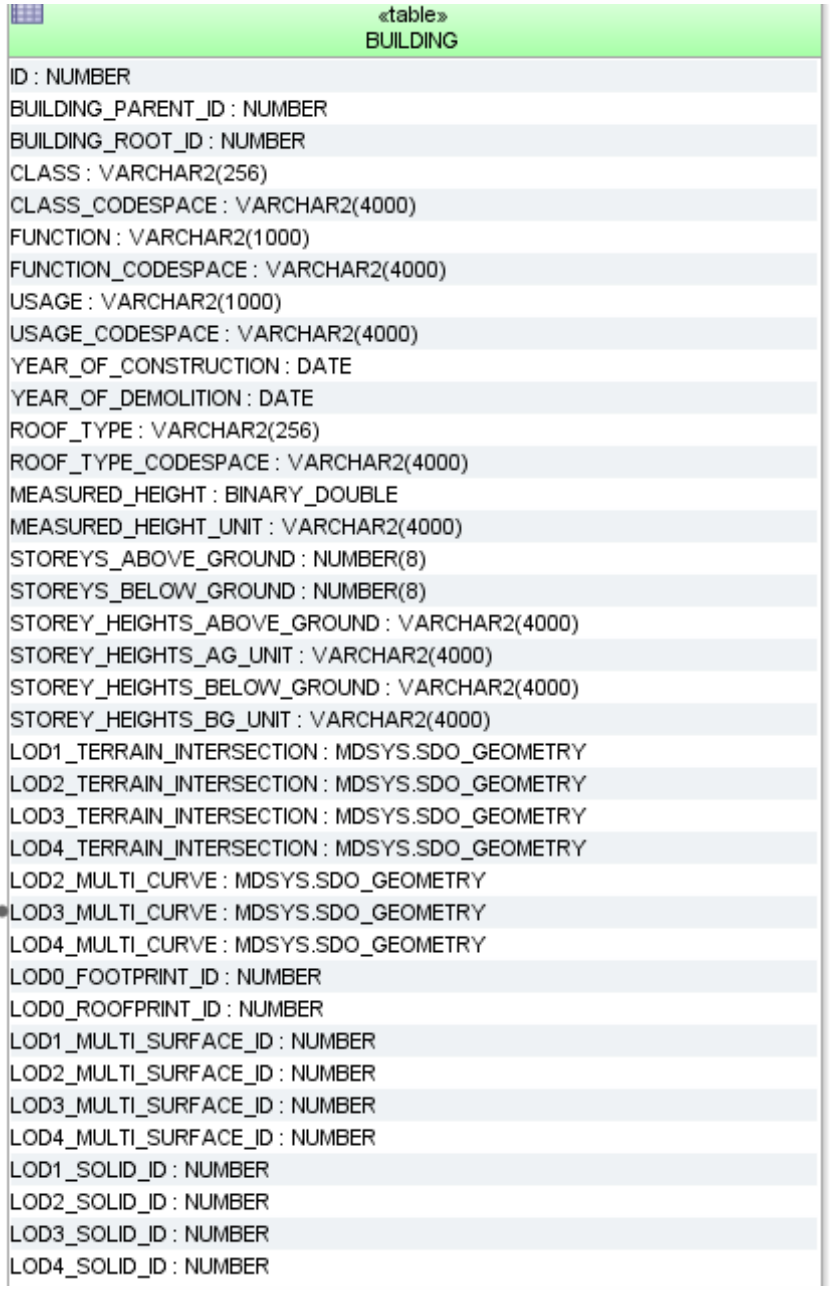


Figure 13: Building database schema

As can be seen in Figure 14 for the building with id 804, multiple building parts are defined (818, 834, 852, 805). Each building part has semantic information such as year of construction, measured height or storeys above ground defined. In addition, the foreign key to the lod2 geometry is also represented (3322, 3368, 3420, 3285).

The meaning and the name of most fields are identical the attributes described in the CityGML standard UML diagrams. For every attribute including measure information such as measuredHeight or storeyHeightsAboveGround, another column is provided indicating the unit of the measurement.

In order to represent the geometry, several foreign keys are defined named lodx_multi_sruface_id ($1 \leq x \leq 4$), and lodx_solid_id ($1 \leq x \leq 4$) which refer to entries in the surface_geometry table and represent each LoD's surface geometry.

	id integer	building_parent_id integer	building_root_id integer	class characte	function character	usage characte	year_of_construction date	roof_type character varyi
1	804		804					
2	818	804	804				1970-01-01	
3	834	804	804				1969-01-01	
4	852	804	804				1969-01-01	
5	805	804	804				1969-01-01	

	measured_height double precision	measured_height_unit character varying(4000)	storeys_above_ground numeric(8,0)	storeys_below_ground numeric(8,0)	lod2_solid_id integer	lod3 inte
1						
2	24.3967278478319		8		3322	
3	15.432256797872		6		3368	
4	16.4710391569344		5		3420	
5	24.3587249070455		8		3285	

Figure 14: Building example in database

Thematic surface

The table thematic_surface represents thematic boundary features in CityGML. Each boundary surface in CityGML has a number of subclasses that represent different types of surfaces. In order to represent the type of surface the column objectclass_id is used in the table thematic_surface. Allowed integer values:

Boundary type integer	Boundary type description
30	CeilingSurface
31	InteriorWallSurface
32	FloorSurface
33	RoofSurface
34	WallSurface
35	GroundSurface
36	ClosureSurface
60	OuterCeilingSurface
61	OurterFloorSurface

The relation between buildings and the corresponding boundary surfaces results from the foreign key building_id within thematic_surface table, which refers to the ID of the respective building.

