# FinEst Twins

# Specifications and Plan for the Urban Open Platform

**FINAL version 1.0, 29.5.2020**

**DISSEMINATION LEVEL: PUBLIC**

This project has received funding from the European Commission (H2020 grant No. 856602) and the Estonian Government (Estonian Ministry for Education and Research).

REPUBLIC OF ESTONIA
MINISTRY OF EDUCATION AND RESEARCH

EUROPEAN COMMISSION
RESEARCH EXECUTIVE AGENCY
B5 – Spreading excellence and Widening participation & Science with and in society

This page is intentionally left blank.

# EXECUTIVE ABSTRACT

The concept of Urban Open Platform originates to the European Innovation Partnership on Smart Cities and communities (EIP SCC). The EIP SCC has defined six Action Clusters as their key priority areas, including "Integrated Infrastructures and Open Data". A general observation is that open urban platforms are a prerequisite to support fast take-up of smart solutions in cities to allow many stakeholders of a city to participate[1]. It is also expected that the urban platforms would have a key role in the integration of third-party vendor solutions.

The urban platform initiative was supported by launching a *Memorandum of Understanding*[1] that had 85 signatories, including both cities and smart city related vendors. In 2016 it was estimated that the Urban Platforms were fragmented, had uncertainties on the demand side and were lacking interoperability and common standards from the supplier side. At the moment, there is still work to do on all these areas.

The Urban Platform concept has evolved through several Horizon 2020 and ERDF funded projects. Forum Virium Helsinki has worked on the concept in projects such as bIoTope (688203), mySMARTLife (731297), Select4Cities (688196) and SynchroniCity (732240). In many of the projects, the usage of the Urban Platform has focused on being an IoT platform with dashboards. The support of spatial data and large-scale data utilization have been somewhat limited and many of the pilots were experimental platform products not ready nor intended to be in full-scale production use. The limited vision of the scope has also missed the opportunity in supporting the cities in some of their new data-related responsibilities such as maintaining INSPIRE -compliant data services and providing support for SECAP -reporting. The Urban Platform should also not be seen as a monolithic, single service but as a distributed and decoupled collection of services that can complement the already existing data platforms that the cities have. In both Helsinki and Tallinn, a key question is how can the urban platform build on top of the Microsoft Azure data lakes and data warehouses that both the cities are already invested in.

In the FINEST Twins -project, the role of the Urban Open Platform is to both support the research streams, piloting programs and the actual use cases coming from the cities of Helsinki and Tallinn. This document explains the key elements of the design and implementation of the platform, including also description of the recommended operational model with the governance processes explained.

[1] https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=9673

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1. INTRODUCTION

## 1.1. TARGET AUDIENCE

This document is intended for both technical and non-technical readers. It intends to provide a reasonable amount of background and context information in order to better support understanding the work, scope and focus of the Urban open Platform work package.

## 1.2. PURPOSE

While the primary purpose of the document is to act as an official deliverable of the FINEST Twins Work Package 4, this work also aims to provide an overview and guidance to all the stakeholders involved with the smart city projects. It also presents some findings of several Horizon 2020 projects that implemented the Urban Platform concept in a way suitable for the scope of each project.

It should be noted that the technical design and architecture presented in this work is not intended to be generic nor replacement of the earlier reference architectures, such as the Espresso reference architecture. The FINEST Twins project is unique in a way that it strongly builds on top of the existing data platforms and capabilities the cities of Helsinki and Tallinn have. Also, in the Espresso project then intention was that the reference architecture and design principles would be the basis for the cities to choose their own focus areas with related logical architectures and after that to define the specific solution architecture.

# 2. BACKGROUND

## 2.1. About Urban Platform

The Urban Platform vision of the EIP SCC was that by 2025, 300 million EU citizens were served by platforms within their cities and in short term accelerate the adoption of Urban Platforms through an easy-to-implement template approach, and cross-sector collaboration[2]. Meanwhile the cloud platforms have picked up and also the public sector has started to move their services into AWS, Google or Microsoft Azure cloud services.

The Urban Open Platforms are expected to form a core building block by which cities better manage the current explosion in volumes of city data and more easily share this data between city services in order to provide meaningful services to their residents.

It should be noted though that the Urban Open Platform is not a single IT system or server. It is a collection of smart city services that communicate internally and externally with harmonized APIs. The platform should be distributed and decoupled because of various reasons. The city organizational complexity does not support the idea of a single-owner, single-admin platform but the operational models should be based on governance model that takes this into account.

## 2.2. The API Manifesto

The modern IT architectures such as Service Oriented Architecture (SOA) and microservices have emphasized the need of APIs. The new paradigm of business and operational agility is needed to execute at the speed of digital economy. With Open APIs, the public sector and developer communities aim to unlock several growth and efficiency opportunities. As an example, Open APIs are used to expose the available capabilities to partners and developers.

One of the forerunners in both cloud platforms and utilizing the internal APIs has been Amazon.com and the Amazon Web Services (AWS) unit. The company had a strong, internal commitment to API - driven data flows and the internal e-mail sent almost 20 years ago is still a valid guideline for API developers and architects[3]:

- All teams will henceforth expose their data and functionality through service interfaces.
- Teams must communicate with each other through these interfaces.
- There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
- It doesn't matter what technology they use.

[2] https://eu-smartcities.eu/content/urban-platforms

[3] https://apievangelist.com/2012/01/12/the-secret-to-amazons-success-internal-apis/

- All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

In the smart city context, one of the early adopters of API architectures was the Horizon 2020 CitySDK -project (297220). The project intended to create an ecosystem in which the work of a developer is facilitated by having unified and open city interfaces available across different cities in Europe. The project also saw that an essential part of the ecosystem would be to create an environment in which innovation among developers or ICT service companies were stimulated.

The motivation for the CitySDK project was the growing number of apps available that connected to systems in European cities. While many cities had APIs, it was challenging to use the same app in more than one city. It was also noted that the documentation on these interfaces were often limited or lacking altogether. As the outcome of the process, the city of Helsinki started a process to define a governance model for the public API's.

## 2.3. European Data Strategy

The European Strategy for Data[4], published in February 2020, relies in many ways on Urban Platforms. The European strategy for data aims at creating a single market for data that will ensure Europe's global competitiveness and data sovereignty. Data is seen to be essential resource for economic growth, competitiveness, innovation, job creation and societal progress in general.

The main issues identified that would hold back the EU from realizing the full potential in the data economy were:

- Fragmentation between member states
- There is not enough data available for innovative re-use
- Data sharing between companies has not taken off at sufficient scale
- Data interoperability and quality remains an issue
- Individuals need empowering to exercise their rights
- Skills and data literacy are already listed as critical skills and there is a major shortage of data experts

According to the strategy, the European Commission aims to fund data infrastructures, data-sharing tools, architectures and governance mechanisms to support the digital transformation. As a specific initiative, the Commission intends to fund the establishment of EU-wide common, interoperable data spaces in strategic sectors. It should be noted that the EU-wide data spaces are populated with aggregated data so each level of data spaces ((personal) - city – national – EU-wide), would have a different level of granularity.

## 2.4. Open Data

A key function of the Urban Open Platform is the sharing of data. The cities have been forerunners in the open data movement. The city of Helsinki started a systematic move towards opening data ten years ago when the internal discussions of the role of the open data materialized. In 2009 a data

[4] https://ec.europa.eu/digital-single-market/en/policies/building-european-data-economy

vision for year 2020 was created by the cities of the Helsinki Region. One of the proposals was to set up a jointly managed open data service, Helsinki Region Infoshare, HRI.[5] The service opened in March 2011. The main goals set for the data vision and the HRI service was to harmonize and standardize information to improve data interoperability. Access to the data was to be granted with licensing terms that were user friendly and also support data business. To improve the quality of data, the open data service would rely on original sources of information. The open data service was not only a service to external parties but also to internal stakeholders within the city organization: it was expected, that the service would help to remove the need of requesting, copying and transferring the data between parties.

In 2011 the European Commission released a communication of Open Data. Open data (or public data) was seen as one of the resources of the EU2020 strategy to put Europe's economies into a high and sustainable growth path. The communication estimated that the overall economic gains from opening up the public data could amount to €40 billion a year in the EU[6]. The communication also emphasized the role of research and development of the data handling technologies and a significant funding was made available to information management on Horizon 2020 program for the period of 2014 – 2020.

From the city perspective, the aim of the open data is to create a citywide culture of data being *open as a method*. This means a policy where any data that the city creates is open by default, unless there is a specific reason to not to do so (e.g. privacy reasons). Open data refers to information that has been made available for free for anyone to use.

The level of openness can be defined by several aspects. The data is expected to be technically available, free to access, have license that permits reusing the data, be findable and be understandable. In the process of opening the data, one shouldn't consider too much of the applications of the data. Seldom any actual data services would rely on a single dataset but rather generate value by combining several datasets together, sometimes in an unusual combination. Also, the recent developments of the AI have raised the needs of data that is both extensive but also supported by other related datasets to identify correlations and causations in analytical models.

It should be noted that the use of data can also be restricted unintentionally. Dataset has been published in good faith but the data in it is not understandable without additional information about the context. At the same time, it would be preferred that the datasets are self-explanatory and do not require external metadata catalogs.

In order to categorize the open data by its availability and usefulness, Tim Berners-Lee suggested a 5-star deployment scheme for open data. The following diagram illustrates the method:

[5] https://www.hri.fi

[6] https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2011:0882:FIN:EN:PDF

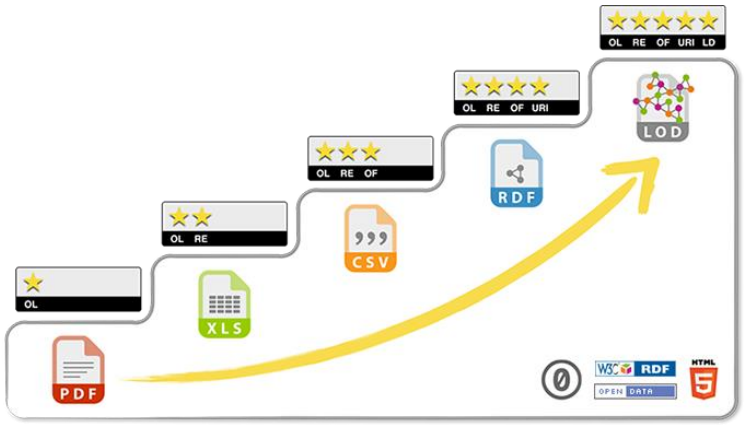


FIGURE 1: 5-STAR ASSESSMENT

The first star is achieved when the data is made available online under an open license. The next star requires that the data is in structured format, as an example as an Excel file. Three stars will require all the above but also that the data file format is non-proprietary, e.g. comma separated file (CSV). Four stars introduces the elements of linked data, using URI's as external references. Finally, all the five stars are given to dataset that is linked with other data to provide deeper context.

# 3. Data Models

## 3.1. Semantic Web

The semantic web is an extension of the World Wide Web which provides a common scalable framework for sharing and reusing data across the Web[7]. It is W3C's vision of the Web of linked data. The objective of the semantic web is to make data in the world wide web machine-readable and enable development of systems that are able to support trusted interactions over the network. The FINEST Twins Urban Open Platform is such a system. The most important semantic web technologies are Resource Description Framework (RDF), SPARQL, Web Ontology Language (OWL) and Simple Knowledge Organization System (SKOS) which provide the foundation for publishing and linking data[8]. For the web RDF is providing a common data abstraction and syntax, RDF schema and OWL together offer a common data modeling language for data in the web and SPARQL Query language and Protocol add a standard for interacting with the data.

The Semantic Web Stack illustrates the architecture of the Semantic Web and it is presented in the following figure[2]. The red line in the figure describes where the semantic part of the Web begins. The most important semantic web technologies are Resource Description Framework (RDF), SPARQL, Web Ontology Language (OWL) and Simple Knowledge Organization System (SKOS) which provides the foundation for publishing and linking data. For the web the RDF is a framework providing a common data abstraction and syntax, RDF schema and OWL together offer a common data modeling language for data in the web. RDF and SKOS are represented in more detail later in this chapter. SPARQL adds a standard for interaction with the data and it is the query language for Semantic Web. Requirements Interchange Format (RIF) is a rule interchange format that is used for describing relations that cannot be directly described with OWL description logic. In Semantic Web Rule Language (SWRL) all rules are expressed in terms of OWL concept and it is intended to be the rule language of the Semantic Web.

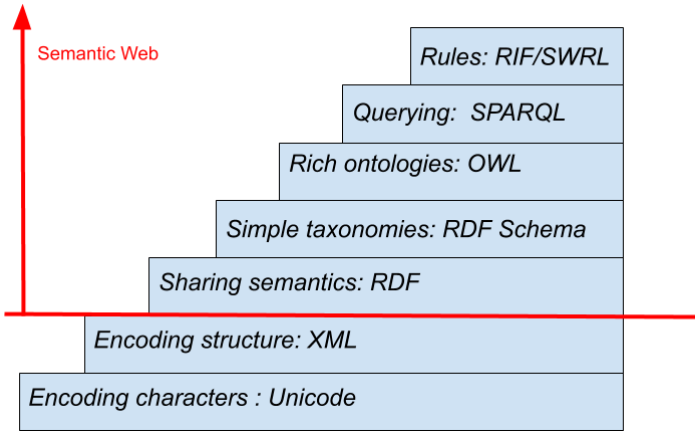


FIGURE 2: THE SEMANTIC WEB LAYERS

[7] https://www.itpro.co.uk/strategy/29606/what-is-the-semantic-web

[8] https://www.w3.org/standards/semanticweb/

## 3.2. Single Source of Truth

Single source of truth is practice where all organizations' componentized data and data related elements exist in only one place and they are also centrally managed. When an element is needed in the system it is used as reference or link to a source instead of duplication. Advantages of following single source of truth practices include[9]:

- As duplicate data entries are removed the time spent verifying which version of the data is correct is greatly reduced and thus workflow is increased
- Maintaining data becomes easier and more efficient
- Decisions can always be made with the correct data and it can help an organization to become data-driven

Single source systems are systems that utilize advantages of a single source of truth practice.

Single source of truth principles be achieved by a process called normalization which contains 3 parts that should be implemented in following order:

- componentizing content,
- removing all redundancy,
- reusing the components wherever they are needed

FINEST Twins platform is designed to be a single source system from the start. This is achieved by using linked data, data warehousing solutions for data storage and stream processing with centralized schema management.

## 3.3. Simple Knowledge Organization System (SKOS)

Simple Knowledge Organization System (SKOS) is a common data model for sharing and linking knowledge organization systems via the Web[10]. SKOS is used in systems such as thesauri, classification schemes, subject heading systems and taxonomies. With SKOS a knowledge organization system can be expressed as machine-readable data which can be exchanged between applications and published in machine-readable format on the web. SKOS belongs to the semantic web concept of standards and it is built on Resource Description Framework (RDF) and RDF schema.

RDF is a framework specified by the World Wide Web Consortium (W3C) for representing information about resources in the Web[11]. A resource can be anything from part of a Web Page to an entire Web site or it can also be a physical object that is not directly accessible via Web or it can be an abstract concept. A resource is identified using an IRI (International Resource Identifier), which is a generalization of URI (Uniform Resource Identifier). RDF enables applications to combine, publish and distribute structured and semi-structured data.

[9] https://www.talend.com/resources/single-source-truth/

[10] https://www.w3.org/TR/skos-reference

[11] https://www.w3.org/TR/rdf11-primer/

Other advantages of using RDF:

- Enables utilization of datasets published as Linked Data.
- API feeds can be linked together so clients are conveniently able to access additional information.
- RDF supports schema evolution over time without the need to update every data user and it has features that make it easier to combine data which has different underlying schemas.
- Enables machine-readable information that can be automatically processed by applications to be added to Web pages.
- Data sets can be enhanced by adding links to third-party dataset that provide access to related resources and information

The abstract data model of RDF is based on the concept of making statements about relationship between two resources consistently in the following structure[12]:

```
<subject> <predicate> <object>
```

These statements are called triples as they contain three elements. The subject and the object are the two connected resources and the predicate describes the nature of the relationship. The relationship is expressed directionally from subject to object and it is named in RDF a property. RDF triples can be also modeled as a graph where subject and object are represented by nodes and predicates from the arcs. SKOS expresses data as RDF triples and the data encoding can be done with any concrete RDF syntax.

The SKOS is founded on a concept-centric view of the. Data model of SKOS regards a knowledge organization system as a set of consistent and structured concepts which can be aggregated into concept schemes. URIs act as identifiers for both SKOS concept schemes and SKOS concepts making them referenceable from any context and part of the Web. For identifying a concept within the extent of a given concept scheme the concept can be assigned one or more lexical codes called notions. A notion is a unique string of characters not recognizable as a word in any natural language. Notions also offer a way to connect concepts with other systems used for classification e.g. classification codes used in library catalogs.

A SKOS lexical label is a string of Unicode characters associated with a concept in a given natural language[13]. SKOS concepts can have preferred, alternative and hidden labels. All labels can include optional language tags. The preferred and alternative labels offer the most accurate indication as to the intention of a SKOS concept and are most useful when describing the concept in human-readable form. Alternative labels are used for representing synonyms for the concept. Hidden labels are the labels not meant for human eyes and they include strings that are useful to associate with the concept. The hidden labels are beneficial when a user is interacting with a text-based search function and is unable to provide an exact search string, desired result can be still returned if the string matches one of the hidden labels. For providing information relating to concepts SKOS offers several

[12] https://www.w3.org/TR/rdf11-primer/

[13] https://www.w3.org/TR/skos-reference/

different documents (or note) properties. There is no limitation on what this information can include, e.g., it could be plain text, definition, an image, information about the scope or reference to another resource.

The SKOS data model supports hierarchical and associative links between SKOS concepts, enabling linking of SKOS concepts to other concepts via semantic relation properties and arranging them into hierarchies and association networks. Concepts can be organized into labeled and/or ordered collections, beneficial when a set of concepts have a common factor and it is convenient to unite them under a common label or it is relevant to order the concepts. The SKOS data model is supporting four types of mapping links which are used for mapping concepts to other SKOS concepts in separate concept schemes: associative, close equivalent, exact equivalent and hierarchical. SKOS concept scheme hierarchy and relations are illustrated in the following figure. In this example, taken from Finnish thesaurus and ontology service (FINTO) hosted by The National Library of Finland, the concept is 'open data' and it is a narrower concept of 'data', broader concept of 'linked open data', related to concept 'data policy', and equivalent of 'avoin data' and 'öppen data' concepts. *Skosmos* is an open source dictionary and ontology web-based tool operating behind the Finnish National Ontology, FINTO[14]. Skosmos is openly developed by The National Library of Finland and it's used also by organizations like UNESCO and FAO. Skosmos supplies services for accessing controlled vocabularies used by indexers for describing documents and searchers looking for appropriate keywords[15]. These vocabularies follow SKOS standard and are accessed via SPARQL endpoints. Besides providing a state-of-the-art user interface for people, Skosmos also supplies a REST-style API and Linked Data access to underlying vocabulary data. For this purpose, Skosmos server is used as part of FINEST Twins Urban Open Platform.

[14] https://www.kansalliskirjasto.fi/fi/palvelut/jarjestelmaalustapalvelut/skosmos

[15] https://github.com/NatLibFi/Skosmos

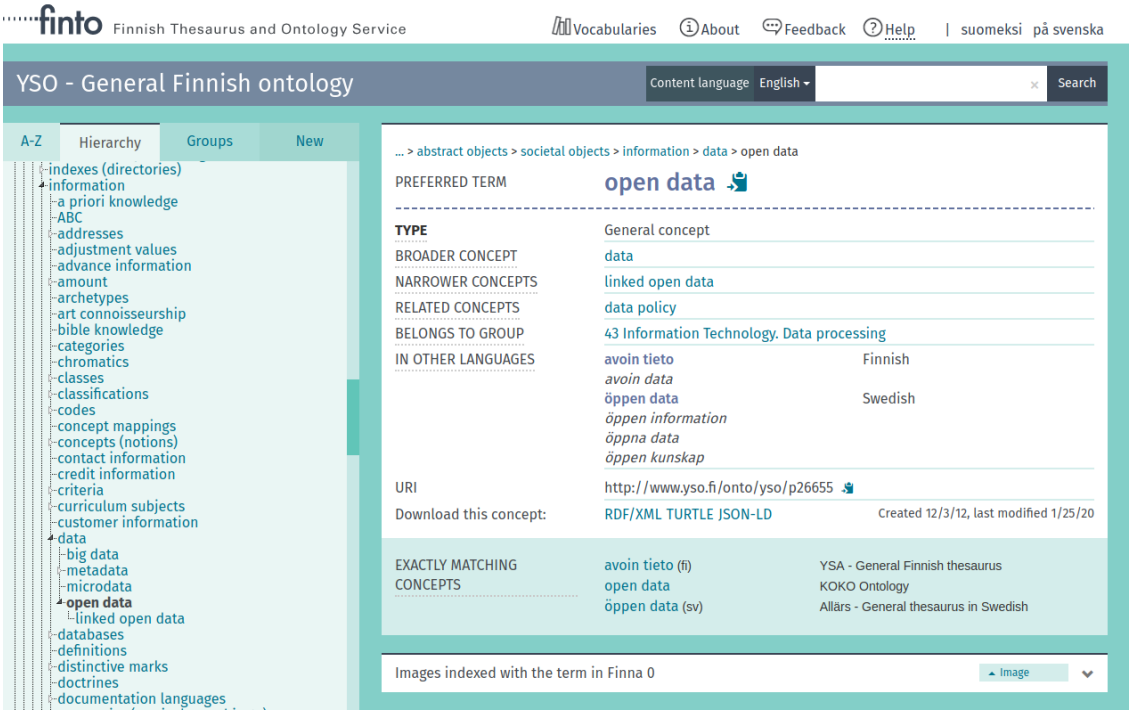


FIGURE 3: CONCEPT VIEW IN SKOSMOS SERVICE

### 3.3.1. Schema.org

*Schema.org* is a collaborative community activity, started in 2011, to improve the web by creating, maintaining and promoting schemas for structured data in the Web[16]. SKOS Concept Schemes are intended for capturing an individual complete vocabulary and are similar to the Schema.org approach. SKOS collections on the other hand are used for describing groupings of closely related concepts within vocabulary. SKOS collections make SKOS a better option when usage of ontologies is required, while Schema.org is better suited for non-hierarchical taxonomies. In SKOS usage of triples, specially the modelling of relationship between concepts, makes it easier to achieve the single source of truth principle. For these reasons SKOS based approach is used in the FINEST Twins Urban Open Platform as a single source of vocabularies instead of Schema.org

### 3.3.2. UCUM as Ontology

The Unified Code for Units of Measure (UCUM) has its foundation on the ISO 80000: 2009 Quantities and Units and standard series which defines the usage of System International (SI) units in publications[17]. It is a code system designed to consist of all units of measures being contemporarily used in international science, engineering and business. The objective of UCUM is to assist the progress of unambiguous electronic communication of quantities together with their units. It centers around machine to machine communication and the typical application of UCUM are electronic data interchange (EDI) protocols.

[16] https://schema.org

[17] http://unitsofmeasure.org/trac

A SKOS version of the original UCUM, presenting UCUM as an ontology, is hosted by The National Library of Finland's FINTO service[18]. The FINEST Twins Urban Open Platform is based on the SensorThings API data model, which will be discussed in more detail in the next part of this chapter. In this data model every sensor has one or more Datastreams. The Datastream has a property called unitOfMeasurement, which represents the unit of measurement used in observations made by the sensor. These units of measurements are compatible with The Unified Code for Units of Measure. With the help of Skosmos FINEST Twins Urban Open Platform is able to access the FINTO UCUM ontology and link concepts as JSON-LD to unit of measurement property. This knowledge can be utilized for producing automated reasonings from the results of observations made by sensors.

## 3.4. Managed Vocabularies

The role of the managed vocabularies is seen mostly in the attempt of improving semantic interoperability in a system that does not have a pre-defined domain: The Urban Open Platform is not for energy data or mobility data; it is for any city related data that can be streams or datasets. This goal will naturally mean that the data models will also need to rely on managed vocabularies that are generic. This is a new approach, since while the semantic web adoption is only in the early stages data platforms, most of the pilots and implementations utilize a domain specific vocabulary such as the *Project Haystack*[19] in building automation and energy management projects.

At the first stage of the project, data models are supported with two mature and well-governed vocabularies that are organized as hierarchical ontologies: The ISO 80000 and CITYKeys. More ontologies can naturally be added, since the project will host an own instance of the SKOSMOS server as a part of the Urban Open Platform.

### 3.4.1. ISO 80000

The ISO 80000 standard collection defines the International System of Units (SI -System). It consists of 13 different parts, featuring 11 from ISO and two from the International Electrotechnical Commission (IEC).

The series gives terms, definitions, recommended symbols, units and any other important information related to quantities used in science, engineering, metrology and industry. It is commonly used as a reference when writing scientific or technical documents, textbooks, standards and guides.

The ISO 80000 is maintained under the ISO Technical Committee 12 that has the title of Quantities and units. The governance process of ISO standards and ISO/TC12 especially is through and detailed. The organization started in 2017 a complete revision of all the standards under this group. A good example of the central role of the ISO was that when as part of the revision project a definition of kilogram was redefined, it became international news[20].

[18] http://finto.fi/ucum/fi/?clang=en

[19] https://www.project-haystack.org/

[20] https://www.nbcnews.com/mach/science/definition-kilogram-just-changed-here-s-what-means-ncna1007731

The ISO standards have a copyright license that restrict the publication of their contents in any other media than the standard documents purchased from the ISO store. As part of another project, Forum Virium Helsinki requested the ISO Secretariat a permission to use the contents of the ISO 80000 standards as an ontology, but this was not granted in a meaningful way. The license could have been only granted for a service where all the users downloading unit definitions from the SKOSMOS server would have purchased a license separately. In the way how semantic web works, this is naturally impossible.

### 3.4.2. CITYKeys

Funded by the Horizon 2020 program, the *CITYKeys project*[21] developed and validated together with the participating cities key performance indicators (KPI) and data collection procedures for the common and transparent monitoring as well as the comparability of smart city solutions across European cities. The work was done together with five cities: Rotterdam, Tampere, Vienna, Zagreb and Zaragoza together with the EUROCITIES network. The project started in 2015 and was finalized in 2017. While the project has finished, the original partners and researchers of the project have been helpful in defining indicators for the other projects and the CITYKeys KPI's have been used as part of the monitoring phase on several Horizon 2020 Lighthouse -projects, such as mySMARTLife, Triangulum, Remourban and Smarter Together.

The Performance Measurement Framework introduced in the CITYKeys project is structured followingly:

- People
  - Health
  - Safety
  - Access to (other) services
  - Education
  - Diversity & social cohesion
  - Quality of housing and the built environment
- Planet
  - Energy & Mitigation
  - Materials, water and land
  - Climate resilience
  - Pollution & Waste
  - Ecosystem
- Prosperity
  - Employment
  - Equity
  - Green economy
  - Economic performance
  - Innovation
  - Attractiveness & competitiveness
- Governance
  - Organization

[21] http://www.citykeys-project.eu/

    - Community involvement
    - Multi-level governance
- Propagation
    - Scalability
    - Replicability

The framework is based on an extensive analysis from 43 existing indicator frameworks. The project added some indicators to fill the identified gaps. As a result, the framework contains in total 92 project indicators and 3 city indicators. The project indicators are used to assess smart city projects. They aim to quantify the impact the project has made. They can also be used to benchmark projects and programs against each other, thus supporting the cities in evaluating future project proposals as part of the Digital Twin concept.

All indicators have been described in detail with an indication of expected data sources. As part of the Work Package 4 Urban Open Platform work, a SKOS -ontology is to be created of the CITYKeys framework to support data management and smart city management support tools, e.g. BI dashboards.

## 3.5. SensorThings

SensorThings API is an Open Geospatial Consortium (OGC) standard specification offering a centralized and open framework for Internet of Things sensing devices and applications to connect with each other over the Internet[22]. SensorThings API is an open standard that deals with IoT systems ability using and understanding the exchanged data and it can effortlessly be integrated into existing Spatial Data Infrastructures and Geographic Information Systems. Because SensorThings API is designed especially for the IoT devices with limited resources and the Web developer community, the design follows REST principles, JSON encoding, and the OASIS Open Data Protocol and URL conventions. SensorThings API contains two parts, each offering a functionality. They are called the Sensing part and the Tasking part. The Sensing part functions are similar to OGC Sensor Observation Service and they offer standardized way for managing and retiring observations and meta from varied IoT sensor systems. It permits IoT devices and applications in a SensorThings service to CREATE, READ, UPDATE and DELETE IoT data and metadata. The Tasking part functions are similar to OGC Sensor Planning Service and they offer standardized way for parameterizing of task-able IoT devices such as sensors and actuators.

[22] https://www.ogc.org/standards/sensorthings

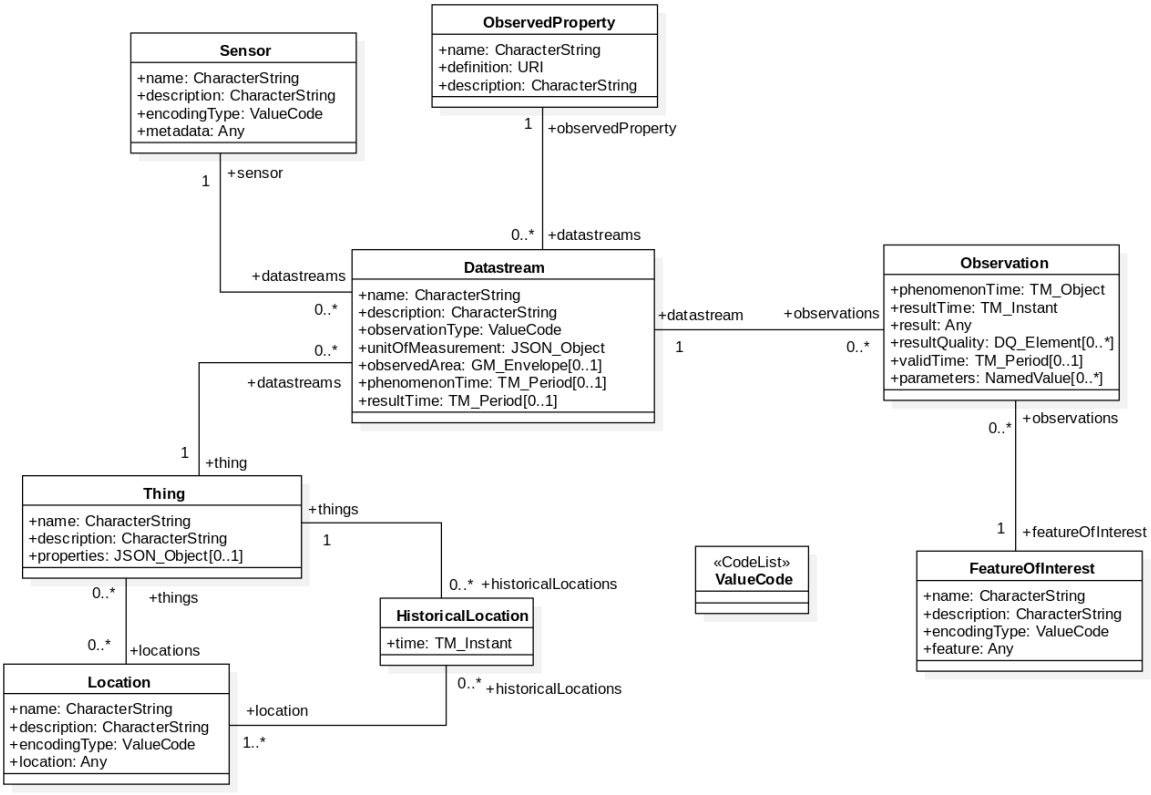


FIGURE 4: SENSORTHINGS API DATA MODEL

In the center of SensorThings API is its data model, based on the ISO 19156 observations and measurements standard specifying a conceptual model for observations and features involved in the observations. In SensorThings data model features are things (actual physical IoT devices or systems), sensors (instruments that observe a property or phenomenon) and features of interests (feature of interest of an observation e.g. it can be the geographical area or volume that is being observed by a sensor). In the model each Thing has an unspecified amount of Locations and Datastreams. Location geographically locates the thing or things it is associated with. A datastream is a collection of observations observed by the same sensor and measuring the same ObservedProperty. An ObservedProperty specifies the phenomenon of an Observation. Each Observation observes one specific FeatureOfInterest. FINEST Twins Urban Open Platform main data model, is based on SensorThings API data model. However, in the FINEST Twins Urban Open Platform version some of the elements are presented as linked data.

## 3.6. Spatial Data

Spatial data is information about a physical object that can be represented by numerical values in geographic coordinate system. The spatial data represents the location, size, shape and characteristics of objects. Geographic Information Systems (GIS) can be used to access, visualize, manipulate and analyze geospatial data.

The spatial attributes of data are highly important in public services like cities, municipalities or countries. The decisions of public bodies oftentimes target on certain areas or locations like a block or district. The public services are placed to cover geographical areas and the budgets, especially on recreational services, are allocated evenly throughout the city space.

### 3.6.1. ISO/TC 211

The International Organization of Standardization (ISO) organizes the major domains of its work under Technical Committees (TC). The ISO/TC 211 is a technical committee for geographic information and geomatics. It works in co-operation with Open Geospatial Consortium (OGC) that brings the user perspective to the standardization process.

The specified scope for the work of ISO/TC 211 is to *aim to establish a structured set of standards for information concerning objects or phenomena that are directly or indirectly associated with a location relative to the Earth*. *These standards may specify, for geographic information, methods, tools and services for data management (including definition and description), acquiring, processing, analyzing, accessing, presenting and transferring such data in digital / electronic form between different users, systems and locations.*[23]

### 3.6.2. INSPIRE

The Directive 2007/2/EC of the European Parliament and Council established mandatory requirement for the member states to establish an infrastructure of spatial information. The directive entered into force on the 15th May 2007.

The INSPIRE directive aims to create a European Union spatial data infrastructure for the purposes of EU environmental policies and activities which may have an impact on the environment. The European Spatial Data Infrastructure was expected to enable the sharing of environmental spatial information among public sector organizations facilitate public access to spatial information across Europe and assist in policy-making across boundaries.

The directive addresses 34 spatial data themes needed for environmental applications. Prior to the legislation process, extensive fact-finding and public consultations took place with the following key results:

1. Spatial data is often missing or incomplete
2. The description of available spatial data is often incomplete
3. Spatial datasets can often not be combined with other spatial datasets
4. The systems to find, access and use spatial data often function in isolation only and are not compatible with each other
5. Cultural, institutional, financial and legal barriers prevent or delay the sharing and re-use of existing spatial data.

While the origins of INSPIRE were in environmental data, the technical requirements were set to be built on top of the existing geographical information systems used in urban planning and cartography. The technical requirements of the directive widely adopted the ISO/TC 211 standard collection.

The requirements for the datasets are set in the Annexes of the INSPIRE directive. From the Urban Platform perspective, it is important to be aware that base map layers such as addresses, land cover and transport networks are INSPIRE requirements but so are environmental and meteorological monitoring facilities (e.g. air quality monitoring stations).

[23] https://iso.org/committee/54904.html

### 3.6.3. Simple Feature Access

Simple Feature Access is an Open Geospatial Consortium and International Organization for Standardization standard ISO 19125 that defines a common storage access model of mainly two-dimensional geometries (point, line, polygon multipolygon, etc.) used by geographic information systems[24][25] . FINEST Twins Urban Open Platform is applying this standard for managing geospatial data.

Simple Feature Access consists of two parts. Part one, Simple Feature Access - Part 1: Common Architecture (ISO 19125-1), describes the common architecture for simple feature geometry. The simple feature geometry object model is platform neutral and is using UML notation. The hierarchical object model for geometry is shown in Figure 3.2. Root class in the hierarchy is Geometry and it has subclasses for Point, Curve, Surface and GeometryCollection. There is an association between every geometric object and Spatial Reference System, describing the coordinate space where the geometric object is defined.

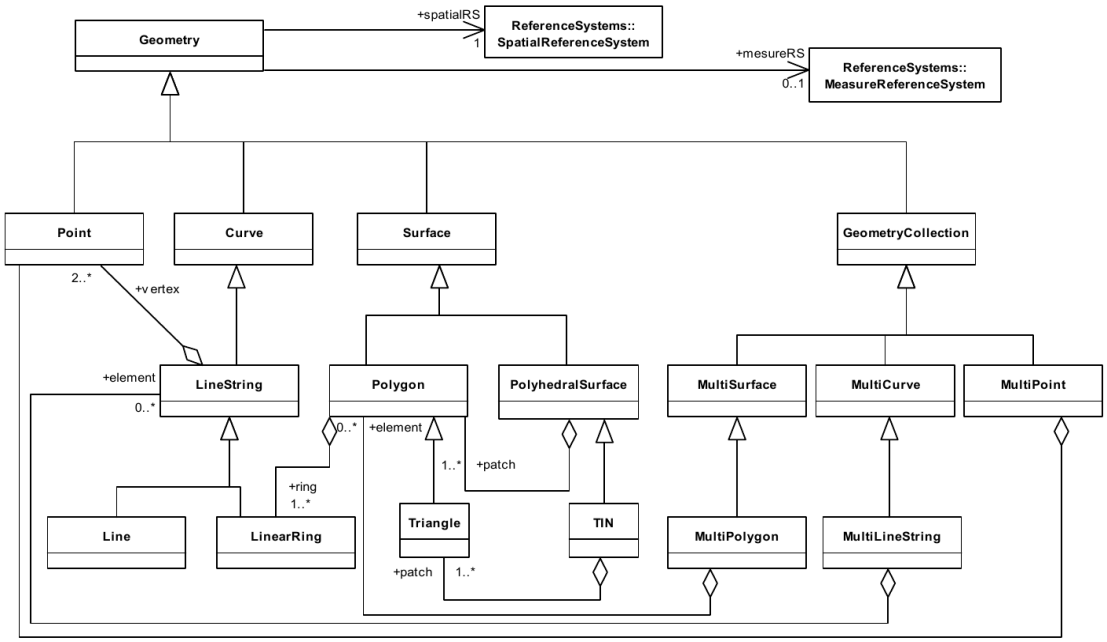


FIGURE 5: SIMPLE FEATURES

Part two of the standard is called, Features and Geometry – Part 2: Measure[26] (ISO 19125-2 describes an implementation of simple feature geometry using Structured Query Language (SQL). It provides an architecture for implementing the feature tables and defines an SQL schema for structuring, accessing and manipulating simple geometry feature data collections. Many of the database extensions created for managing geospatial data are adoption of this standard.

[24] https://www.ogc.org/standards/sfa

[25] https://www.iso.org/standard/40114.html

[26] https://www.iso.org/standard/40115.html

### 3.6.4. CityGML

CityGML is standardized open data model and XML-based format for storage and exchange of virtual 3D models of cities and landscapes[27]. It is implemented as GML application schema for Geography Markup Language version 3.1.1 (GML3), extendable international standard for spatial data exchange issued by the ISO TC211 and Open Geospatial Consortium (OGC). The goal motivating CityGML development is to have a common definition of the basic entities, attributes and relations of a 3D city model so their upkeep can be cost-effective and sustainable. This is achieved by supporting the reuse of the same data in different application fields.

Beside portraying the graphical appearances of city models the CityGML also addresses the representation of the semantic and thematic properties, taxonomies and aggregations. CityGML contains a geometry model and a thematic model. With the geometry model geometrical and topological properties of spatial objects can be specified within the 3D city model consistently and homogeneously. The thematic model applies the geometry model for various thematic fields like Digital Terrain Models, buildings, bridges, tunnels, vegetation, land use, water bodies, transportation facilities and city furniture. Besides spatial properties, features of CityGML can be assigned appearances which are not bound to visual data but offers arbitrary observable properties of the feature's surface such as infrared radiation, noise pollution or earthquake-induced structural stress.

CityGML functionality is divided into modules which consists of a core module and thematic extension modules[3]. The core module of CityGML specifies the basic concepts and components of the CityGML data model making it universally lower bound to it. All thematic extensions modules have dependency to the core module and it must be implemented by any conformant system. These extension modules contain logically separate thematic components of the CityGML data model which provide support for the definitions of different thematic models such as building, land use, city furniture, transportation, land use, vegetation and so on. Additional thematic extensions can be appended without the need to release a new version of the data model. This approach has been applied when the use of the model was extended towards solar energy potential, noise emission simulations or testing the effects of flooding.