As can be seen in the Figure 15, the boundary surfaces of the building 818 are the following ones. Most of them are WallSurfaces and one of them is a RoofSurface. Each of the thematic surface is connected with the surface geometry table using the lod2_multi_surface_id.

	id integer	objectclass_id integer	building_id integer	room_id integer	building_installation_id integer	lod2_multi_surface_id integer	lod3_multi_surface_id integer	lod4_multi_surface_id integer
1	819	34	818			3338		
2	820	34	818			3340		
3	821	34	818			3342		
4	822	34	818			3344		
5	823	34	818			3346		
6	824	34	818			3348		
7	825	34	818			3350		
8	826	34	818			3352		
9	827	34	818			3354		
10	828	34	818			3356		
11	829	34	818			3358		
12	830	34	818			3360		
13	831	34	818			3362		
14	832	34	818			3364		
15	833	33	818			3366		

Figure 15: Thematic surface example in database

Surface geometry

The surface_geometry, which for example geometrically defines a roof, should at the same time be a part of the volume geometry of the parent feature the roof belongs to.

As can be seen in the Figure 16 the surface geometry of the ids (3338, 3340, 3342, 3344, 3346 and 3348) is presented. The geometry columns stores the coordinates of each polygon.

	id integer	gmlid character varying(256)	parent_id integer	root_id integer	is_solid numeric	is_composite numeric	is_triangulated numeric	is_xlink numeric	is_reverse numeric	solid_g geome
1	3338	UUID d9080ce2-6803-4db8-8d13-ee04b2eb8db9		3338	0	0	0	0	0	
2	3339	PolyID7120 4021 2660 4908	3338	3338	0	0	0	1	0	
3	3340	UUID 858705f4-ec7d-4b91-b941-f5c0a2989c9e		3340	0	0	0	0	0	
4	3341	PolyID8391 9092 7114 3836	3340	3340	0	0	0	1	0	
5	3342	UUID e56be9e9-457a-462e-bf74-e82a234fe6d3		3342	0	0	0	0	0	
6	3343	PolyID8205 2880 5865 3581	3342	3342	0	0	0	1	0	
7	3344	UUID 9bf07558-c53f-40f0-b046-1507be2d0b33		3344	0	0	0	0	0	
8	3345	PolyID3983 6572 3334 5200	3344	3344	0	0	0	1	0	
9	3346	UUID 38b8a21f-58e2-484c-846e-4bd89089a1f9		3346	0	0	0	0	0	
10	3347	PolyID137 6026 1907 9349	3346	3346	0	0	0	1	0	
11	3348	UUID 1604bee1-4397-4e1d-b3f1-43a3c5f373d8		3348	0	0	0	0	0	
12	3349	PolyID1842 2184 4610 2336	3348	3348	0	0	0	1	0	

	st_astext text
1	
2	POLYGON Z ((526421.567000001 4744453.349 514.573489999995,526421.567000001 4744453.3
3	
4	POLYGON Z ((526423.925000002 4744446.01900001 514.479059999967,526423.925000002 4744
5	
6	POLYGON Z ((526419.262 4744444.518 514.517379999999,526419.262 4744444.518 538.84065
7	
8	POLYGON Z ((526418.283 4744444.203 514.52,526418.283 4744444.203 538.840657978836,52
9	
10	POLYGON Z ((526417.708 4744442.476 514.50768000002,526417.708 4744442.476 538.840657
11	
12	POLYGON Z ((526419.239000001 4744440.424 514.469459999968,526419.239000001 4744440.4

Figure 16: Surface geometry example in database

GIS Repository

This repository is divided in four categories (see Figure 17) corresponding to: urban structures, street furniture, parks and gardens, urban mobility:

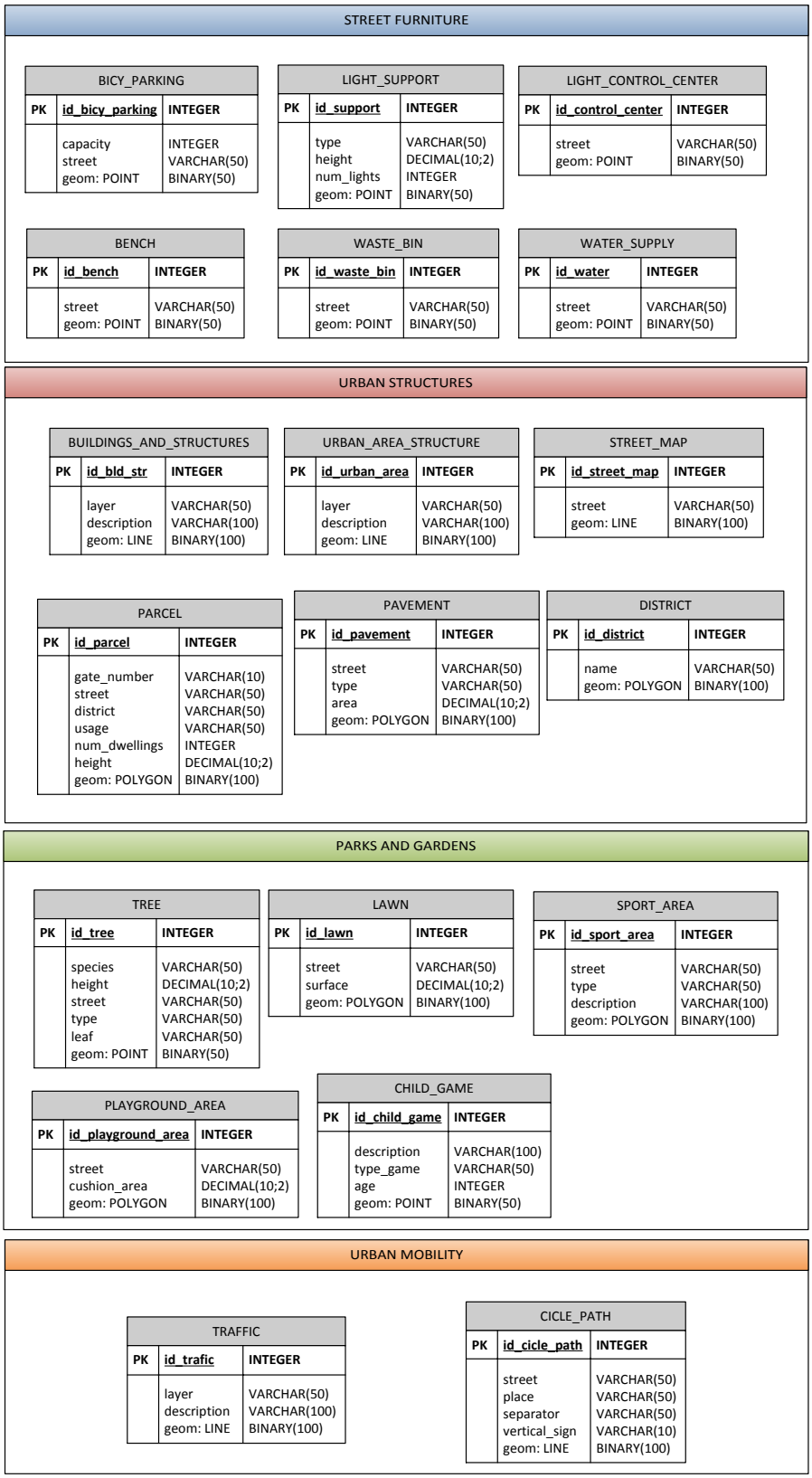


Figure 17 Structure of GIS Repository

Urban Structures

BUILDINGS_AND_STRUCTURES (LINE LAYER)

This layer will store the information about the buildings elements such as: roofs, dividing walls, inner courtyards…

- ***id_bld_str*** (INT): buildings and structures numeric identifier.
- ***layer*** (VARCHAR): identifier of the official 1:500 scale cartography of the Vitoria-Gasteiz City Council containing these elements.
- ***description*** (VARCHAR): Description of the elements contained.
- ***geom*** (BINARY): layer's geometry.

URBAN_AREA_STRUCTURE (LINE LAYER)

This layer will store the information about the elements defining urban areas such as: curbs, sidewalks, access ramps…

- ***id_urban_area*** (INT): urban area structure numeric identifier.
- ***layer*** (VARCHAR): identifier of the official 1:500 scale cartography of the Vitoria-Gasteiz City Council containing these elements.
- ***description*** (VARCHAR): Description of the elements contained.
- ***geom*** (BINARY): layer's geometry.

STREET_MAP (LINE LAYER)

This table will store the street edges with the corresponding street names.

- ***id_street_map*** (INT): street map numeric identifier.
- ***street*** (VARCHAR): name of the street.
- ***geom*** (BINARY): layer's geometry.

PARCEL (POLYGON LAYER)

In this layer it is saved the significant information about the cadastral parcels in the Coronación district.

- ***id_parcel*** (INT): parcel numeric identifier.
- ***gate_number*** (VARCHAR): Number to identify each portal.
- ***district*** (VARCHAR): district name.
- ***usage*** (VARCHAR): type of usage of the building.
- ***num_dwellings*** (INT): Number of dwellings in the parcel.
- ***height*** (DECIMAL): Parcel height.
- ***geom*** (BINARY): layer's geometry.

PAVEMENT (POLYGON LAYER)

In this table it is saved the different types of sidewalks pavement.

- ***id_pavement*** (INT): pavement's numeric identifier.
- ***street*** (VARCHAR): name of the street where the pavement is located.
- ***area*** (DECIMAL): extent (in m^2) of the element.
- ***geom*** (BINARY): layer's geometry.

DISTRICT (POLYGON LAYER)

It will contain the district delimitation.

- ***id_district*** (INT): district's numeric identifier.
- ***name*** (VARCHAR): district name.
- ***geom*** (BINARY): layer's geometry.

Street furniture

BICY_PARKING (POINT LAYER)

This table will contain the location of the bicycle parking lots in the study area.

- ***id_bicy_parking*** (INT): bicycle parking numeric identifier.
- ***capacity*** (INT): Number of places.
- ***street*** (VARCHAR): name of the street where the bicycle parking is located.
- ***geom*** (BINARY): layer's geometry.

LIGHT_SUPPORT (POINT LAYER)

This table will contain the location of the light supports in the study area.

- ***id_support*** (INT): light support numeric identifier.
- ***type*** (VARCHAR): Type of the support.
- ***height*** (DECIMAL): light support height.
- ***num_lights*** (INT): number of lights on the support.
- ***geom*** (BINARY): layer's geometry.

LIGHT_CONTROL_CENTER (POINT LAYER)

This table will contain the location of the light control centers in the study area.

- ***id_bench*** (INT): bench numeric identifier.
- ***street*** (VARCHAR): name of the street where the light control center is located.
- ***geom*** (BINARY): layer's geometry.

WATER_SUPPLY (POINT LAYER)

This table will contain the location of the water supplies in the study area.

- ***id_water*** (INT): water supply numeric identifier.
- ***street*** (VARCHAR): name of the street where the water supply is located.
- ***geom*** (BINARY): layer's geometry.

BENCH (POINT LAYER)

This table will contain the location of the benches in the study area.

- ***id_bench*** (INT): bench numeric identifier.
- ***street*** (VARCHAR): name of the street where the bench is located.
- ***geom*** (BINARY): layer's geometry.

WASTE_BIN (POINT LAYER)

This table will contain the location of the waste bins in the study area.