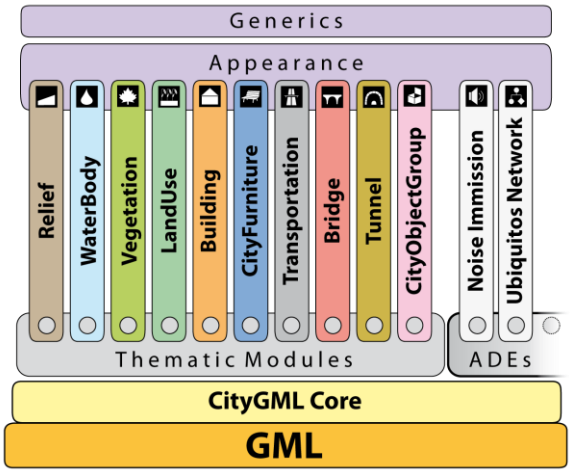


FIGURE 6: CITYGML MODULAR DATA MODEL

[27] https://www.ogc.org/standards/citygml

Additions to the CityGML data model are defined with Application Domain Extensions (ADE)[28]. These additions can be either additions of new properties to existing CityGML classes e.g. the number of persons inside of a building or the definition of new object types e.g. adding a new module for city walls and monuments. An ADE has to be specified in an extra XML schema definition file. The file needs to have its own namespace and import the XML schema definition of extended CityGML modules. This approach results in the extension being formally defined and it is also possible to have ADEs standardized by information communities. ADEs offer great flexibility to the CityGML data model as they can be specified for more than one CityGML module making ADE mechanism orthogonally aligned with the modularization approach of CityGML.

Despite its name, the CityGML data model can be implemented with three different encodings: GML, SQL and JSON. CityJSON is the newest version and was created to support mobile apps. CitySQL is not used as a name, but the 3DCityDB[29] project has created a relational database schema that supports all the thematic modules of CityGML 2.0 version with five Level of Detail (LoD), appearance data and WFS 2.0 interface.

## 3.7. CitySDK Interfaces

Goal of the CitySDK, service development kit for cities and developers offering a range of tools and information, is to have harmonized and reliable application programming interfaces (APIs) across cities that empower swift development, scaling and reusing of services and applications[30]. CitySDK project started in January 2012, is a collaboration between 8 cities around Europe aiming to build reusable interfaces and processes following best practices. The collaborative sustainable development of the open source toolkit has involved cities, innovation agencies, universities and developer communities. Committed partner cities have made open data available to be used by the different APIs and applications, also all CitySDK resources are freely available and online. Next in this document we present Open311, Open Decision and Linked Events of CitySDK APIs.

### 3.7.1. Open311 API

The Open311 is an issue reporting API which enables people to send a different kinds of issue reports directly to the city's feedback system from external devices[31]. The API offers an efficient way for gathering data as the issue reports can include images and locations. Besides reporting issues to the city, citizens are also able to track the progress of the issues which increases engagement by providing two-way communication with the civil servants.

The Open311 API is based on the international Open 311 standard[32] with a few extensions. FINEST Twins Urban Open Platform is able to determine based on observation data faults in the related IoT device and generate an Open311 service request. The Open311 API is used by the following cities:

[28] https://www.ogc.org/standards/citygml

[29] https://www.3dcitydb.org/3dcitydb/

[30] https://www.citysdk.eu/about-the-project/

[31] https://www.citysdk.eu/wp-content/uploads/2016/12/20171109_HarmonisedSmartCityAPIs_WEB.pdf

[32] http://www.open311.org/

Helsinki Finland, Espoo Finland, Oulu Finland, Turku Finland, Amsterdam Netherlands, Rome Italy, Barcelona Spain, Lisbon Portugal and Lamia Greece.

- Open311 API provides following functionalities:
- Submit issue reports via third-party apps and services
- Utilize a standardized reporting protocol with service request categories defined by each city
- Send media files and location details along with issue reports
- Search and follow up on submitted reports and their status

### 3.7.2. Open Decision API

Open Decision API is used for making local decisions and decision-making processes more transparent and accessible to the public[33]. Agendas and records of municipal administration and their attachments can be published as open data, which can be filtered by date, keywords and location. With the API it is also possible to combine information with data from other sources e.g. an application could combine data from several cities. The international Popolo standard[34] which is a model for relationships between organizations and people is used in the information model of Open Decision API. The Open Decision API is used by following cities in Finland: Helsinki, Espoo, Tampere, Vantaa, Oulu and Turku.

Open Decision API provides following functionalities:

- Make use of categorized data on city decisions
- Obtain decision data in a harmonized and machine-readable format
- Search for data by date, keywords and location

### 3.7.3. Linked Events API

Linked Events collects event data from different sources into a harmonized city-specific open API, which promotes accurate information as the data is not copied and pasted from the database but accessed directly through the API in a JSON-LD format[35]. Harmonized Linked Events API can be used to produce scalable intercity event management applications, which can help citizens to find events that interest them. The data model of linked events uses schema.org vocabulary. The Linked Events API is used by following cities in Finland: Helsinki, Espoo, Tampere and Turku.

Linked Events API provides following functionalities:

- Search for categorized event data in harmonized and machine-readable format
- Search for event data by date, keywords and location
- Search for event data in a specified area

[33] https://www.citysdk.eu/wp-content/uploads/2016/12/20171109_HarmonisedSmartCityAPIs_WEB.pdf

[34] www.popoloproject.com

[35] https://www.citysdk.eu/wp-content/uploads/2016/12/20171109_HarmonisedSmartCityAPIs_WEB.pdf

## 3.8. Schema Management in Urban Platform

The design of schema management in FINEST Twins Urban Open Platform is based around a centralized approach and following the single source of truth principle. Every component in the Urban Open Platform processing data is communicating via RESTful APIs with a single schema management system where all the schemas, including version history, used in the platform are stored. There are several benefits for having a schema for the data[36]. Schema specifies the structure, the type and the meaning of the data, and makes data encoding more efficient. The schema management system currently has simultaneous support for AVRO, JSON and Protobuf schemas. Before data objects can be transferred as a record within Urban Open Platform they must be converted into bytes of arrays. This process is known as serialization. When the record has reached its destination, a process called deserialization is used to convert the byte array back to structured data. A schema retrieved from the schema management system is required both for serialization and deserialization of the data. Sending a unique schema ID instead of a full schema with the record is enough. If a consumer of the record does not have the schema cached it can use the ID to retrieve the full schema from the schema management system. Not sending a full schema with each record saves time as it speeds up the serialization process. As was mentioned earlier in this chapter, in Urban Open Platform's data model many elements are presented as URI links instead of objects. This is beneficial for schema management and overall robustness of the system as it prevents object nesting which can cause problems with the Datastreams. Using linked data also greatly simplifies the schemas and therefore reduces the complexity of schema evolution.

When an application or a system is updated there may rise a need to evolve the initially defined schema. Schema evolution is a significant part of the data management and overlooking it in production can become costly. It is vital that all consumers of the content are able to process data encoded with every version of the schema. If versions of schema don't match but are compatible, then the record can still be transferred with aid of schema evolution. Schema management used in Urban Open Platform is able to validate if a schema is compatible with a particular version and there exist four compatibility levels: none, backward, forward and full. None level means that there is no schema compatibility validation (none compatibility should never be used in production), backward compatible schema is able to read data written with an older version of a schema, forward compatible schema is able to read data written with a newer version of schema and full compatibility means that the schema is both backward and forward compatible. Figure 3.5 describes a situation where compatibility level of a schema is 'full'. There are two separate content producers producing different versions of records and using different versions of a schema. Same datastream is used for transferring both versions of the serialized data to content consumers. Each consumer is able to deserialize both versions of the data with help of the schema management system as the schema is validated to be fully compatible.

[36] https://docs.confluent.io/current/schema-registry/index.html

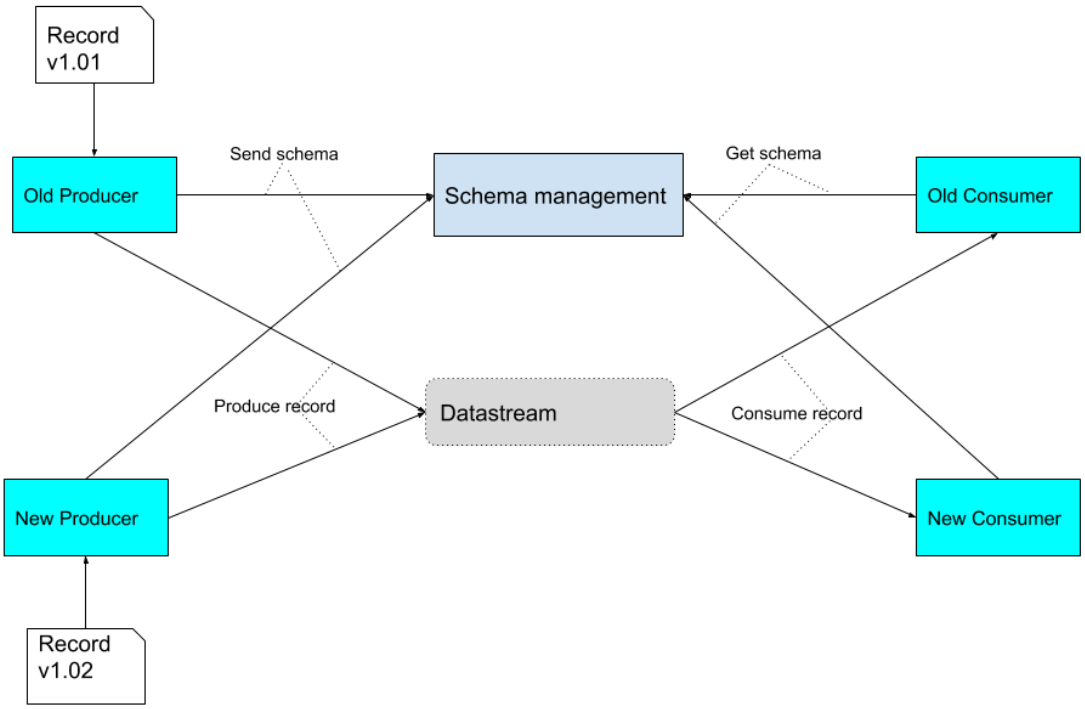


FIGURE 7: SCHEMA MANAGEMENT IN URBAN PLATFORM

# 4. Geographic Information Systems

## 4.1. About Spatial Data

Spatial data is any data with a direct or indirect reference to a specific location or geographical area. Spatial data is often referred to as geospatial data or geographic information. A direct reference usually means coordinates and an indirect reference means addresses, codes or names that refer to a place or region.

### 4.1.1. Spatial Data Flows in City of Helsinki

Municipalities have many responsibilities for the development and maintenance of the infrastructure for spatial data. The key tasks for the production and use of spatial data are real estate formation, urban planning, building control, traffic planning, parks and public areas, public transport, fire and rescue, environmental management, water supply, energy management and waste management. Spatial data is also increasingly used in decision-making and service network planning as well as in information services for residents.

In the cities, the production and maintenance of spatial data takes place mainly in land use planning, real estate, city survey services and environment services. Cities themselves take care of many of their spatial data sets and maintain them themselves and partly in cooperation with consultants. The city's spatial data system acts as a data warehouse containing many interconnected spatial data sets.

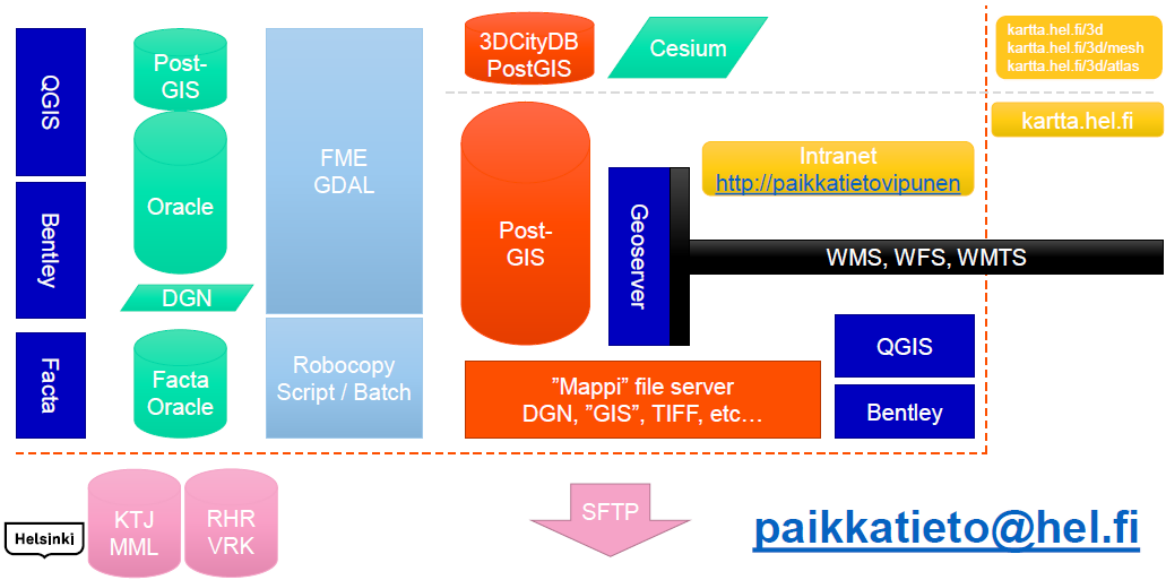


FIGURE 8: SPATIAL DATA FLOWS IN HELSINKI

The City Survey Services take care of several spatial datasets:

- Land use planning data from detailed planning processes
- Real estate formation and legal cadastral procedures
- Building inspection measurements
- Addresses

- Map of underground pipes and cables
- Aerial photographs
- Laser scanning data
- Registries

The map data maintenance in Helsinki relies on Bentley MicroStation software. The municipal registry system is called Facta and it contains information on the properties, buildings and plans of the City. The municipal registry is mainly maintained for official use. Facta has a connection to the National Cadastral Register (KTJ, maintained by the National Land Survey of Finland) and to the Building and Dwelling Register (RHR, Digital and Population Data Services).

The use of Oracle GIS database since 2009 has accelerated the transform from a file-based to database-driven data management. The use of Oracle GIS will increase in the future.

The data processing is an elemental function in spatial data management. The Mappi file service is mostly used as a workspace for Bentley MicroStation workflows. FME by Safe Software is a data conversion tool supporting well different kind of geospatial file formats and it is used especially with GeoServer to produce spatial data for its open WMS, WMTS and WFS interfaces. For internal use, the city has an online geographic information system called Paikkatietovipunen. The City of Helsinki open map service (https://kartta.hel.fi) is also widely used by external and internal users.

The processes are widely automated but the large volumes of data sometimes create bottlenecks. Also, the requirements for data with higher accuracy have been increased.

### 4.1.2. Detailed Planning and Strategic Urban Planning

Land use planning creates the preconditions for a good and functional living environment. General guidance for land use planning is based on Land Use and Building Act and regional and municipal planning are directed by national land use guidelines. City's land use planning instruments include e.g. urban plan (the local master plan) and detailed plan, land use strategies, land policy and building ordinance. Spatial data plays an important role in the development and measurement of a livable city.

Land use and city structure department (which includes e.g. strategic urban planning and detailed planning) in the Urban Environment Division …

Land use and city structure department (which includes e.g. strategic urban planning and detailed planning) in the Urban Environment Division has a spatial information system (geographic information system, GIS) which contain a wide range of interconnected spatial data set (geographic dataset):

- Buildings (internal WFS/WFS, SeutuCD by HSY)
- Detailed plans (approved plans are available for download in CAD format)
- Population data (SeutuCD, aluesarjat.fi, SYKE YKR)
- Companies and business locations data (SeutuCD)
- Realization of the plans (SeutuCD), forecasting of the plans (internal WFS)
- Conservation data

- Survey data
- Background maps: Cadastral map, Base map (City Survey Services)
- District Divisions, School Enrolment Regions (City Survey Services)
- Transportation networks
- Green and recreational areas (internal WFS)

City of Helsinki is digitizing and developing the information management of local detailed plans. The goal is that land use data is interoperable and readable regardless of the stage of the land use planning process. Land use data will be published through technically up-to-date open interfaces (as open data) and the data will flow throughout the entire value chain of land use planning, construction and maintenance.

### 4.1.3. Open Spatial Data

More and more spatial data sets are open. The regulations and guidance of the INSPIRE Directive (2007/2/EC) require public administrations to ensure that the spatial data should be readily and transparently available. Most of the City of Helsinki maps and geographical information databases have been opened to open use. Helsinki City Survey Services (in the City Urban Environment Division) publishes open data through open map interfaces (WMS / WMST / WFS). The technical platform of the service is GeoServer[37]. Open data and maps provided by City Survey Services can be found at the following address: https://kartta.hel.fi/avoindata/

## 4.2. 3D City Models

There are two next generation 3D city models of Helsinki available: a semantic city information model and a visually high-quality reality mesh model. The models are available as open data[38].

A reality mesh model is a photorealistic city model that creates a visual geometric model which matches the reality of a city based on aerial photographs. The reality mesh model can be utilized in various online services or as the basis for all kinds of design projects, such as planning the exit routes and the locations of performance stages and sales stalls for city events.

The CityGML model is a semantic city information model according to the Global OGC (Open Geospatial Consortium) standard (CityGML v2.0). The city information model allows users to perform a variety of analyses focusing e.g. on energy consumption, greenhouse gases or the environmental impacts of traffic.

Prior to the production of the Helsinki's 3D city models, the most important source data for both models (reality mesh model and city information model) came from aerial images and point cloud data sets as well as registries and spatial data. After the spatial data sets came the model creation process and the modelling of planned buildings. Maintenance and updating the CityGML city

[37] https://www.geoserver.org

[38] https://hri.fi/data/en_GB/dataset/helsingin-3d-kaupunkimalli

information model of Helsinki is carried out at Helsinki City Survey Services in the City Urban Environment Division.

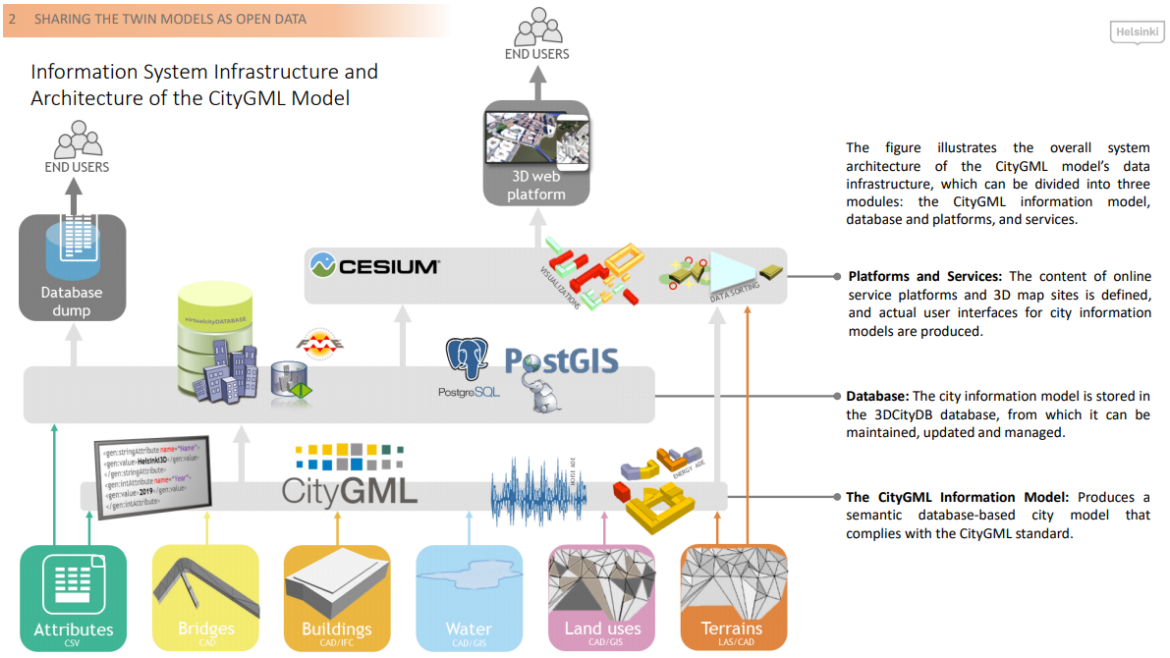


FIGURE 9: CITYGML DATA INFRASTRUCTURE

For more information about the advanced use cases and simulations created with the 3D City Model, the final report of the Kalasatama Digital Twins Project[39] will provide a detailed overview on the recent pilots.

## 4.3. Application Model Extensions

CityGML (the City Geography Markup Language) was published in 2008 by OGC. CityGML provides an extension mechanism called Application Domain Extension (ADE). When using this mechanism, extensions are determined by ADE rules. ADE is a built-in mechanism of CityGML to augment its data model with additional concepts required by particular use cases, and many ADEs have been developed for various fields. For example Technical University of Munich has developed CityGML Utility Network ADE for the modeling and 3D representation of supply and disposal infrastructures e.g. electricity, gas, district heating, telecommunication, fresh and waste water networks within one framework.

### 4.3.1. Energy ADE

The design of the Energy ADE has been driven by two mains objectives:[4]

[39]https://www.hel.fi/hel2/tietokeskus/data/helsinki/kaupunginkanslia/3D-malli/Helsinki3D_Kalasatama_Digital_Twins_020519.pdf

- Store and manage energy-relevant data collected at urban scale, based for instance on the buildings data specifications of INSPIRE Directive, or concept defined by the US Building Energy Data Exchange Specification (BEDES), such as building usage, year of construction, number of dwellings and residents etc.
- Provide data to assess the energy performance of buildings using a variety of different approaches and software tools, e.g. ranging from standard-based energy balance methods as of ISO 13790, to sub-hourly dynamic simulations by means of specific software programs.

The Energy ADE extends the standard Building object of the CityGML model with four new modules: Occupant Behavior, Material and Construction, Energy Systems and Building Physics. The model also links the energy related features with Weather Data, Energy Demand and other energy system attributes. With this approach the model can contain more detailed information of the building exterior shell and key thermal properties such as the U-values and g-values.

The city of Helsinki is currently in process of extending the data model of the 3D City Model with the Energy ADE extensions. This work supports the Energy Renaissance climate program of the city.

### 4.3.2. Urban Planning ADE

In Japan municipalities using OGC CityGML have developed the “i-Urban Revitalization” database to attract high quality investments for urban regeneration. Urban areas in Japan are facing continuous population decline and aging society and this poses major challenges for urban structure. Rapid increase of empty apartments and lands, real estate markets low transparency and the utilization of information have raised a recognition for the significance and importance of compact urban development from the perspective of administrative costs. Taking these into account the Japanese government strongly promotes formation of a high-quality urban revitalization project for regional hub cities.

The “i-UR” is an information infrastructure for urban revitalization. It allows the quantitative analysis and visualization of urban areas using geospatial information and virtual reality technologies. The i-UR Data is the combination of following data:

- 3D city objects and city model
- Detailed information of city objects (e.g. buildings, transportation, land use) for analysis
- Constraints/conditions (e.g. plans and regulation) related to urban revitalization
- Statistical grid data for global analysis and visualization

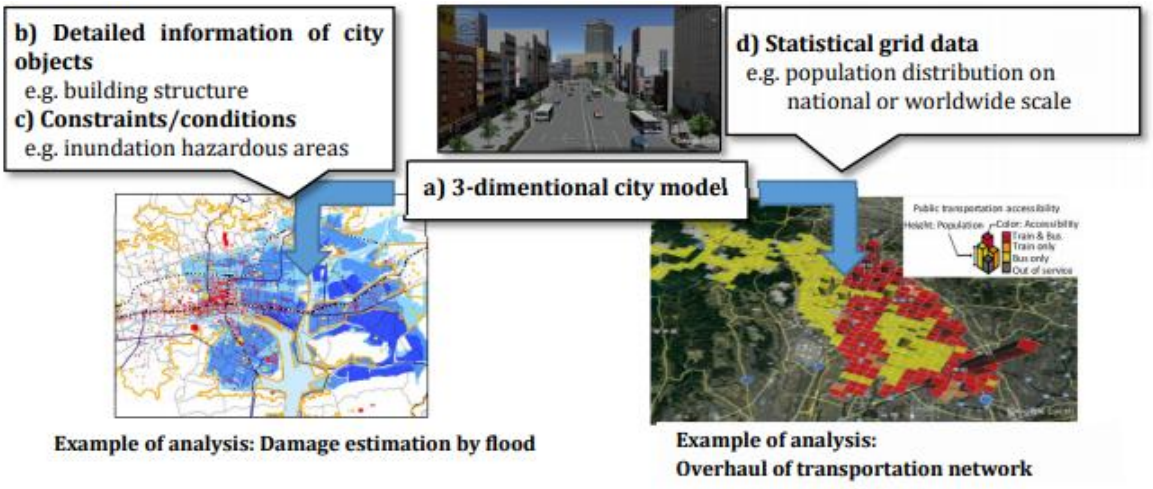


FIGURE 10: STRUCTURE OF I-UR DATA

The i-UR Data aims to be utilized in various application fields, such as disaster prevention, tourism and to carry out urban revitalization by attracting investment. Encoding specifications is meant to be applied not only in Japan but also in other countries, and therefore general-purpose data specification has been created[40].

## 4.4. PPGIS

Public Participation Geographic Information Systems (PPGIS) methods are used to map people's experiences related to certain locations. PPGIS methods bridges the gap between city planners and residents and turns regular people into experts. Online mapping technologies are easily used by non-experts, particularly for public engagement.

According to recent study[5], Map-based questionnaires are often used (31 %) and seen as useful (38 %) tools for urban planners. City of Helsinki has utilized participatory mapping methods especially in planning. Participatory planning highlights the importance of dialogue between planners and participants and the participants' role as knowledge producers. PPGIS methods are used to gain place-based information, reach a wider number of participants and build trust between participants and the planning organization.

Benefits for using map-based surveys:

- Public participation is nowadays integral part of urban and transport planning processes
- PPGIS data is location-based and well structured
- Efficient way to collect information from a large crowd within a short period of time
- Reaches those citizens who wouldn´t join the public meetings or workshops
- People can express their views and habits voluntarily
- Easy to use – people can participate whenever and wherever they want

[40] http://www.kantei.go.jp/jp/singi/tiiki/toshisaisei/itoshisaisei/iur/shiyouan1_3.pdf

- Can be used in projects of different scale (state-wide to specific neighborhoods)
- Collected data is easy to organize and analyze (and share as an open data)
- Dialogue and engaging in early stages reduce risks in planning projects

### 4.4.1. Case Maptionnaire & How Walkable is Helsinki? (2019)

Maptionnaire (by Mapita Oy) is one of the most popular PPGIS to have been applied in many planning contexts, mostly in Finland, but also internationally. Maptionnaire is a map-based survey tool which facilitates simple and effective public participation. The online questionnaire editor allows for the independent creation, customization and publishing of survey projects. The tool enables planners and researchers to collect, analyze and visualize map-based data, as well as citizens to co-design project areas and express their preferences and opinions. Maptionnaire has been used to engage thousands of urban residents, and also been customized to engage target groups in urban planning e.g. older residents and children.

Maptionnaire is widely used by urban planners and researchers and has been used in:

- Land-use planning and built environment
- Transportation planning and mobility
- Social sustainability
- Parks and recreation
- Real estate services
- Public services and the business sector
- Environmental services

The new version of Maptionnaire was released in spring 2020. Maptionnaire has now OpenLayers platform, which allows to add new features into Maptionnaire, including more types of interactive backgrounds, such as 3D models and visualizations and gamified planning or decision making e.g. participatory budgeting.

### 4.4.2. Case Walkability in Helsinki (2018-2019)

The city of Helsinki has set an ambitious aim to be the most functional city in the world. One core dimension of this vision is improving walkability in the city center. To start a dialogue with residents about the vision, Helsinki decided to open a map-based survey asking how people perceived the current state of walkability in the city center and where it could be developed.

The survey was co-created with Maptionnaire and Helsinki´s planning experts. It was possible to answer in three languages and via mobile devices. The survey included questions about e.g.

- the everyday movements of pedestrians,
- people’s wishes for improvements, and
- the plans the city already had underway.


FIGURE 11: DAILY AND RECREATIONAL WALKING ROUTES

Ultimately, the survey attracted 1600 respondents to mark over 8700 routes and places on the map. The survey's results will serve as the foundation for Helsinki's walkability development program. So far, the results have been used as background information in a study about the possibilities of building an underground collector street, in a study on the enlargement of the pedestrian areas in the city center, as well as in the visioning work for the city center.

## 4.5. Digital Twins

The definition of Digital Twin is to create a digital replica of a physical object and use the twin as the target of digital interaction. In the smart city context this definition does not raise the bar that high, since the city models, especially the semantic ones, are already "digital replicas" of the city. The definition should be extended to include enough characteristics, semantics and functions in order to be meaningful. In other words, the digital twin should have a life of its own to be relevant platform for simulations and analysis.

The concept of digital twin fits also well in services such as augmented reality (AR) and virtual reality (VR). In UrbanSense -project, Finnish company xD Visuals[41] piloted a 5G and AR app to visualize city plans on the ground level using AR technologies. The 5G network was also utilized to improve the positioning service of mobile phone.

[41] https://xd-visuals.fi

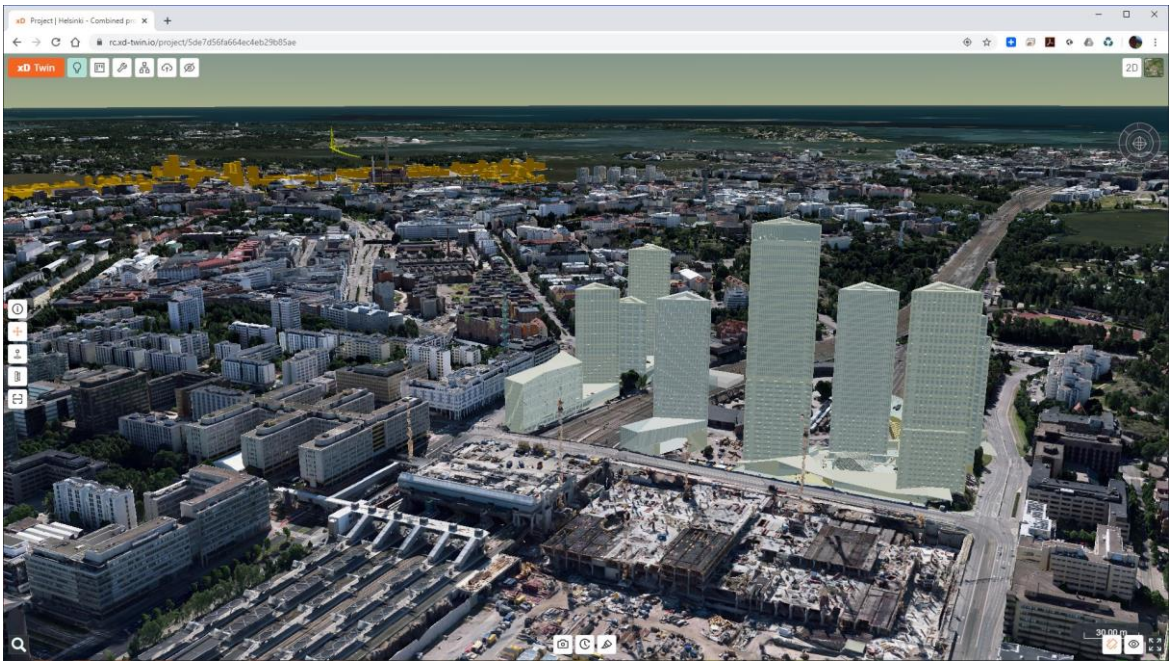


FIGURE 12: XD TWIN VISUALISATION TOOL

The requirement of the digital twin having a "life of its own" also means that as addition to static data, the model should also be capable of triggering events and processes. Events could be scheduled maintenance calls based on live data and after provide a traceable log of events to monitor the maintenance processes related to physical assets. There have been pilots on supporting this in geospatial models, however the most advanced pilot used IndoorGML instead of CityGML as a basis. The i-locate project created an implementation of the ISO 55000 asset management standard that linked geospatial features with events, processes and users[42].

[42] http://www.i-locate.eu/training/asset-management-through-extension-of-indoorgml/

# 5. Urban Open Platform Architecture

## 5.1. Espresso Reference Architecture

One of the key findings of the Espresso -project was that while all the cities are unique, they all share similar challenges such as the design, implementation and running of their open urban platform. The commonalities found from several implementations of urban platforms were identified followingly:[1]

- New open urban platforms may reuse existing urban platforms, thus improving speed and efficiency in their development. Likewise, open urban platforms may be purchased or developed and then shared by a group of cities and communities, again improving speed and efficiency, and making the use of an open urban platform more feasible for smaller cities and communities.
- Open urban platforms that are similar in different cities & communities, provide more consistency towards citizens and other stakeholders in terms of how their data is handled, how privacy and security are managed, how transparency is provided about (automated) decision making, and so on, thus supporting the mobility and free movement of stakeholders.
- The market for open urban platforms becomes larger and more interesting for suppliers of urban platform products and value adding services. Instead of having to develop a unique, tailor made urban platform for each and every city, suppliers may develop open urban platforms as a more standardized product, which can be purchased by multiple cities and communities. This may, in turn, provide more competition and may lower the costs of open urban platforms.

The Espresso -project begun the description of architecture with a capability map illustrated in the following figure.

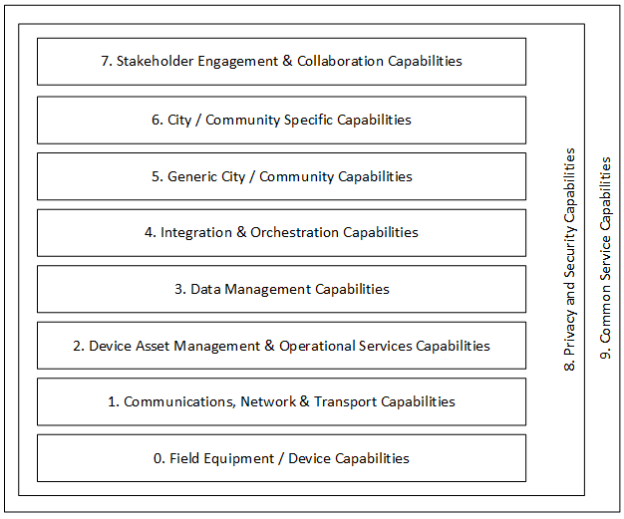


FIGURE 13: ESPRESSO CAPABILITY MAP

The map also describes the interdependency of categories: Category 0 provides the foundation to Category 1 and so on. For each category a set of capabilities has been identified. The capabilities are expected to be defined for each project so it is not feasible to define as generic capabilities. As an example, the category 4 (Integration & Orchestration Capabilities) could contain the following kind of capabilities:

- Data Exchange
- Messaging
- Load Balancing
- API Management
- Rules Management

And so on. While the definition of capabilities was not seen relevant in the FinEst Twins -project, this method might be useful as a reference in defining the overall architecture, development backlog or maintenance responsibilities especially when the urban platform is a distributes system with several actors involved.

The Espresso -project's Open Urban Platform design principles are also useful background information for future implementations:[1]

- A layered approach towards architecture is desirable, to ensure common and valuable decomposition of logical clusters, within and between standardization must take place
- Applications within the open urban platform must be linked to the capabilities to promote consistency, prevent overlap and support knowledge exchange or even re-use across cities
- The architecture of the open urban platform should allow and certainly not hinder incremental, iterative evolutionary approaches for implementation of open urban platforms
- Open urban platforms should be based as much as possible on open standards
- Modularity: the architecture of the open urban platform should enable tcities to use and deploy various modules or components
- Open urban platforms must be able to integrate with both new and existing systems taking into account that in most cases there is no 'green field'
- Open urban platforms should focus on interoperability (data formats, protocols)
- Privacy & Security are integral part of any open urban platform
- A pace-layered approach is promoted (i.e. recognizing that some modules or certain layers change faster than others)
- Urban data is currently often under-utilized, and then often in single vertical application. An important focus area of urban platforms should therefore be on harmonization of data from different domains and data logistics.
- Urban platforms should promote and facilitate the publication of (Linked) Open Data
- An important focus area of urban platforms should be on collaboration and sharing processes across domains vs. single entity/domain processes

It was also noted, that when designing and implementing an open urban platform, adopting or reusing existing public infrastructures should be considered.

## 5.2. FinEst Twins Urban Open Platform Architecture

While the FinEst Twins Urban Open Platform is an implementation of the Espresso reference architecture, some areas have evolved over the years. The Espresso architecture was seen to drive towards a monolithic, independent entity approach, yet the market trends are moving towards serverless, microservice and nanoservice architectures. The modern cloud architectures make it harder to describe the entity as a layered capability matrix and even the typical “hamburger model” of IoT platforms with southbound, data lake and northbound is misleading in many ways.

The following diagram illustrates the key functional areas of the FinEst Twins Urban Open Platform.

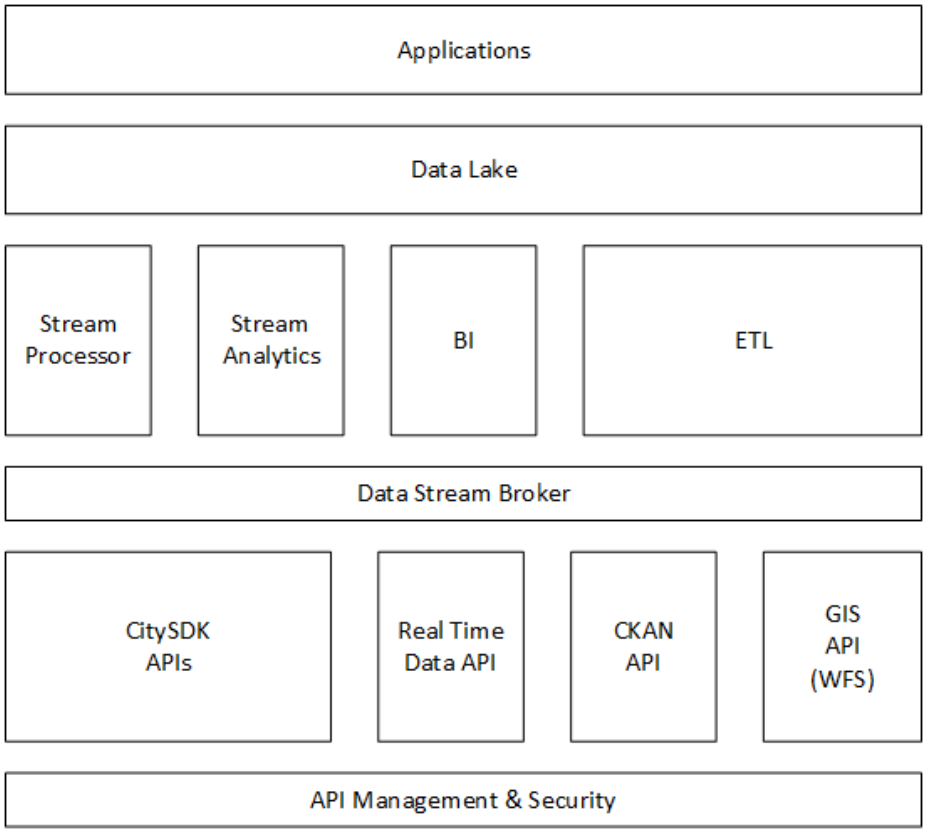


FIGURE 14: FT URBAN OPEN PLATFORM ARCHITECTURE

The platform is expected to support not only data acquisition but also various types of data processing. Data gets aggregated, processed, manipulated and extended with context. By default, any data can have the spatial component – not only as origin coordinates, but also as the abstract city model feature as an origin. Also, real time data comes in many forms. Compared to other implementations of urban data platform, the FinEst platform is not expected to directly support IoT sensor connections. It is assumed, that nowadays practically all large-scale installations would connect the platform through gateways that the sensors connect with automation field-bus networks such as BACnet in building automation.