- ***id_waste_bin*** (INT): waste bin numeric identifier.
- ***street*** (VARCHAR): name of the street where the waste bin is located.

- ***geom*** (BINARY): layer's geometry.

Parks and gardens

TREE (POINT LAYER)

This table will store the location of the trees in the study area.

- ***id_tree*** (INT): tree numeric identifier.
- ***species*** (VARCHAR): name of the tree species.
- ***height*** (DECIMAL): tree height.
- ***street*** (VARCHAR): name of the street where the tree is located.
- ***type*** (VARCHAR): Type of the tree: leafy or conifer.
- ***leaf*** (VARCHAR): Type of tree leafs: perennial or deciduous.
- ***geom*** (BINARY): layer's geometry.

LAWN (POLYGON LAYER)

This table will contain the landscaped areas in the study area.

- ***id_lawn*** (INT): landscaped area numeric identifier.
- ***street*** (VARCHAR): name of the street where the landscaped area is located.
- ***surface*** (DECIMAL): extent (in m^2) of the landscaped area.
- ***geom*** (BINARY): layer's geometry.

PLAYGROUND_AREA (POLYGON LAYER)

This table will contain the playground areas in the study area.

- ***id_playground_area*** (INT): playground area numeric identifier.
- ***street*** (VARCHAR): name of the street where the playground area is located.
- ***cushion_area*** (DECIMAL): extent (in m^2) of the cushioned area.
- ***geom*** (BINARY): layer's geometry.

CHILD_GAME (POINT LAYER)

This table contains information about each game located in each playground area.

- ***id_child_game*** (INT): child game numeric identifier.
- ***description*** (VARCHAR): description of the game.
- ***type_game*** (VARCHAR): type of the game.
- ***age*** (INT): age ranges to use the game.
- ***geom*** (BINARY): layer's geometry.

SPORT_AREA (POLYGON LAYER)

This table will contain the sport areas in the study area.

- ***id_sport_area*** (INT): sport area numeric identifier.
- ***street*** (VARCHAR): name of the street where the sport area is located.
- ***type*** (VARCHAR): type of sport area, such as: sandbox, sports court, fronton…
- ***description*** (VARCHAR): description of the area, such as: football field, basketball court…
- ***geom*** (BINARY): layer's geometry.

Urban mobility

TRAFFIC (LINE LAYER)

This table stores the information about the roads horizontal signaling.

- ***id_traffic*** (INT): traffic numeric identifier.
- ***layer*** (VARCHAR): identifier of the official 1:500 scale cartography of the Vitoria-Gasteiz City Council containing these elements.
- ***description*** (VARCHAR): Description of the elements contained.
- ***geom*** (BINARY): layer's geometry.

CYCLE_PATH (LINE LAYER)

In this table it will be stored the information about the cycle path route.

- ***id_cicle_path*** (INT): cycle path numeric identifier.
- ***street*** (VARCHAR): name of the street where the cycle path is located.
- ***place*** (VARCHAR): indicates where the cycle path runs, such as: sidewalk, road, own lane…
- ***separator*** (VARCHAR): this field indicates if the cycle path is physically separated from the other road elements.
- ***vertical_sign*** (VARCHAR): Indicates if there is a vertical sign to signalize the cycle path.
- ***geom*** (BINARY): layer's geometry.

Configuration Repository

In this repository all the transversal components that interact with the rest of the layers of the platform are included.

Security support

The platform have to guarantee certain issues of security along al the layers, modules and components. Some of the features to be supported are:

- Support for the authentication and authorization
- Manage the access to any data or any component of the platform. On one hand at low level, by managing the access to sensors, devices and infrastructure elements and on the other hand, by managing the access of the applications and verticals.
- Guarantee the confidentiality of the data in the way that each role can only access to the data allowed to this role.
- Define and manage security policies
- Provide a web interface to manage the users, roles and components
- Support different mechanisms of authentication as users, passwords, tokens, OAuth, digital certificates… and any identification mechanism.
- Integration with other external user's repositories
- Adaptation and configuration according to each city needs

Profiling

The platform has to guarantee the privacy and the security of all data stored and managed by the different components and modules of the platform.

The platform has to guarantee the secure and right sending and reception of the information among different components of the platform and among the external services and the platform. At least in both side of the data transference the privacy and security has to be achieved.
The platform must guarantee different roles and type of users and different level of access to data. The platform will allow the access or deny it applying the adequate level of permissions to each role or application.
The consumers of the data could be applications, verticals, APIs, users, services…
There are three levels of security:

- Access to data: limiting the data that each users can access
- Access to the platform internal components: limiting the access to internal modules, components and reports.
- Functionality: limiting the features and functionalities of each role and user.

Maintenance and support

In order to provide a continuous service and performance of the platform some maintenance services has to be offered:

- Preventive maintenance: Managing some KPIs of performance in order to evaluate the behavior and the needs of maintenance. Generating maintenance and preventive actions.
- Corrective maintenance: Manage, apply corrective actions and solve the possible errors, notifications and alarms that may occur in the platform. Possibility of using smartphone applications and mobile devices for repairing and supporting the corrective actions.

Real Time Repository

Real Time Repository is the database where all the data coming from sensors, devices and infrastructure of the city are stored before being treated and processed. Most of the Real Time Repository are based on Time Series. A Time Series is a series of data points indexed in time order. Most commonly, a Time Series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of data in discrete time. For example: the temperature of a home taken each 15 minutes or the position of a car taken each minute.