### 5.2.1. API Management and Security

Initially, the Urban Open Platform implements an open source API gateway from tyk.io[43]. The Tyk API gateway contains a TCP proxy, security policies and session handling features. The product is cloud native, meaning that it is operated either via SSH client or management APIs.

The following diagram illustrates the Tyk API Gateway:

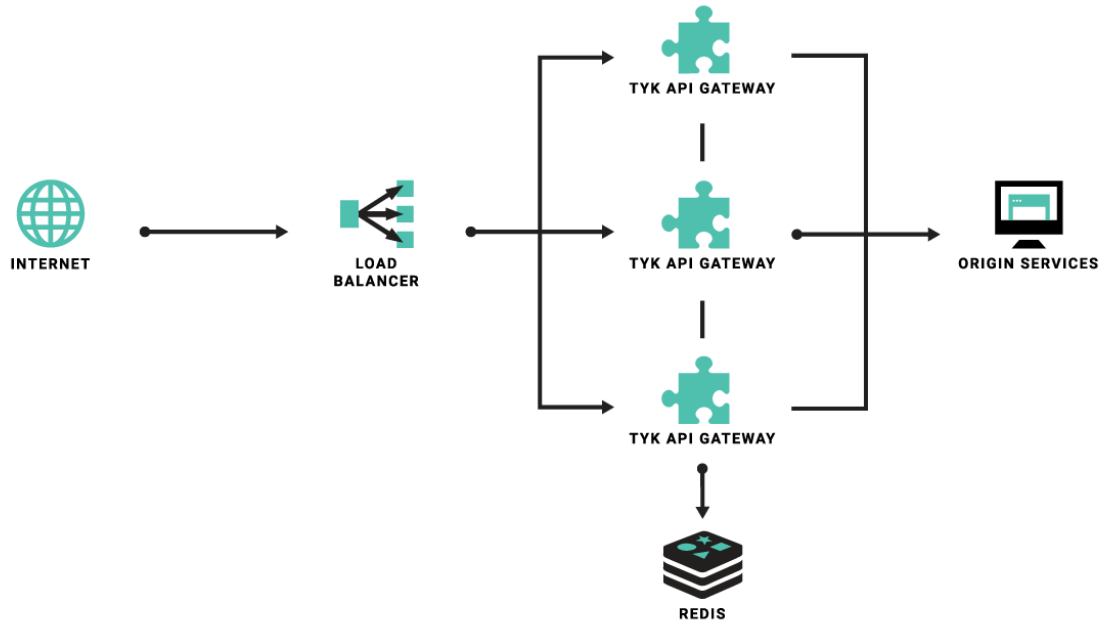


FIGURE 15: TYK API GATEWAY

The Tyk Gateway will take inbound requests, run them through a set of middleware components which apply transforms and any other service-specific operations, and then proxy the request out again to the origin, intercepting the response, running a set of response middleware and then returning. Tyk can also run custom middleware components written in JavaScript.

[43] https://tyk.io

For the southbound security, the endpoints are naturally protected with firewall rules and the hosts running the endpoints have a hardened configuration.

For secured connections to other locations such as the building automation gateways, the network traffic is routed through a VPN tunnel. The VPN (or in some cases MPLS) connections are preferred. Since setting up the VPN is oftentimes complicated and require maintenance, some of the earlier pilots have adopted the Tosibox technology to streamline the process. Tosibox, a company from Finland, manufactures network gateway devices that create an encrypted VPN tunnel to transfer the data to the cloud service. In the cloud side, the receicing end of the VPN tunnel is a virtual software that runs on server or on Azure or AWS cloud.

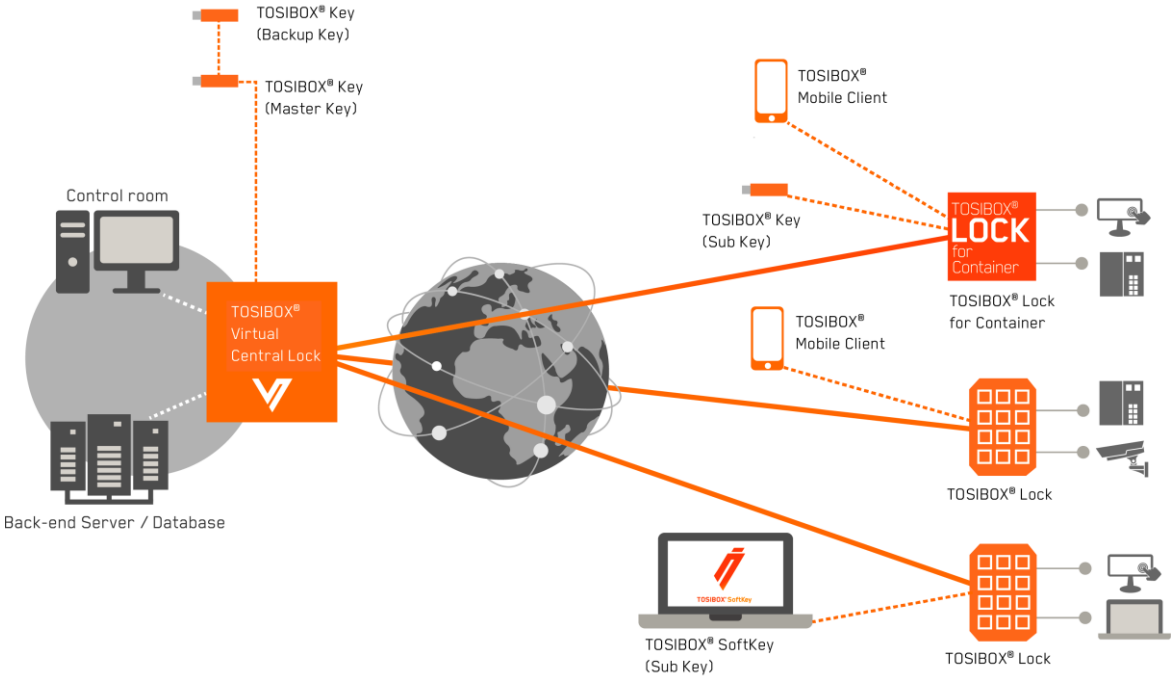


FIGURE 16: TOSIBOX SECURITY ECOSYSTEM

## 5.2.2. Southbound API's

The southbound API's are implemented as part of their own subsystem or as an endpoint to Apache Kafka.

The geospatial data API's (WMS, WFS) are based on Geoserver. It is an open source product and industry standard for sharing geospatial data.[44] The geospatial data maintained in Geoserver can be visualized with various tools, including OpenLayers and Leaflet but also with other mapping applications such as Google Maps, Microsoft Bing Maps or MapBox.

[44] https://www.geoserver.org

The CKAN API is part of the CKAN open data portal and it allows direct API access to the datasets and the metadata of datasets in the portal.

The real-time data API is following the SensorThings API standards. It allows the retrieval of sensor observations as timeseries by location. The SensorThings implementation is made by the project and the endpoint is deployed as a Kafka consumer.

### 5.2.3. Data Stream Broker

While many EIP-SCC Urban Platform projects have a centric context broker implemented, the Urban Open Platform is based on Apache Kafka. This is because the context brokers have faced performance issues and have limited data models that are a risk for semantic interoperability. Also, the security and privacy requirements have been hard to meet in previous implementations.

The capacity expectations are hard to define in the smart city context. If we assume that one day all the street lights of Helsinki were in digital control and each lamp driver had 10 datapoints, only this single use case would produce nearly a million data streams. The stream brokers are useful also in a way how they make live data more economic than stored data: the data gateway to automation system can implement all the datapoints and the decision of what is to be stored can be made in the stream broker. In this way it is possible to start ad hoc collection of new types of data without having to reconfigure or reprogram every automation system and gateway.

Apache Kafka is a distributed streaming platform. Its key capabilities are:[45]

- Publish and subscribe to streams of records, similar to message queue (MQ) or enterprise service bus (ESB)
- Store streams of records in a fault-tolerant durable way
- Process streams of records as they occur

Kafka is generally used for two types of applications:

- Building real-time streaming data pipelines that reliably get data between systems or applications
- Building real-time streaming applications that transform or react to the streams of data

Kafka is run as a cluster of three or more servers that can span multiple datacenters. The Kafka cluster stores streams of records in categories called topics. Each record consist of a key, a value and a timestamp.

Kafka has five core APIs:

- The Producer API allows an application to publish a stream of records to one or more Kafka topics.
- The Consumer API allows an application to subscribe to one or more topics and process the stream of records produced to them.

[45] https://kafka.apache.org/intro

- The Streams API allows an application to act as a *stream processor*, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.
- The Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.
- The Admin API allows managing and inspecting topics, brokers and other Kafka objects.

The following diagram illustrates the roles of apps utilizing these APIs.

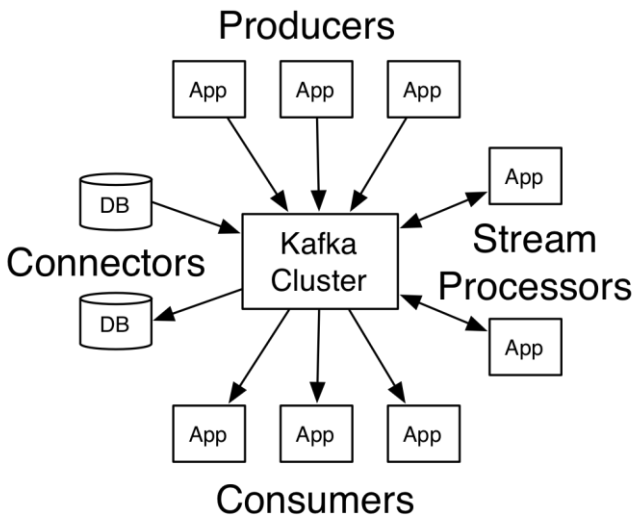


FIGURE 17: KAFKA APIS

In Kafka the communication between the clients and the servers is done with a simple, high-performance, language agnostic TCP protocol. This protocol is versioned and maintains backwards compatibility with older versions. We provide a Java client for Kafka, but clients are available in many languages.

Messaging traditionally has two models: queuing and publish-subscribe. In a queue, a pool of consumers may read from a server and each record goes to one of them; in publish-subscribe the record is broadcast to all consumers. Each of these two models has a strength and a weakness. The strength of queuing is that it allows you to divide up the processing of data over multiple consumer instances, which lets you scale your processing. Unfortunately, queues aren't multi-subscriber—once one process reads the data it's gone. Publish-subscribe allows you broadcast data to multiple processes, but has no way of scaling processing since every message goes to every subscriber.

The consumer group concept in Kafka generalizes these two concepts. As with a queue the consumer group allows you to divide up processing over a collection of processes (the members of the consumer group). As with publish-subscribe, Kafka allows you to broadcast messages to multiple consumer groups.

In many IoT platforms, the connection to sensors is created using MQTT protocol. The Urban Open Platform does not support MQTT endpoint, but it can be later added if needed.

The main benefits of using Kafka as a core component of the Urban Open Platform can be seen from the use case descriptions in Appendix B: There are very little differences between the use cases even though they support many different domains.

## 5.3. Stream Processing

Stream processing platforms are a set of processes that capture data from many different sources as and when they arrive, collect them in one place and provide a means of extracting them for analysis, visualizations, transformation and storage. Streaming platforms are also called data pipelines or ETL pipelines where ETL stands for Extract Transform and Load. The data may come from different sources at regular or irregular intervals for e.g. from a smart sensor device that emits a measurement every minute, or from social media posts about the city. It may vary in quality. It may vary in structure. But the data pipeline becomes a place to aggregate these data, validate them, transform them or structure them to prepare them for further processing, analysis, visualization and as input for other valuable services[46].

Many cloud providers, including AWS, Azure and Google Cloud Platform offer managed streaming data services and databases that can be used to build data pipelines. When it comes to open source, Apache Kafka is a very popular streaming platform[47].

When it comes to a city, there are a lot of actionable data from different sources. For e.g. noise sensors, events from a people counter at various building entrances, electricity data coming from smart energy meters, etc. Therefore, the stream processing capabilities of the Urban Data Platform can prove to be an invaluable asset for the city.

Some of the ways in which a streaming platform can be leveraged to serve the city is described in the following sections with examples.

### 5.3.1. Automated Detection of Anomalies

The main thing that differentiates a data streaming platform from other data processing platforms is that the data is ingested and actionable in real time. There is no wait for a 'batch' to arrive and then process. The data can be checked in real time to see if the incoming datapoint is an anomaly or an outlier. Let's take the data point here to mean a measurement of air quality or an event from a people counter camera at the entrance of a building. It can be detected if the air quality is rising beyond permissible levels as soon as it happens and this can then be registered with the concerned authorities. It can be detected if the air quality is suspicious in some way that could then point to a faulty equipment or give some other insight into the data. When it comes to the people counter events, it could detect a suspicious entry to a building past midnight.

The streaming platforms usually support SQL or SQL based querying which allows us to query the platform to gather these insights. They also support creating streaming applications with other programming languages like Java or Python to process the streaming data with the help of existing

[46] https://medium.com/stream-processing/what-is-stream-processing-1eadfca11b97

[47] https://kafka.apache.org/

libraries in more complicated scenarios. It can be decided based on the use case if running a set of SQL queries serves the purpose or a streaming application is needed to be deployed.

### 5.3.2. Real-time Alerting Mechanism

The automated detection of anomalies in real time can be used to alert authorities in real time by integrating a ticketing system to the Urban Data Platform. As soon as an issue is detected, the relevant information can be sent to this ticketing system.

One such ticketing system available is Zendesk[48] customer support ticketing system. It provides a REST API to which the issue can be sent programmatically from the platform. The issues sent to the platform creates tickets on the Zendesk platform. These tickets can be assigned with priorities and be assigned to the person in charge of the concerned data sources. The Zendesk platform auto-generates emails to the assigned person to notify them of the ticket

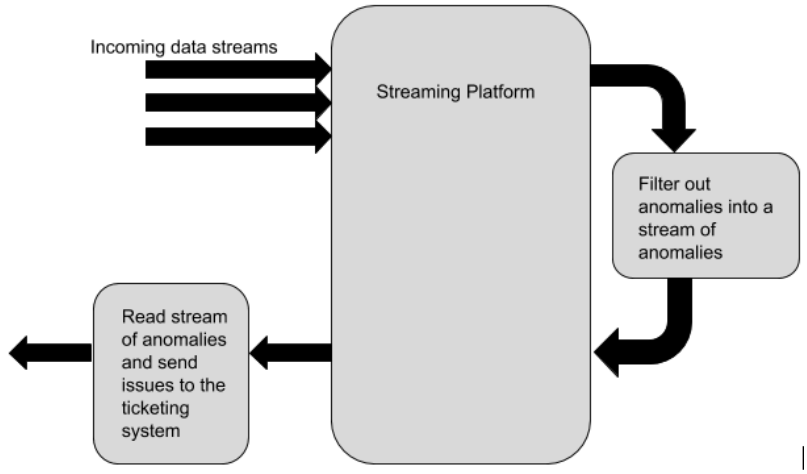


FIGURE 18: REAL TIME ALERTING

There is also the Open311[49] standard which is an open standard for civic issue tracking. Traditionally in the USA, to report issues regarding infrastructure like a pothole, a broken street light or a dysfunctional electricity meter, citizens could call at 311 - the designated phoneline given set by US government for the purpose. The Open311 standard allows for any government or city to automate this process by setting up a web service where such issues can be reported along with relevant information like the longitude and latitude of the location in question complete with a possibility to upload photographs and the other documents pertaining to the issue. This ticket can then be routed (automatically or with manual intervention) to the relevant organization or department. The reporter gets a tracking id to track the status of the reported issue and respond to any further queries.

[48] https://www.zendesk.com

[49] https://www.open311.org/

City of Helsinki has an implementation of Open311[50] as well which provides a REST API end point to which the Urban Data Platform can report issues programmatically as soon as a datapoint is detected to be an anomaly and requires attention.

### 5.3.3. Predictive Analysis Models

Analytics is defined as models of data analysis that provide predictive or forecasted insights via statistical analysis. With the variety and volume of data that a city might gather, a city committed to using its data will find many use cases but it is important to define the scope and goals of each analytics project. As per the reference[51], the applications can be as varied as predicting areas of rodent infestation in the city and predicting which landlords are most probably discriminating their tenants based on race or religion.

Analytics applications run on the streaming platform in combination with a database consisting of historical data to provide insights gathered utilizing the historical information which is also constantly updated in real time with incoming data.

## 5.4. ETL Processes

The ETL process – extract, transform, load – is a general procedure of copying data from one or more sources into a destination system which represents the data differently. The ETL processes have been around since 1970's and they still have a role in data warehousing.

In the Urban Open Platform, the ETL has two main functions:

- Create scheduled and on-demand conversions of datasets into others, or fetch data for indicator calculation
- Run critical conversions of XML-like files with data validation, e.g. to create CityGML files from other geospatial data formats

For the latter function, the de facto standard tool is FME from Safe Systems. It is available as a desktop, server and online version. The product supports over 450 data formats and applications to help the definition of data integration and transformation tasks. The transformation is defined using a drag-and-drop tool. It is also possible to extend the FME workflows with analytical code using Python or R programming languages. The tool also supports data validation which is important in critical conversions such as creating spatial datasets.

[50] https://dev.hel.fi/apis/open311/

[51] https://datasmart.ash.harvard.edu/news/article/analytics-city-government

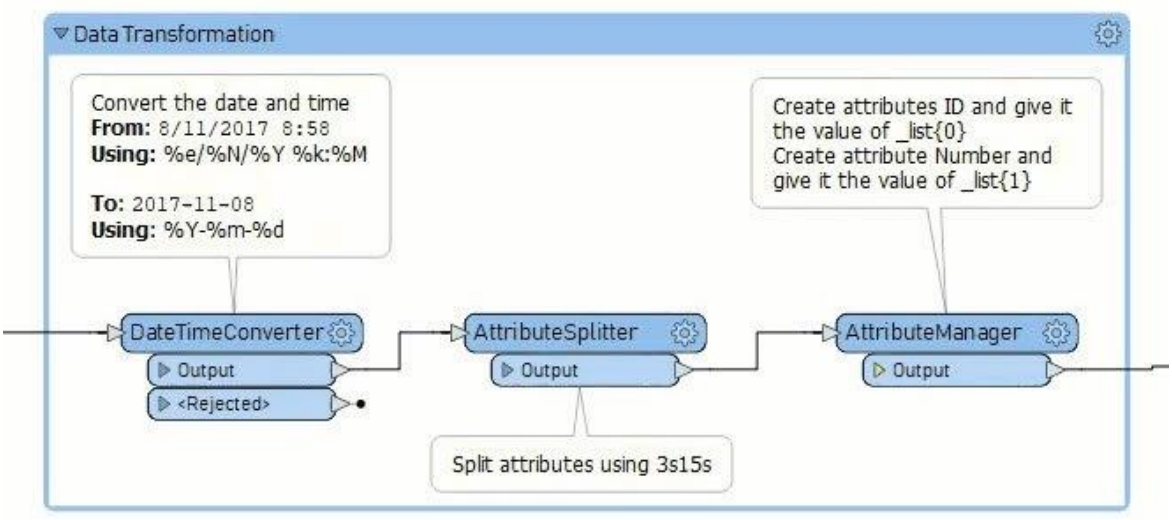


FIGURE 19: SIMPLE DATA TRANSFORMATION USING FME

The second tool used as part of the Urban Data Platform is MapForce by Altova. It is similar to FME but provides a wider set of functions to transform text or numeric values. MapForce can fetch data from databases, web services, flat text files, XML or JSON files or from REST interfaces. It can provide the output to same sources but it also supports the latest Excel file formats. The MapForce is seen as an interesting tool to automate such city reporting processes, where the report is expected to be created on Excel and uploaded to the data portal.

Both the FME and MapForce can be used without any programming skills.

## 5.5. Data Stores

The design of the Urban Open Platform attempts to keep all the options available for data storage. For geospatial datasets the PostGIS is used and it also has enough performance to store timeseries data from the first pilots. As an object key-value store, the standard PostgreSQL hstore is used.

The RDF storage requirements are covered with the implementation of SKOSMOS ontology server. It includes Apache Jena Fuseki as a SPARQL triple store.

In the next phases of the project, the role of Azure data lake is studied especially for the larger datasets. The Azure can store object data in AVRO format which is well supported by the Apache Kafka and Spark services. There is also a Kafka Sink that supports Azure data lake with minimal configuration.

The Urban Open Platform will also contain an open source data portal for piloting purposes. This service is based on CKAN project[52], which is also the technical core of the Helsinki Region Information System.

[52] https://www.ckan.org

## 5.6. Dashboards and Visualizations

In the earlier IoT platform -type urban platform projects dashboards were used to visualize the sensor observations. In many projects the approach has been simplistic, creating dashboards to visualize sensor data which seldom has value by itself. At the same time the cities have faced an increasing demand to provide reports with calculated indicators to various initiatives, e.g. the SECAP reports for the Covenant of Mayors for Climate and Energy[53]. In such practical use cases the dashboard would provide a little value when the city official is expected to provide the indicators in an Excel file. If the Urban Platform could provide the file instead, the time required for collecting the report would be minimized down to minutes.

### 5.6.1. Dashboards

The FINEST Twins project is expected to deliver customizable, widget-based dashboards. The actual technologies will be chosen at the time when the specific use cases are defined. The project is also expected to utilize the already existing platforms and licenses the cities already have. Since both Helsinki and Tallinn use the Microsoft Azure data platform, the project will also study the feasibility of using the existing Microsoft PowerBI licenses for some visualization tasks, especially in cases where some of the source data exists in the cities' information systems and access to it would require use of the internal city network.

For technical platform monitoring the Urban open Platform uses the Elastic Stack[54]. It contains Kibana -module for the visualization. The Kibana provides various types of charts. It also includes location analysis tools, supports analyzing time series data. To detect anomalies in the system performance machine learning models can be used to process the data and graph exploration can identify relationships and correlations within the data. The Elastic Stack comes with a dual license model with free open source version and commercial subscriptions, available as standalone or Software-as-a-Service -version.

[53] https://www.covenantofmayors.eu/en/

[54] https://www.elastic.co/elastic-stack

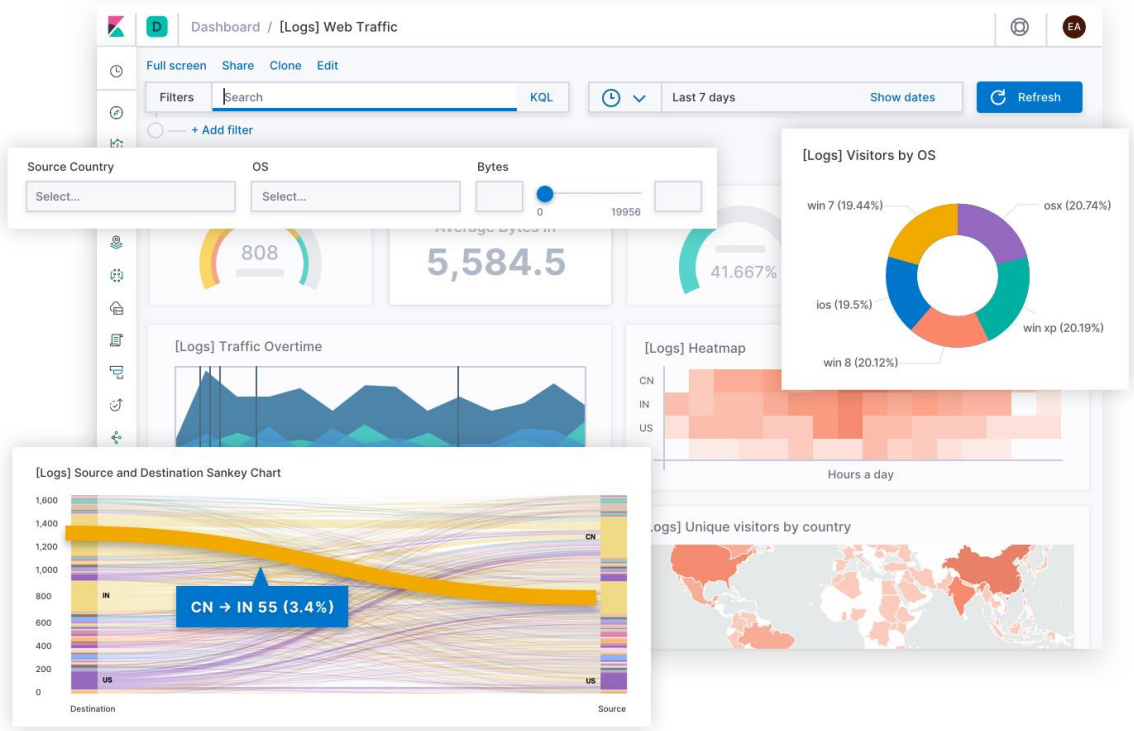


FIGURE 20: KIBANA WEB SERVER MONITORING

## 5.7. Connectivity

### 5.7.1. Mobile Networks

The role of the mobile networks in piloting phase is not currently clear. Both the cities have had experiments with the latest 5G mobile network technology. 5G is also strongly supported by the EU initiatives and according to the latest news of the budget for the next program season, there is going to be innovation programs supporting 5G adoption.

The main benefits of the 5G are higher data rates and lower latencies. The new technology also allows new options for the cloud system architecture, since the operators network platform can also host services that process the data before sending it to the customer data services. This approach is called *edge computing* and it has been piloted in Helsinki in early 2020.

### 5.7.2. LoRaWAN Networks

LoRaWAN network is a Low Power, Wide Area (LPWA) networking protocol designed to wirelessly connect battery operated 'things' to the internet in regional, national or global networks. It targets the key Internet of Things (IoT) requirements such as bi-directional communication, end-to-end security, mobility and localization services. Due to low power consumption, a typical sensor can run even several years with a single battery.

The LoRaWAN network architecture is based on a “star-of-stars” topology. Base stations act as gateways and relay messages between the sensors and central network server. The gateways communicate with the server using standard IP connections. The data rates of the radio link range from 0,3 kbps to 50 kbps. Unique 128-bit session keys secure the communication on both the network and application level.

In Finland, Digita Ltd started to build a nationwide LoRaWAN network and provide services to companies in 2018. Digita was originally formed to manage the broadcast networks in Finland and because of that history, they maintain a network of 200 broadcast masts and operate a 24/7 service operation center. The same facilities are the basis for the LoRaWAN -service, that covers well large areas of the country. In the following picture the blue coverage area represents coverage that includes building indoors.

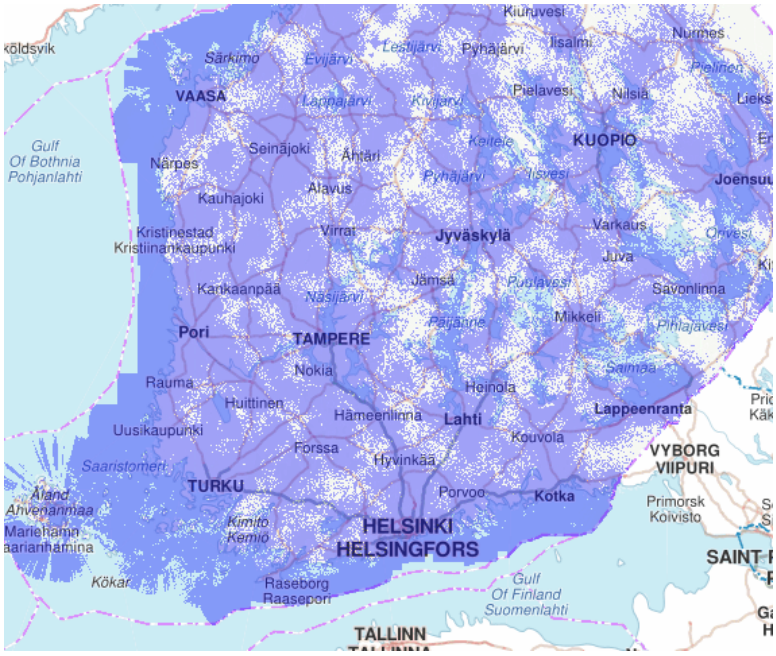


FIGURE 21: DIGITA LORAWAN COVERAGE IN FINLAND, MAY 2020

## 5.8. Urban Platform and Existing Systems

As the Espresso -project stated as one of their design principles, when designing and implementing an open urban platform, adopting/reusing public infrastructures should be considered.[1]

The cities are in an ongoing process with improving and replacing their services. According to the latest inventory, the city of Helsinki uses 967 different software products in its operations. Some of the system may have complex dependencies or have other reasons that make their replacement complicated even if the software itself is not anymore supported.

Helsinki also recently started a project to renovate its digital foundation. The project will look for a complete overhaul of systems with the objectives of being able to have functional core ICT services

with support functions, cost-effective and high-performance capacity services, fast and easy support and benefits with centralization.

When the cities improve their internal ICT service offerings, that development also benefits the work on Urban Open Platform. If the city already has a data lake or data warehouse with 24/7 support and service level agreements (SLA), utilizing them will also make the platform more reliable and easier to maintain. While in Helsinki the data lake level is to be provided, the middleware support remains an open question. Therefore, the events streaming and processing system nicely complements the Microsoft Azure platform, that will host large datasets that are made available for BI -processes with tools like ETL.

# 6. Platform Operations

## 6.1. Requirements Management

With potentially many direct and indirect users for city data, the requirements gathering and management is not a straightforward task. In fact, they could come from the users themselves, from research interests of the collaborating and partnering organizations, from the product owner and the platform team themselves or from experts from the field who might be called in for consultation or guidance[55].

## 6.2. Platform Governance Model

A governance model or framework can be described as one which facilitates the software or infrastructure development organization to go from requirements to operate in production. The stakeholders must agree upon the governance model, i.e. processes and practices that define the 'readiness' and 'doneness' of changes to the system and how to achieve them. The team shall adopt the agile way of working and DevOps practices to work incrementally on features. DevOps allow to incorporate the 'process' to a large extend into the automation.

The changes that will be implemented can be largely grouped as:

- Model for adding new Use cases/Features: This is where agile and DevOps strictly applies. The team shall follow the agile model to break work items in such a way that it fits the agreed upon window of releases. The definition of doneness is strictly observed in the deployment pipeline either as pipeline as a code or using manual interventions. This process is described in detail in the upcoming sections.

- Model for making changes to Infrastructure: The platform infrastructure which consists of the streaming service, logging service, application servers, application gateways would need maintenance from time to time. It would need changes like increasing security, upgrades, scaling up or down, retiring older services, adding new services etc. Some of these changes may need platform downtime and same may be rolled out without affecting the service. This falls in the domain of site reliability and the process of informing the stakeholders of the downtime and assessing reliability after changes shall be established.

### 6.2.1. Key roles in Platform Development

There are three main contributors to the Urban Data Platform:

- **Users:** They are the users of the Urban Data Platform. They provide the data sources for the platform and hence expect to use the platform in a way that would benefit them.
- **Product manager:** The product manager is the contact point between the stakeholders, i.e. the users and the platform team members. It is their role to be in touch regularly with the stakeholders and collect requirements for the product being built (in this case the data platform). The role of the product manager is essential because without them, the product

[55] http://agilemodeling.com/essays/agileRequirements.htm#WhereFrom

development team is out of sync with the potential users of the product and hence may end up building an unusable product.

- **Platform Team:** The team members responsible for developing, testing and deploying the applications and services that make up the Urban Data Platform.

The platform team should be experienced with the various roles and tasks that are related to data platform management. Background in testing, orchestration and configuration management, key technologies (e.g. Apache Kafka and Spark) and some programming skills are mandatory. The programming in the Urban Open Platform has been mostly done using the Python software language in Flask and Django frameworks.

## 6.3. Agile Development Process

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Working with smaller increments going from basic features of a product to more advanced features and then value additions. Each revision shall be a usable, testable product. By doing so the team is better able to respond to changes in requirements[56].

### 6.3.1. Workflow in Agile Framework

In the agile approach work is done in pieces in certain fixed periods of time called the time that can be called a timebox. The work to be done forms a 'backlog'. These work items have

1. A definition of ready i.e., it can be worked on in the upcoming timebox uninterrupted because the prerequisites are available

2. A definition of done i.e. it can be taken to the next stage. For a development task, this could mean that the feature is ready for testing. For a functional test task, this could mean that it is functional and is ready for performance testing and so on.

3. An estimate of effort required. This is estimated by the team mates and if it exceeds the timebox, the work item shall be broken down to fit the timebox.

Requirements, plans, and results are evaluated continuously, at least once during every timebox so as to keep the backlog of tasks up to date. Therefore, the teams have a natural mechanism for responding to change quickly.

### 6.3.2. Kanban

Kanban is one of the many frameworks used for agile software development. In Kanban work items are visually represented on a kanban board (physical or online) and arranged under columns representing different stages, for example To-do, Work-In-Progress, In-Review, Blocked, Merged etc. depending upon what is agreed upon by the team. This provides a visual tracking of the flow of work. The team is usually free to choose its own framework and to what extent it chooses to stick to a

[56] https://www.agilealliance.org/agile101/

framework. There are no rules and almost always the frameworks need to be customized to suit the needs of the team.

## 6.4. DevOps

The agile methodologies were adopted by the developer community first, increasing the speed of their work output which resulted in a 'gap' between the developers and the operations team and 'DevOps' way of working addresses this gap[57].

DevOps refers to strong collaboration between operations, development and testing of software. DevOps complements agile in such a way that each deployable incremental unit of software (in the agile iterative model) developed, is tested and deployed continuously and hence released as a functioning product.

Essentially there are three pillars of DevOps as per [https://theagileadmin.com/what-is-devops/]

**Infrastructure Automation** – create your systems, OS configs, and app deployments as code.

**Continuous Delivery** – build, test, deploy your apps in a fast and automated manner.

**Site Reliability Engineering** – operate your systems; monitoring and orchestration, sure, but also designing for operability in the first place.

## 6.5. Infrastructure Automation

The team shall utilize, wherever possible, tools like Ansible and Docker for IT infrastructure automation and deployment of applications.

Ansible is an infrastructure automation tool which allows to specify jobs for installing, upgrading, uninstalling software packages in a YAML script as long as the host system which runs Ansible can connect (usually by SSH) to the inventory or the destination systems where the software packages are to reside[58]. The destination systems are typically virtual machines on the cloud.

Docker is a containerization technology which allows developers to create a platform independent standalone environment for their code to run. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings[59]. Containerization has numerous benefits. It isolates and standardizes your application. Therefore, it allows to create identical environments for development and testing which eliminates the 'it works on my machine' problem that often arises when the configuration of the dev environment and test environment is not the same. It also facilitates automated deployment using container orchestration technologies like Kubernetes. Since the data platform will have many microservices sending in different kinds of data, it will need a good orchestration engine like Kubernetes.

[57] http://www.jedi.be/presentations/IEEE-Agile-Infrastructure.pdf

[58] https://www.ansible.com/overview/how-ansible-works

[59] https://www.docker.com/resources/what-container

## 6.6. Configuration Management

Container images hold the configuration pre-built and pre baked in them. Since the applications around the data streaming platform are containerized, they do not need a configuration management system. In other words, the docker container registry holds the configuration information. The configuration of the underlying systems need management. For e.g., setting up the Kubernetes if it is being used for orchestration, setting up monitoring systems, i.e. ELK stack and the setting up the streaming platform itself. This can be done with Ansible.

## 6.7. Continuous Delivery

The pillar of infrastructure automation essentially makes continuous delivery possible which is ultimate goal or the output of 'DevOps'. Continuous delivery[60] starts with continuous integration[61].

For a successful continuous integration and delivery (CI/CD), there must be 3 different configurations/environments of the 'system' called the development environment, staging environment and production environment. A deployment pipeline[62] is created which is, essentially scripts (therefore Infrastructure as Code or more specifically Pipeline as code). The scripts poll the version control system (e.g. Git) for new commits and when there is a new commit, it checks out the updated codebase to a machine/container, builds the code and runs all the tests in the dev environment. If the tests pass, the codebase is then built and tested in the staging environment. The test coverage shall be ideal in the staging environment and the configuration of the staging environment shall be as identical to production as possible. If the tests pass here, then the software is ready for production. The production deployment can be fully automated but can also have manual checkpoints.

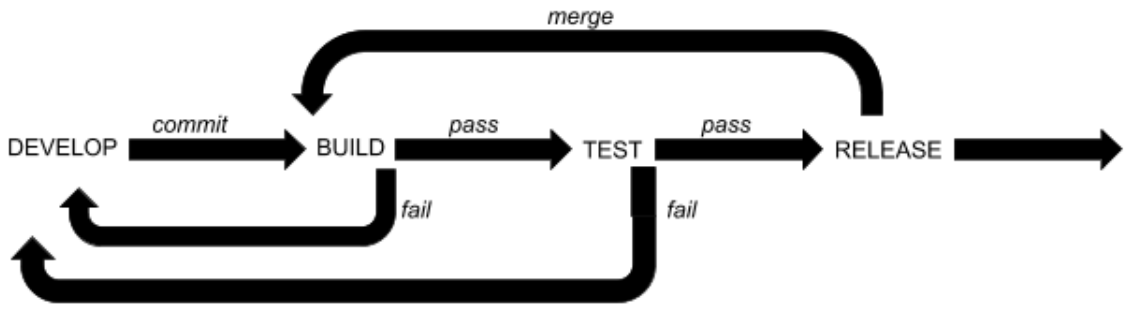


FIGURE 22: CONTINUOUS INTEGRATION (CI)

[60] https://martinfowler.com/bliki/ContinuousDelivery.html

[61] https://martinfowler.com/articles/continuousIntegration.html

[62]https://medium.com/the-making-of-whereby/what-is-a-deployment-pipeline-and-how-it-helps-software-development-teams-6cb29917ceea

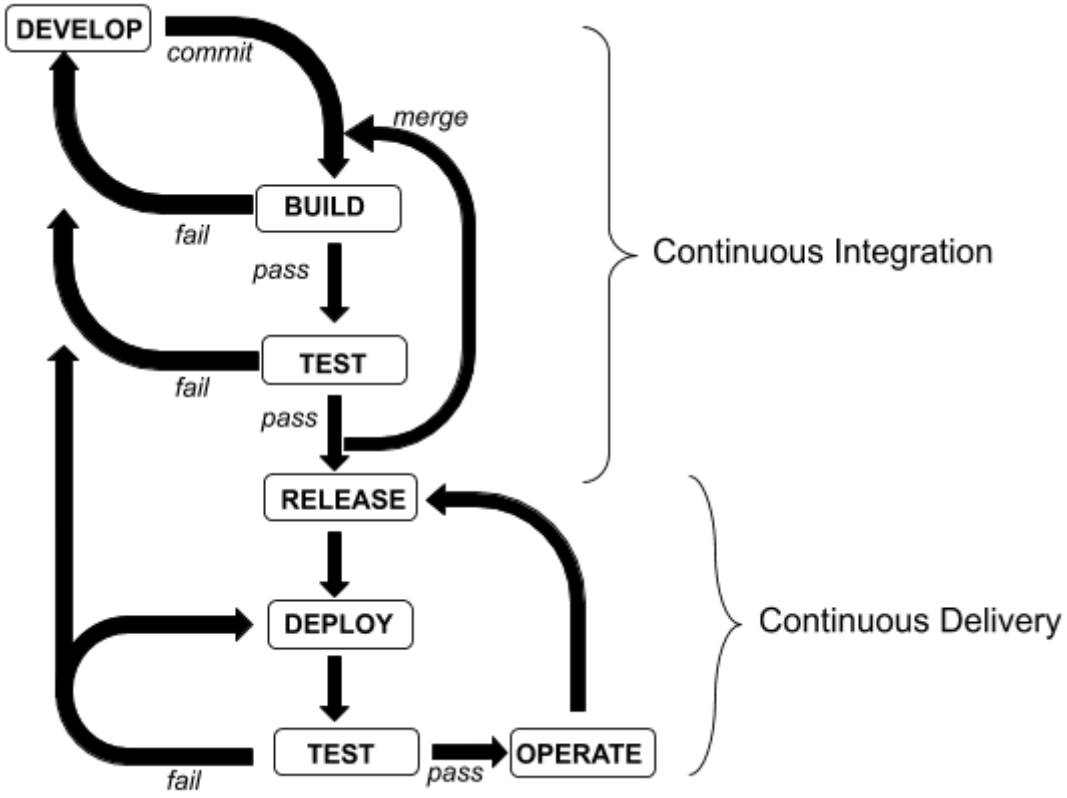


FIGURE 23: CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY (CI/CD)

The deployment pipeline can be created in GitLab CI/CD with the addition of container orchestration tools. The pipeline as code model creates automated processes that help developers build applications better and faster.[63]

## 6.8. Site Reliability Engineering

Availability, Scalability and Security constitutes site reliability. A certain service level is agreed upon with the stakeholders and the site reliability engineers have the responsibility ones with the responsibility of engineering the system to adhere to those agreed service levels.