As the time sequence is different for each type of data coming from the sensors and devices there will be type of tables where each data is stored. Time Series analysis comprises methods for analyzing Time Series data in order to extract meaningful statistics and other characteristics of the data. Time Series forecasting is the use of a model to predict future values based on previously observed values. While regression analysis is often employed in such a way as to test theories that the current values of one or more independent Time Series affect the current value of another time series, this type of analysis of Time Series is not called " Time Series Analysis", which focuses on comparing values of a single Time Series or multiple dependent time series at different points in time

Time Series data have a natural temporal ordering. This makes Time Series analysis distinct from cross-sectional studies, in which there is no natural ordering of the observations (e.g. explaining people's wages by reference to their respective education levels, where the individuals' data could be entered in any order). Time Series analysis is also distinct from spatial data analysis where the observations typically relate to geographical locations (e.g.

accounting for house prices by the location as well as the intrinsic characteristics of the houses). A stochastic model for a Time Series will generally reflect the fact that observations close together in time will be more closely related than observations further apart. In addition, Time Series models will often make use of the natural one-way ordering of time so that values for a given period will be expressed as deriving in some way from past values, rather than from future values. Time Series analysis can be applied to real-valued, continuous data, discrete numeric data, or discrete symbolic data.

As stated in the demonstrator description, the platform gets external data from energy monitoring systems. In this case, the energy monitoring system is deployed in a Mondragon University laboratory and measures electrical energy consumption of the lighting of the laboratory. It also measures ambient temperature. The energy monitoring system can be fitted with up to 11 sensors, each one measuring different magnitudes (electrical consumption, gas consumption, water consumption, ambient temperature, etc.).

In order to store this information, we use a TSDB solution built on top of a SQL relational database. We use PostgreSQL as the base SQL solution, complemented with the TimeScale TSDB extension. PostgreSQL not only supports traditional relational datatypes (integer, varchar, text, timestamp, etc.), but also JSON and JSONB data types and operations, which enables storing non-relational/non-structured data in those columns.

Thus, the tables comprising the Real Time Data repository data model for the demonstrator are described below, along with their columns and Data Types.

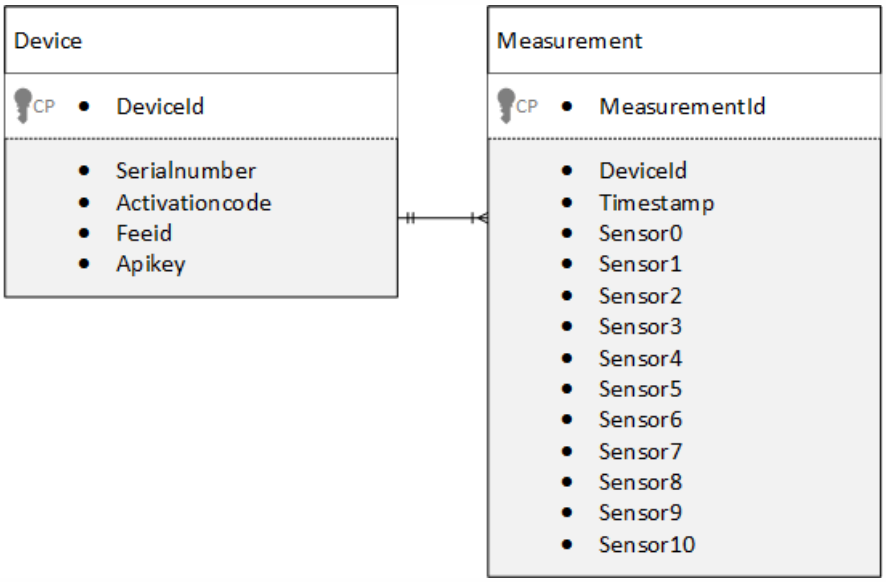


Figure 18: Real time repository data model for the demonstrator

Device

This table stores details about installed energy monitoring systems. These systems need to be provisioned, activated and given an API Key to be able to send data to the Real Time repository (actually, to the HTTP REST interoperability adapter that interacts with the Real Time repository itself). This table stores all the values needed for the provisioning and activation process.

- **DeviceId** INTEGER: Device's auto numeric identifier inside the system.
- **Serialnumber** CHARACTER VARYING(128) NOT NULL: serial number of the

energy monitoring device.

- **Activationcode** CHARACTER VARYING(128) NOT NULL: activation code that the device needs to provide to be activated.
- **Feedid** CHARACTER VARYING(128) NOT NULL: ID of the data stream the energy monitoring device will use to send its data.
- **Apikey** CHARACTER VARYING(128) NOT NUL: the API Key the energy monitoring device needs to use to store data in the Real Time repository.

Measurement

This table stores the individual values of the measurements sent by the energy monitoring devices.

- **MeasurementID** INT: Measurement's auto numeric identifier inside the system.
- **DeviceId** INTEGER NOT NULL: Foreign key referencing DeviceId column in Device table.
- **Timestamp** TIMESTAMP NOT NULL: time stamp (in Universal Coordinated Time format, without time zone) for the measurement.
- **Sensor0** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the first input of the energy monitoring device. If the energy monitoring device did not send any value for this sensor (e.g., it does not have a sensor attached to this input), '0.0' is stored.
- **Sensor1** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the second input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.
- **Sensor2** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the third input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.
- **Sensor3** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the fourth input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.
- **Sensor4** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the fifth input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.
- **Sensor5** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the sixth input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.
- **Sensor6** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the seventh input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.
- **Sensor7** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the eighth input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.
- **Sensor8** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the nineth input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.
- **Sensor9** DOUBLE PRECISION NOT NULL: Measured value for the sensor attached to the tenth input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.
- **Sensor10** DOUBLE PRECISION NOT NULL: Measured value for the sensor

attached to the eleventh input of the energy monitoring device. Again, 0.0 is stored if a value is not received for this input.

A TimeScale hypertable is created based on the Measurement table (using *create_hypertable* command), partitioned on the "timestamp" column, setting a *chunk_time_interval*[3] of 1 week (604,800,000,000 microseconds, which is the unit TimeScale uses for event time, when using a TIMESTAMP or TIMESTAMPTZ data types in the original table):

```
SELECT create_hypertable('Measurement, 'Timestamp',
                         chunk_time_interval => 604800000000);
```

Open/external Data

Open Data is about transparency, about giving citizens access to the data their taxes help create and it is about innovation by enabling software developers to transform that data into useful applications that make city services available anytime, anywhere. Some cities have already started working on becoming a city with distributed and public data.

In this context, data is a valuable asset and an essential resource for almost any activity in our society that everyone assumes that it should be shared. Proper management of all the data that occurs in the daily life of the city through automated processing will be pivotal to understanding what is happening in our cities and to make the right decisions to ensure optimal management of resources, as well as to meet the demands of its people efficiently.

In the context of SmartEnCity, Open Data can be used in two completely different contexts. On the one hand third party Open Data can be an additional source of information to the platform (gathered through the Acquisition/Interconnection Layer). On the other hand, the platform can offer some of its own data to third parties through existing Open Data interoperability standards.

In the context of the demonstrator, third party Open Data will be consumed to provide additional environmental information in order to give more precise recommendations to final users. Open Data related to climate and meteorological historic data and forecasts for the following hours/days will be used. AEMET, Spanish Meteorological Agency, has an Open Data portal (http://www.aemet.es/en/datos_abiertos/AEMET_OpenData) where it provides documentation about the data they provide and several ways to consume its data. In the case of the demonstrator, we will consume their HTTP(S) REST API.

That data will then be processed and stored in either of two repositories. The weather forecast data will be added to the Vertical Data Repository, to provide better recommendations to final users). The weather historic data will be added to the Historical Data Repository, to be able to calculate those KPIs that depend on it, and to produce added-value reports, dashboards and applications with mashed-up data that include climate and meteorological data.

[3] Interval, in event time, that each chunk of data covers.

5.2.3 User Guide

A visor guide for the demonstrator is available online. The guide presents the necessary support to show data from the example implemented. Important information about access and the visor guide next:

During the development phase the visor for the Vitoria demonstrator will be accessible through this URL: http://geoservergis.azurewebsites.net/gisviewer/viewer.do

Where login is needed, please use the following information for access to the platform to view the information.

User: demo

Password: smartencity

For more detailed of the solution and to access specific repositories please contact project partners listed on those sites.

Access guide (Visor guide)

The map viewer provides access, throught OGC standards, to the set of geographic information and has been designed for viewing and querying that information and creating associated reports in a easy and simple way.

The main functionalities offered by the viewer include navigation (zoom in, zoom out and scroll through the map), enable and disable layers, get information about an item that is querible by clicking on it and create KPI reports.

The browser in which geographic information is displayed occuppies most of the screen and allows navigation in 2D. For optimal visualization it is recommended to use updated version of Firefox, Chrome or Internet Explorer.

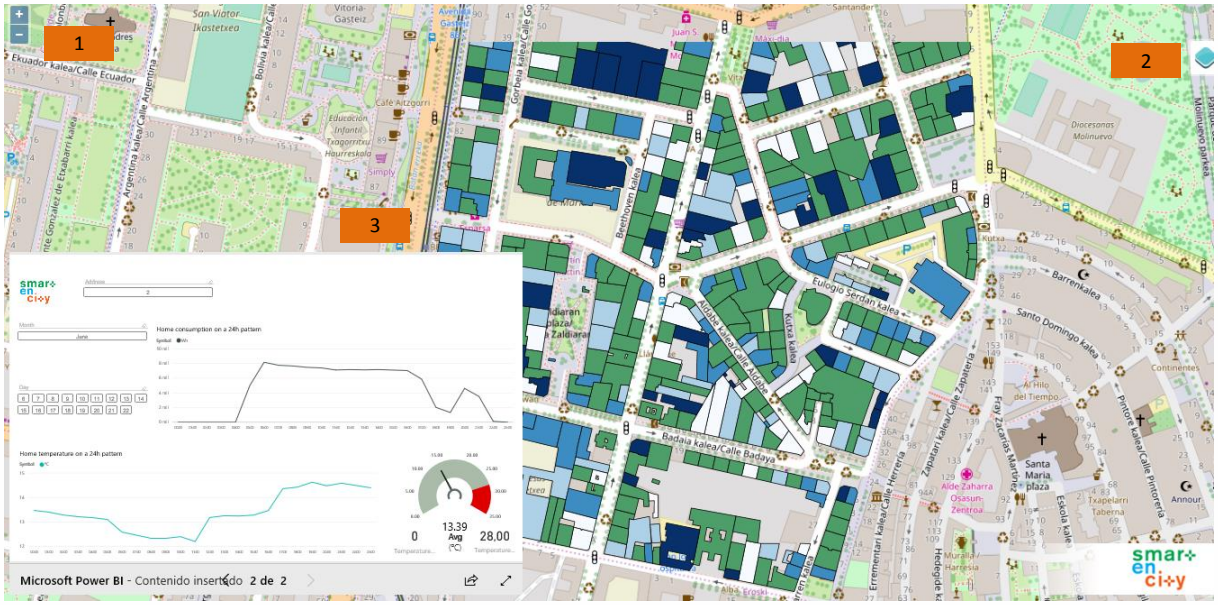


Figure 19: Viewer elements: zoom tools (1), map layers (2) and reports window (3)

The following details the viewer functionalities:

1) Navigation

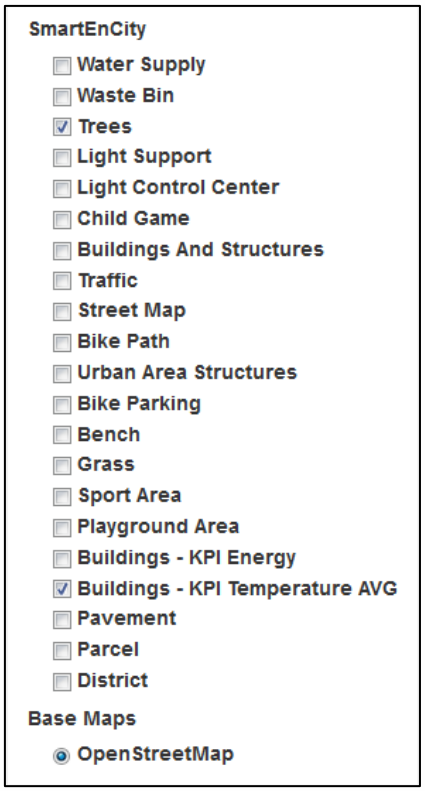
Navigation around the map can be done using one on the following options:

- Mouse navigation
  It is the most intuitive form of navigation. It includes:
  - Move the map: Click with the left mouse button and drag the map in the desired direction: up, down, right and left.
  - Zoom map at cursor location: Double click on the point of interest with the left mouse button
  - Zoom map at cursor location: 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.
- Zoom tools
  Zoom tools appear in the top right if the viewer (No. 1). It includes:
  - Zoom in: Click the Plus (+) button to zoom in on the map.
  - Zoom out: Click the Minus (-) button to zoom out on the map.

2) Map layers

Map layers menu allows seeing what layers are available in order to select them and add them to the map. This menu is accessed by placing the cursor over the icon on the top left of the viewer (No.2).

The list of layers available for visualization is displayed immediately. Beside each layer, there is a check box that is used to turn a layer on or off .



In addition to the thematic layers, the viewer provides a base map as a background map by default (OpenStreetMap). This map serves as support for locating information related to the territory and cannot be deactivated.

3) Map tip

The viewer allows identifying and visualizing the alphanumeric information of the layers loaded on the map. 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 to get information. A window with the information of the geographic object will be displayed. In order to close the window, click on the icon ✕ on the top left.

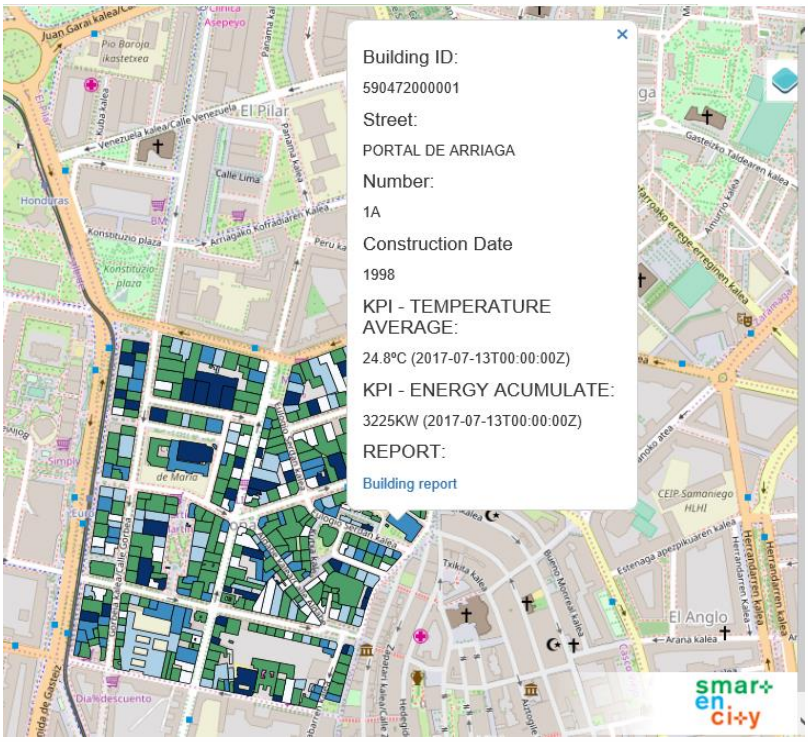


Figure 20: Map tip

4) Reports windows

The reports window appears on the bottom right-hand corner of the viewer. This window displays graphic and alphanumeric data of the KPIs in an easy-to-use interface. The information is displayed according to the data selected in the different fields (address, date, etc.). In order to see information about an specific time once the graphs are displayed with the chosen criteria, run the mouse over the graph. A pop-up window will be displayed immediately with the related alphanumeric information.