It is observability or monitoring that makes a system reliable. The culture of observability revolves around the logs, metrics and traces collected from our systems. These datapoints collected over time, once visualized provide an insight into the system, network and applications. These insights are then used to not just avoid mishaps but constantly optimize the performance of the system. Observability systems makes it possible to measure performance, thereby collecting feedback from the system. The feedback unless addressed does nothing for reliability.

The data streaming platform is initially set up based on the expected load of streaming data. Monitoring of these system parameters, allows the team to scale up or down as necessary to reduce

[63] https://about.gitlab.com/blog/2019/07/12/guide-to-ci-cd-pipelines/

unnecessary cost when data flow is low or expected to be low or avoid crashes due to reaching memory limit or processor limit during times of high traffic.

The observability systems also quantify the service level of the platform that the stakeholders are interested in.

## 6.9. Monitoring

Logs, Metrics and Traces form three pillars of observability in distributed systems.[64]

Logs are the information printed out from running systems and applications. It is absolutely essential for troubleshooting and debugging. In distributed systems where we have multiple VMs running in the cloud it is next to impossible to log into each system and look through the logs. In fact, when it comes to containerized applications, containers do not persist their logs. When a container goes down, the logs are gone too. In such a scenario, it is important to collect these logs from containers and servers to a central server where we can then visualize the different logs and outputs from different services side by side and make sense of the system state. The most popular system of tools used for this is the Elasticsearch - Logstash - Kibana, also known as ELK stack. It has agents on systems to collect logs and tools to index them, visualize them and search them. Logstash transforms logs, Elasticsearch indexes the logs (as documents in ES terminology) and Kibana is used to visualize them. The team uses ELK stack and Elastic APM for log aggregation as well as application performance monitoring.

Metrics are the charts and graphs that reflect the system health, application health and network health over time. A metrics observability system, therefore requires a timeseries database to store the datapoints and a visualizer tool to plot the graphs and charts from the data points. The most commonly used tools for this purpose are Prometheus and Grafana. Prometheus and Grafana shall be set up for the streaming platform.

Traces are particularly important in distributed systems and in application performance monitoring. It traces transaction between networked systems in a networked architecture, gives the span information, i.e., how the packets are travelling or how the app is travelling, how long it is taking etc. Jaeger and Zipkin are available options for traces and shall be used if and when complexity increases and need arises.

[64] https://www.oreilly.com/library/view/distributed-systems-observability/9781492033431/ch04.html

# 7. Urban Platform Deployment Plan

## 7.1. Project Plan

The FINEST Twins Description of Action (DoA) defines the key role of the Urban Open Platform (UoP) in supporting the project vision of having a globally unique focus on developing user-driven clean and sustainable smart city solutions that are “cross-border-by-default” in the context of linking the cities of Tallinn and Helsinki deeper together. The UoP is expected to develop and implement Research and Innovation pilots together with the UoP.Lab, public and private organizations.

## 7.2. Key Milestones

The Description of Action sets the following deadlines for the platform development:

- M6 (May, 2020): Specification and Plan for Urban Open Platform
- M12 (November, 2020): Urban Open Platform Beta ready, transfer to CoE begins
- M24 – M72: Reports on UoP developers’ engagement and functionalities development

As addition to the set deliverable deadlines, the following milestones can be derived from other project activities:

- M12 (November 2020): UoP.Lab developer engagement
- M24 (November 2021): Implementation of the production version of UoP
- M36 (November 2022): Takeover of platform to FT CoE

## 7.3. The Scope of the UoP Beta

The Urban Open Platform will be delivered to the FinEst Twins Smart City Center of Excellence (FT-CoE) for testing as a beta in the end of November, 2020 (M12).

The definition of beta is a nearly complete prototype of a product. Since the UoP heavily relies on existing software products, information systems and the needs of the SRIA are not fully clear, the prototype version is seen as a product that can demonstrate a set of use cases as proof of concept. The software products used in the configuration are chosen to support the scalability and performance expectations of production use in two major cities. It is expected, that over time the platform should be capable of operating millions of data streams simultaneously. This can be achieved by scaling up the implementation (e.g. adding new server nodes). The Beta version will however run in a temporary hosting platform and will be migrated to another platform when the work progresses.

In order to demonstrate the support for the future pilots on various domains, the UoP Beta has implemented the following use cases (for more details, see Appendix B):

Use Case 1: Environmental noise monitoring

Use Case 2: Smart Home Sensor Using LoRaWAN network

Use Case 3: Solar Panel Monitoring

Use Case 4: Smart Street Lighting

Use Case 5: People Counters

Use Case 6: Electrical Vehicle Charging Monitoring

Use Case 7: Maintenance Vehicle Telemetry

Use Case 8: Building Automation System

Use Case 9: Dynamic Attributes in 3D City Model

Use Case 10: Natural Language Processing in Helpdesk

The use cases have been selected to support all the research streams and the current interests of the cities. All the use cases have been developed using real data from the Helsinki city operations. Not all the data is open.

The Use Cases will naturally depend on data interfaces. The following APIs are used to support the pilots and research:

1. OGC SensorThings API
2. CitySDK Open311 Feedback API
3. CitySDK Resource Reservation API
4. CKAN Data API
5. OGC API for Features (formerly WFS)

Each of the use cases may add another API's to the list. As an example, the building automation use case utilizes the BACnet Web Service API of the building automation system.

## 7.4. Urban Open Platform Development After Beta

After the beta phase is completed and testing starts, the FVH platform team will continue to support the UoP.Lab and the FinEst Twins Smart City Center of Excellence in taking over the platform in production. Meanwhile, larger tasks such as the integration of Urban Open Platform and the cities geographical data systems (including the 3D City Model) is continued. It is expected that when the OGC approves the new CityGML version in the latter half of 2020 there would be production-ready implementations of the city model in early 2021. This would also introduce the OGC API for Features (formerly WFS) to be part of the Urban Open Platform.

# APPENDIX A. ACRONYMS

| Acronym | Description |
|---|---|
| **API** | Application Programming Interface |
| **FT-CoE** | FinEst Twins Smart City Center of Excellence |
| **CSV** | Comma Separated File |
| **EIP** | European Innovation Partnership |
| **ELK** | Elastic Logstash Kibana |
| **ESPRESSO** | systemic Standardization approach to Empower Smart citieS and cOmmunities |
| **ETL** | Extract – Transform - Load |
| **FVH** | Forum Virium Helsinki Oy |
| **GIS** | Geographic Information System |
| **GML** | Geographical Markup Language |
| **ISO** | International Organization for Standardization |
| **JSON** | JavaScript Object Notation |
| **KPI** | Key Performance Indicators |
| **OBIX** | Open Building Information Exchange |
| **OGC** | Open Geospatial Consortium |
| **OWL** | Web Ontology Language |
| **PPGIS** | Public Participation GIS |
| **RDF** | Resource Definition Framework |
| **REST** | Representational State Transfer |
| **SCC** | Smart Cities and Communities |
| **SECAP** | Sustainable Energy and Climate Action Plan |
| **SKOS** | Simple Knowledge Organization System |
| **SSH** | Secure Shell |
| **UOP** | Urban Open Platform |

| | |
|---|---|
| **WFS** | Web Feature Service |
| **XML** | Extended Markup Language |
| **YAML** | Yet Another Markup Language |
| | |
| | |

TABLE 1. ACRONYMS

# APPENDIX B. USE CASES

## UC1: ENVIRONMENTAL NOISE MONITORING

| DOMAIN | ENVIRONMENT |
|---|---|
| DESCRIPTION | Measure environmental noise levels and generate daily aggregated values |
| MOTIVATION | Generate meaningful indicators from raw sensor data |

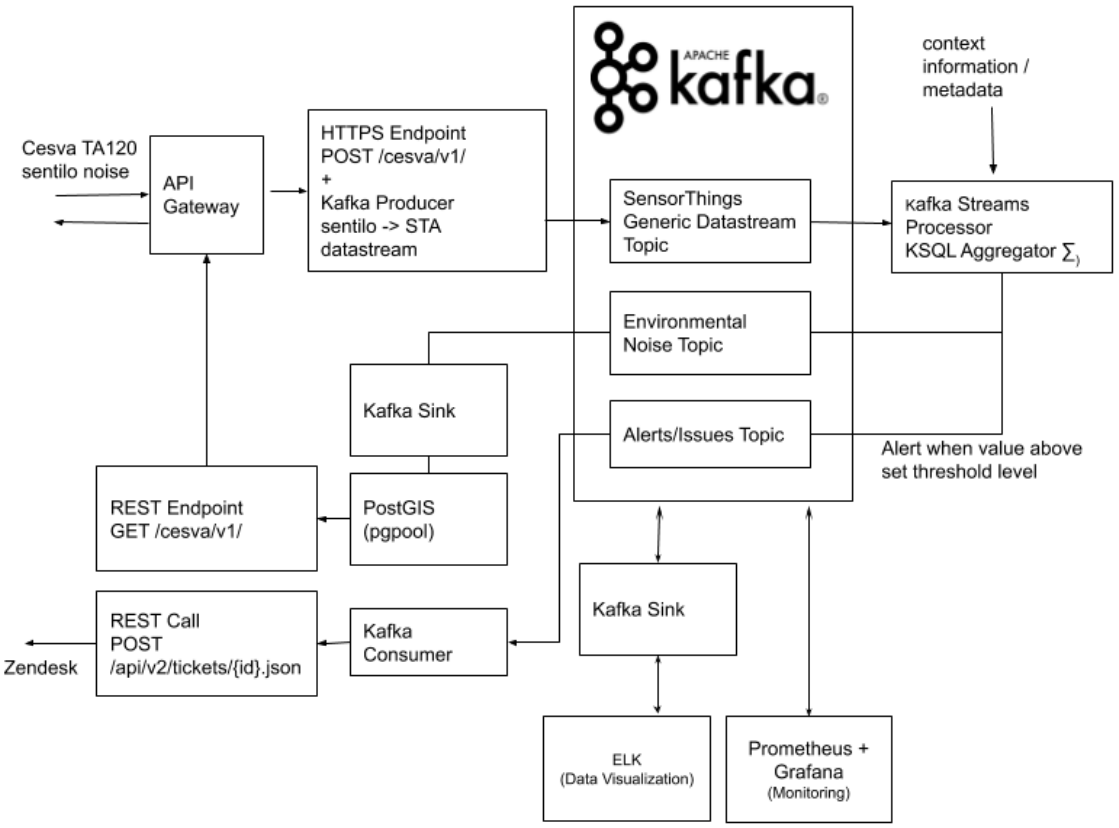


FIGURE 24: UC1 NOISE MONITORING DATA FLOW

The incoming data is a stream of events indicating noise levels in a POST request to a specific url which reaches the platform first at the API Gateway

The API Gateway serves many purposes some of which are

- having a common endpoint for multiple backend services of the platform
- authorising the access to the platform using API Keys
- setting a rate limit of requests to the backend HTTPS end-point

The HTTPS end-point receives the noise data as a POST request to the corresponding url from the API gateway. The POST data is parsed and converted into SensorThings API compliant format and fed(produced) into Kafka.

A Kafka streams application or a KSQL application obtains aggregates/summary from the generic stream and creates a new aggregate stream back into Kafka. Another stream processor application filters out anomalies like louder than a specific upper limit and creates a kafka stream of these anomalies with the required information.

The stream of anomalies is read by a Kafka consumer and a ticket is raised in Zendesk via a POST request to its REST API. The necessary context information if required in the processing should be possible to be obtained in the stream processor application using a GET request to the relevant url of the asset data store for the stream in question.

The required streams ( the generic one, the transformed one (of aggregates/summary) , the stream of anomalies ) are sinked (via Kafka Connectors) into the postgres (postgis) database as a store of historical data.

An outgoing endpoint that processes GET requests to obtain a certain measurements of the environmental noise in a way complaint with the Sensor Things API spec shall be served by another backend service that would query the Postgres database.

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The noise levels, aggregates and limit alerts info can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

## UC2: LORAWAN ROOM SENSOR

| DOMAIN | BUILT ENVIRONMENT |
|---|---|
| DESCRIPTION | Monitor room conditions with typical observations (temperature, humidity, CO2) |
| MOTIVATION | Create monitoring network that can scale to large numbers of locations |

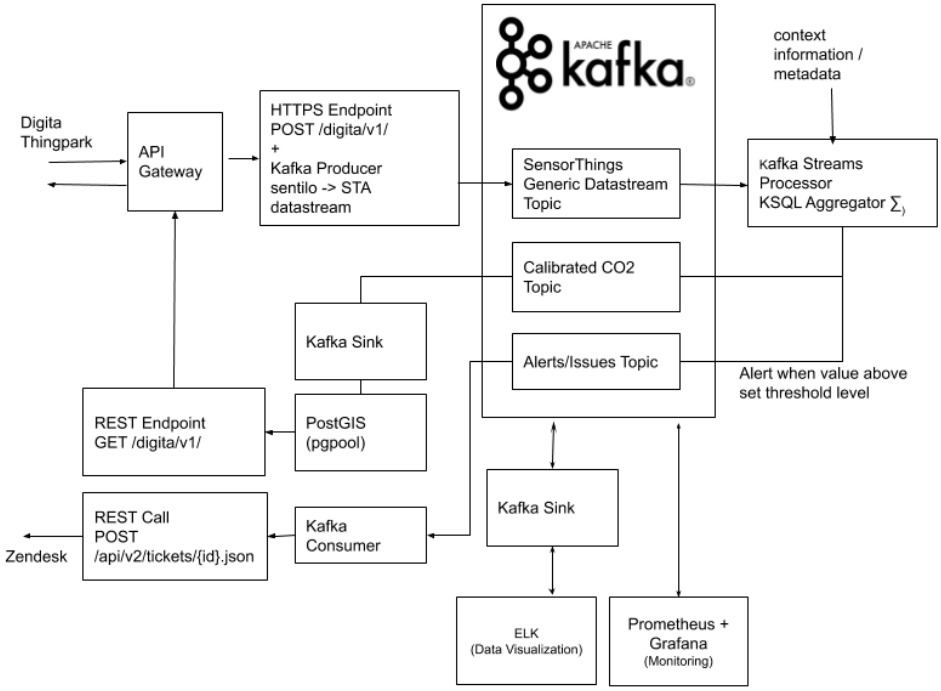


FIGURE 25: UC2 ROOM SENSOR DATA FLOW

The incoming data is a stream of events indicating measurements in a POST request to a specific URL which reaches the platform first at the API Gateway

The API Gateway serves many purposes some of which are

- having a common endpoint for multiple backend services of the platform
- authorizing the access to the platform using API Keys
- setting a rate limit of requests to the backend HTTPS end-point

The HTTPS end-point receives the input data as a POST request to the corresponding URL from the API gateway. The POST data is parsed and converted into SensorThings API compliant format and fed(produced) into Kafka.

A Kafka streams application or a KSQL application obtains aggregates/summary from the generic stream and creates a new aggregate stream back into Kafka. Another stream processor application filters out anomalies and creates a Kafka stream of these anomalies with the required information.

The stream of anomalies is read by a Kafka consumer and a ticket is raised in Zendesk via a POST request to its REST API. The necessary context information if required in the processing should be possible to be obtained in the stream processor application using a GET request to the relevant url of the asset data store for the stream in question.

The required streams (the generic one, the transformed one (of aggregates/summary), the stream of anomalies) are sinked (via Kafka Connectors) into the PostgreSQL (PostGIS) database as a store of historical data.

An outgoing endpoint that processes GET requests to obtain measurements in a way compliant with the Sensor Things API spec shall be served by another backend service that would query the PostGIS database.

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The aggregates and limit alerts info can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

UC3: SOLAR PANEL DATA ACQUISITION

| DOMAIN | ENERGY, RES |
|---|---|
| DESCRIPTION | Measure the production values (power, energy, voltage) of solar panels |
| MOTIVATION | Generate accurate production profiles for RES |

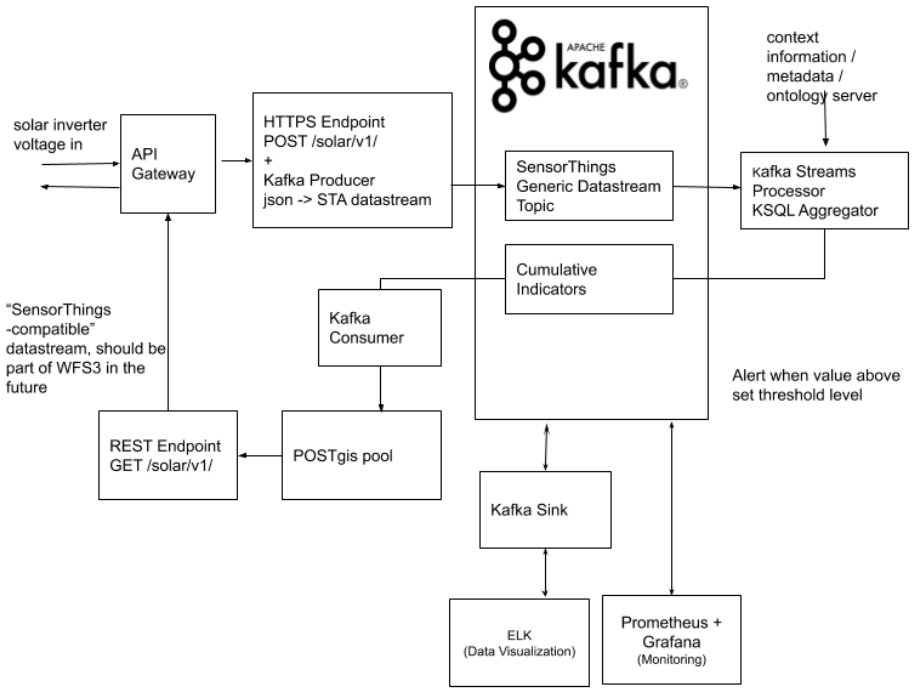


FIGURE 26: UC3 SOLAR PANEL DATA FLOW

The incoming data is a stream of events indicating measurements in a POST request to a specific URL which reaches the platform first at the API Gateway.

The API Gateway serves many purposes some of which are

- having a common endpoint for multiple backend services of the platform
- authorizing the access to the platform using API Keys
- setting a rate limit of requests to the backend HTTPS end-point

The HTTPS end-point receives the input data as a POST request to the corresponding url from the API gateway. The POST data is parsed and converted into SensorThings API compliant format and fed(produced) into Kafka.

A Kafka streams application or a KSQL application obtains aggregates/summary from the generic stream and creates a new aggregate stream back into Kafka. The necessary context information if required in the processing should be possible to be obtained in the stream processor application using a GET request to the relevant URL of the asset data store for the stream in question.

The required streams (the generic one, the transformed one (of aggregates/summary), the stream of anomalies) are sinked (via Kafka Connectors) into the PostgreSQL (PostGIS) database as a store of historical data.

An outgoing endpoint that processes GET requests to obtain measurements in a way compliant with the Sensor Things API spec shall be served by a backend service that would query the PostGIS database.