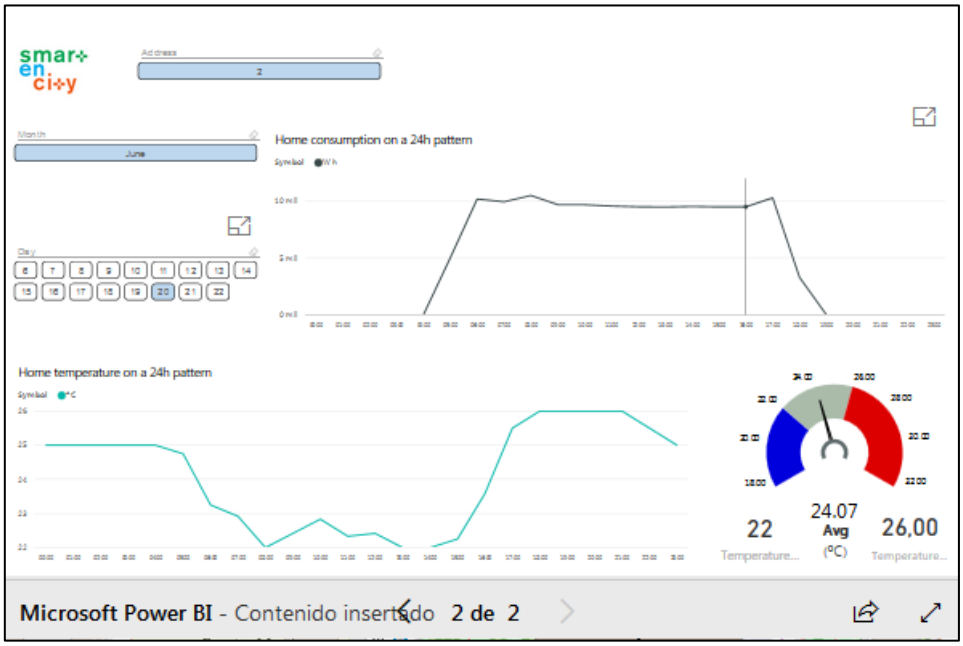


Figure 21: Report Window

Besides, there are some icons displayed in the window with additional functions:

- Access the full-screen mode.
- Display in a larger size each part of the report.
- Share the URL.

5.2.4 RA Demonstrator Functionality map

This section indicates which technologies, tools and mechanisms are used to build the demonstrator. The matrix in Table 4 includes summary of the modules/layers from the RA and indicating the technologies/tools/mechanisms used.

Layer	Module	Platform Technology, tool or mechanism
Acquisition/Interconnection layer	Protocol Adapters	Non applicable in this demonstrator.
	Development kit	Non applicable in this demonstrator.
	Protocol Abstraction Semantic	Non applicable in this demonstrator.
	Notifications	Non applicable in this demonstrator.
	Security	Non applicable in this demonstrator.
	Plug-in New Adapters	Non applicable in this demonstrator.
Knowledge layer	**Historic Repo**	**Hadoop File System (HDFS)**
	City Semantic	No implemented in demonstrator
	Real Time Repository	**PostgreSQL complemented with the TimeScale TSDB extension.**
	GIS Repository	**SQL Server deployed in Azure**
	Vertical Repository	**SQL Server deployed in Azure**
	KPI Repository	**SQL Server deployed in Azure**
	Structural Data Repository	**SQL Server deployed in Azure**
	Real Time Processing	No implemented in demonstrator
	Batch Processing	Aggregation of data using Azure tools
	Analytics	Reporting with Power BI in Azure
Interoperability layer	Open Data	Non applicable in this demonstrator.
	Development Kit	Non applicable in this demonstrator.
	APIs	Non applicable in this demonstrator.
	Security	Non applicable in this demonstrator.
Intelligent Services layer		Non applicable in this demonstrator.
Support layer	Audit	Non applicable in this demonstrator.
	Monitoring	Non applicable in this demonstrator.
	Logging	Non applicable in this demonstrator.
	Schedule	Non applicable in this demonstrator.
	Platform Management	Non applicable in this demonstrator.
	Repo Config	**Azure provides support layer modules as basis services to manage the platform using a configuration repository.**
	Connectors	Non applicable in this demonstrator.

Table 4: RA functionality – Platform functionality matching

6 Conclusions, deviations and outputs for other WPs

There are two main results produced in Task 6.3. The first result is a demonstrator available online. The demonstrator or prototype is a platform where the data models necessary for SmartEnCity are implemented. The deployed platform agrees with the Reference Architecture described in Task 6.2 (SmartEnCityD6.2, 2017). The demonstrator offers the data models necessary to build the CIOP in the lighthouses. The data models are used by means of an example (specific data flow). Section 5.2.3 includes access to the demonstrator´s user guide (online).

The prototype is a specific instantiation of the Reference Architecture. A demonstrator or prototype consists of a technological solution that fulfils the requirements of a Reference Architecture and provides the modules and functionality specific for the domain it represents. Several demonstrators built with different technologies and frameworks can agree with a common Reference Architecture and consequently be valid instantiations or implementations of that architecture. The development and deployment of the reference architecture could vary for the different demonstrators. Thus other technologies and tools might be used to build and manage the data models.

Data models are the real representation of a domain. Data models necessary to represent the Smart City domain have been constructed with the requirements available at this stage. Demonstrator work packages (WP 3, 4 and 5) could use these data models to implement the solutions to be developed in their lighthouses. Different implementations for the data models are also valid. They have to agree with the Reference Architecture definitions given in Section 4. Most common data models such as KPIs and infrastructure (structural and configuration repositories) might not change but new verticals and specific implementations of those verticals might differ from the ones propose here. Verticals in most cases are application dependent. In summary, the platform complements the CIOP framework presented in Task 6.2 with the data models.

The second result is outlined in this document. The description of the CIOP Reference Architecture data models is included in section 4. The document includes a general description of data model identified for the architecture including the type of data collected, the relation to other data models and the possible technologies to be used.

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 mainly on other activities of the WP6 and on activities related with the deployment of the CIOP platform in the three lighthouse cities (Vitoria-Gasteiz WP3, Tartu WP4 and Sonderborg WP5).

7 References

AENOR CTN-178. (2015). UNE 17804:2015. *https://www.aenor.es/aenor/normas/ctn/fichactn.asp?codigonorm=AEN/CTN%20178#.WG58xBvhC71*.

Ralph Kimball, L. R. (1998). The Data Warehouse Lifecycle Toolkit : Expert Methods for Designing, Developing, and Deploying Data Warehouses. In L. R. Ralph Kimball, *The Data Warehouse Lifecycle Toolkit : Expert Methods for Designing, Developing, and Deploying Data Warehouses.*

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

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

SmartEnCityD7.9. (2017). *SmartEnCityD7.9 "Data Collection Approach".*

Wikipedia. (2016, December). *https://en.wikipedia.org/wiki/Reference_architecture*.