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The aggregates info can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

UC4: SMART STREET LIGHTING

| DOMAIN | BUILT ENVIRONMENT, ENERGY |
|---|---|
| DESCRIPTION | Monitor street lighting system |
| MOTIVATION | Monitor large scale system and create meaningful operative indicators |

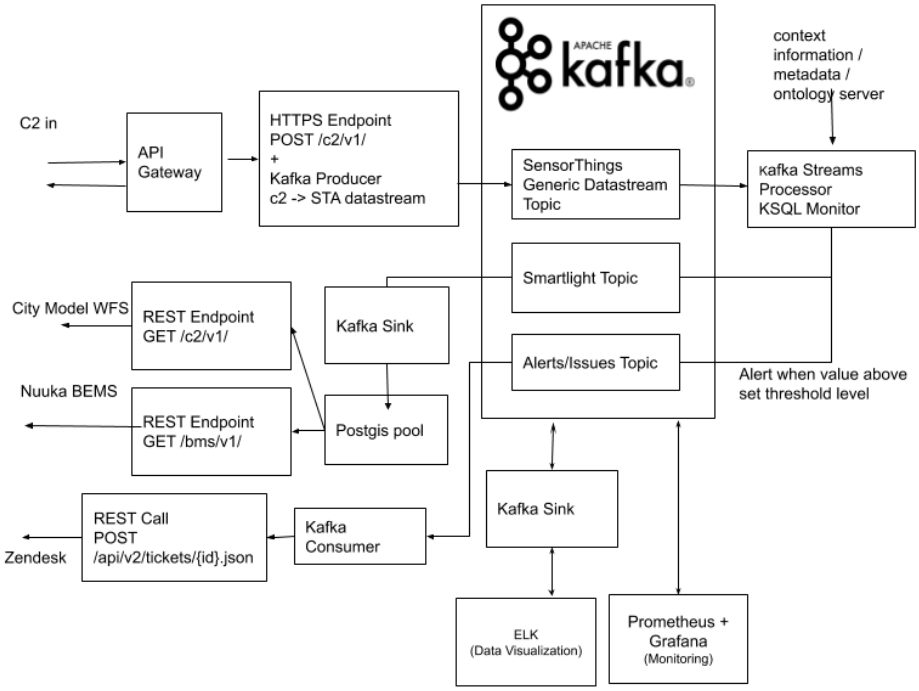


FIGURE 27: UC4 SMART STREET LIGHTING DATAFLOW

The incoming data is a stream of events indicating measurements in a POST request to a specific URL which reaches the platform first at the API Gateway

The API Gateway serves many purposes some of which are

- having a common endpoint for multiple backend services of the platform
- authorizing the access to the platform using API Keys
- setting a rate limit of requests to the backend HTTPS end-point

The HTTPS end-point receives the input data as a POST request to the corresponding url from the API gateway. The POST data is parsed and converted into SensorThings API compliant format and fed(produced) into Kafka.

A Kafka streams application or a KSQL application obtains aggregates/summary from the generic stream and creates a new aggregate stream back into Kafka. Another stream processor application filters out anomalies and creates a Kafka stream of these anomalies with the required information.

The stream of anomalies is read by a Kafka consumer and a ticket is raised in Zendesk via a POST request to its REST API. The necessary context information if required in the processing should be possible to be obtained in the stream processor application using a GET request to the relevant URL of the asset data store for the stream in question.

The required streams (the generic one, the transformed one (of aggregates/summary), the stream of anomalies) are sinked (via Kafka Connectors) into the PostgreSQL (PostGIS) database as a store of historical data.

An outgoing endpoint that processes GET requests to obtain measurements in a way compliant with the SensorThings API spec shall be served by another backend service that would query the PostGIS database. The service shall also serve another end point that supports a building management platform (Nuuka BEMS in this example)

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The aggregates and limit alerts info can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

UC5: PEOPLE COUNTERS

| DOMAIN | BUILT ENVIRONMENT |
|---|---|
| DESCRIPTION | Measure occupancy levels and profiles in a building |
| MOTIVATION | Support energy efficiency measurements by providing building usage information as context |

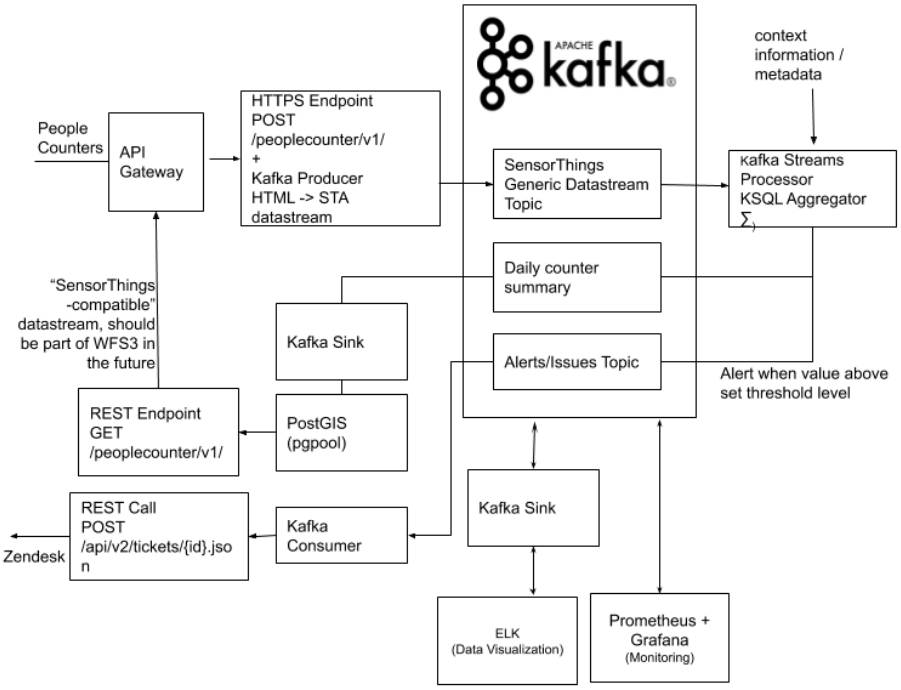


FIGURE 28: UC5 PEOPLE COUNTER DATA FLOW

The incoming data is a stream of events indicating the number of people entering and leaving a building, usually obtained from a camera device in a POST request to a specific URL which reaches the platform first at the API Gateway

The API Gateway serves many purposes some of which are

- having a common endpoint for multiple backend services of the platform
- authorizing the access to the platform using API Keys
- setting a rate limit of requests to the backend HTTPS end-point

The HTTPS end-point receives the people counter data as a POST request to the corresponding URL from the API gateway. The POST data is parsed and converted into SensorThings API compliant format and fed(produced) into Kafka.

A Kafka streams application or a KSQL application obtains aggregates/summary from the generic stream and creates a new aggregate stream back into Kafka. Another stream processor application filters out anomalies like entry at a suspicious hour, number of people in the building exceeding allowed limit etc. and creates a Kafka stream of these anomalies with the required information.

The stream of anomalies is read by a Kafka consumer and a ticket is raised in Zendesk via a POST request to its REST API. The necessary context information if required in the processing should be possible to be obtained in the stream processor application using a GET request to the relevant URL of the asset data store for the stream in question.

The required streams (the generic one, the transformed one (of aggregates/summary), the stream of anomalies) are sinked (via Kafka Connectors) into the PostgreSQL (PostGIS) database as a store of historical data.

An outgoing endpoint that is a GET request to obtain a certain measurement of the people counter as per the Sensor Things API shall be served by another backend service that would query the Postgres database.

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The daily counter and alerts info can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

UC6: ELECTRICAL VEHICLE CHARGING

| DOMAIN | MOBILITY (ENERGY) |
|---|---|
| DESCRIPTION | Measure usage and energy consumption of electrical vehicle chargers |
| MOTIVATION | Generate meaningful indicators on electrified traffic |

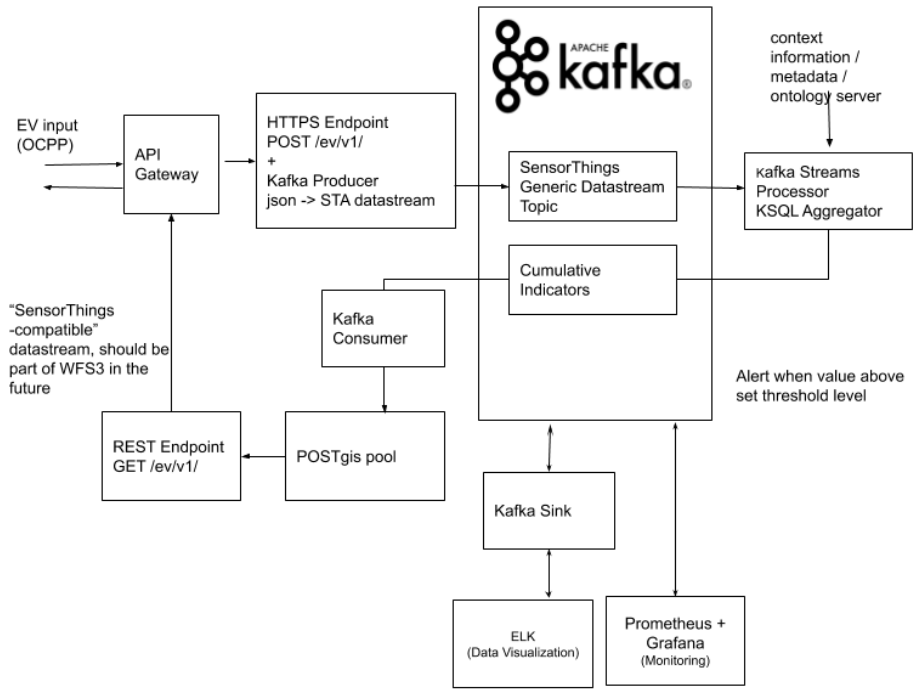


FIGURE 29: UC6 EV CHARGING DATA FLOW

The incoming data is a stream of events indicating measurements in a POST request to a specific URL which reaches the platform first at the API Gateway

The API Gateway serves many purposes some of which are

- having a common endpoint for multiple backend services of the platform
- authorizing the access to the platform using API Keys
- setting a rate limit of requests to the backend HTTPS end-point

The HTTPS end-point receives the measurement data as a POST request in OCPP protocol format to the corresponding URL from the API gateway. The POST data is parsed and converted into SensorThings API compliant format and fed(produced) into Kafka.

A Kafka streams application or a KSQL application obtains aggregates/summary from the generic stream and creates a new aggregate stream back into Kafka. The necessary context information if required in the processing should be possible to be obtained in the stream processor application using a GET request to the relevant URL of the asset data store for the stream in question.

The required streams (the generic one, the transformed one (of aggregates/summary), the stream of anomalies) are sinked (via Kafka Connectors) into the PostgreSQL (PostGIS) database as a store of historical data.

An outgoing endpoint that processes GET requests to obtain measurements in a way that is compliant with the Sensor Things API shall be served by another backend service that would query the PostGIS database.

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The measurements info like summaries etc. can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

UC7: MAINTENANCE VEHICLE TELEMETRY

| DOMAIN | MOBILITY |
|---|---|
| DESCRIPTION | Collect data of moving vehicles operations |
| MOTIVATION | Wireless data collection, dynamic attributes and features on city model |

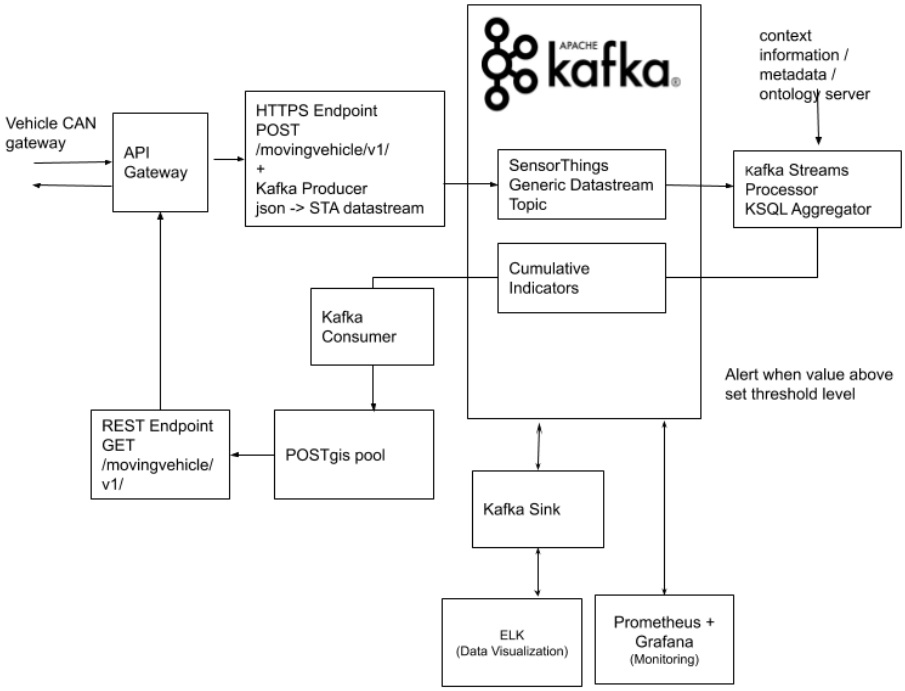


FIGURE 30: UC7 MOVING VEHICLE TELEMETRY

The incoming data is a stream of events of datapoints collected from the vehicle CAN bus using an IoT gateway device in a POST request to a specific URL which reaches the platform first at the API Gateway

The API Gateway serves many purposes some of which are

- having a common endpoint for multiple backend services of the platform
- authorizing the access to the platform using API Keys
- setting a rate limit of requests to the backend HTTPS end-point

The HTTPS end-point receives the measurement data as a POST request in OCPP protocol format to the corresponding URL from the API gateway. The POST data is parsed and converted into SensorThings API compliant format and fed(produced) into Kafka.

A Kafka streams application or a KSQL application obtains aggregates/summary from the generic stream and creates a new aggregate stream back into Kafka. The necessary context information if required in the processing should be possible to be obtained in the stream processor application using a GET request to the relevant URL of the asset data store for the stream in question.

The required streams (the generic one, the transformed one (of aggregates/summary), the stream of anomalies) are sinked (via Kafka Connectors) into the PostgreSQL (PostGIS) database as a store of historical data.

An outgoing endpoint that processes GET requests to obtain measurements in a way that is compliant with the Sensor Things API shall be served by another backend service that would query the PostGIS database.

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The measurements info like summaries etc. can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

## UC8: BUILDING AUTOMATION

| DOMAIN | ENVIRONMENT |
|---|---|
| DESCRIPTION | Measure environmental noise levels and generate daily aggregated values |
| MOTIVATION | Generate meaningful indicators from raw sensor data |

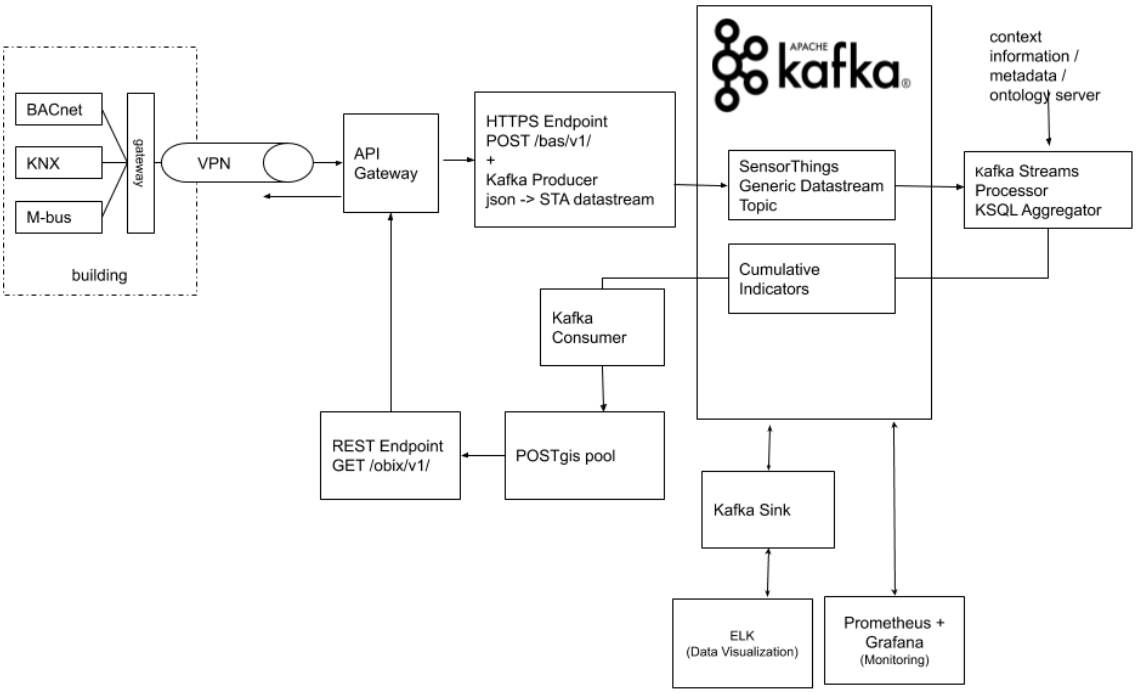


FIGURE 31: UC8 BUILDING AUTOMATION

The building is equipped with a data gateway product that connects to the local automation system and converts their native field bus protocols into SensorThings data model. Typically the protocols are BACnet or Modbus on building automation systems, KNX or Dali on electrical automation and M-Bus for other meters such as water or energy meters.

The incoming data is a stream of events indicating measurements in a POST request to a specific URL which reaches the platform first at the API Gateway

The API Gateway serves many purposes some of which are

- having a common endpoint for multiple backend services of the platform
- authorizing the access to the platform using API Keys
- setting a rate limit of requests to the backend HTTPS end-point

The HTTPS end-point receives the measurement data as a POST request in OCPP protocol format to the corresponding URL from the API gateway. The POST data is parsed and send as SensorThings API compliant format and fed(produced) into Kafka.

A Kafka streams application or a KSQL application obtains aggregates/summary from the generic stream and creates a new aggregate stream back into Kafka. The necessary context information if required in the processing should be possible to be obtained in the stream processor application using a GET request to the relevant URL of the asset data store for the stream in question.

The required streams (the generic one, the transformed one (of aggregates/summary), the stream of anomalies) are sinked (via Kafka Connectors) into the PostgreSQL (PostGIS) database as a store of historical data.

An outgoing endpoint that processes GET requests to obtain measurements in a way that is compliant with the oBIX API shall be served by another backend service that would query the PostGIS database.

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The measurements info like summaries etc. can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

UC9: DYNAMIC ATTRIBUTES IN 3D CITY MODEL

| DOMAIN | BUILT ENVIRONMENT |
|---|---|
| DESCRIPTION | Update attribute values on CityGML model with latest observations |
| MOTIVATION | Create a proof-of-concepts that it is possible to maintain live data values on CityGML model without ADE extensions |

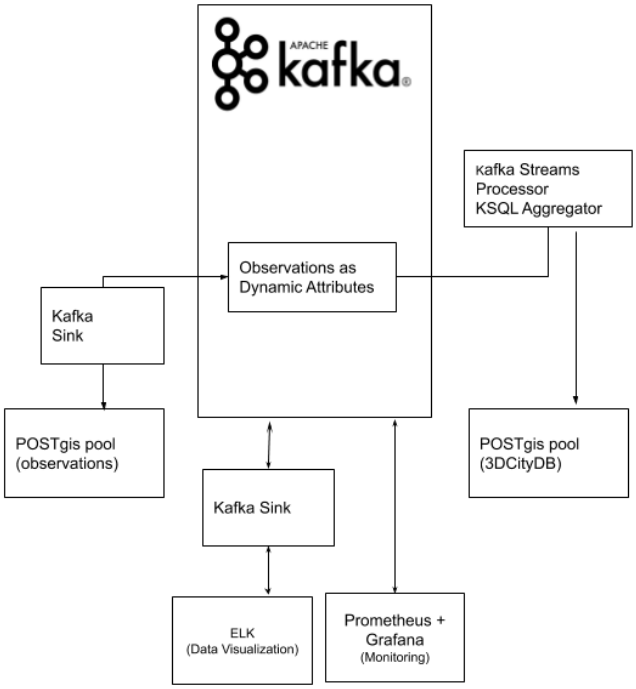


FIGURE 32: UC9 DYNAMIC ATTRIBUTES IN 3D CITY MODEL

A Kafka streams application is notified that an observation related to an attribute of an abstract feature in CityGML model is updated.

The required observation stream is queried from the observation database in the PostGIS cluster. The stream processor runs a database update query that updates the related value of the feature in the 3DCityDB database, that maintains the relational encoding of the CityGML model.

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The measurements info like summaries etc. can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

## UC10: NATURAL LANGUAGE PROCESSING IN OPEN311 HELPDESK

| DOMAIN | COMMON |
|---|---|
| DESCRIPTION | Analyze and prioritize important messages based on their content |
| MOTIVATION | Showcase the role of Natural Language Processing as an example of machine learning supporting the business processes |

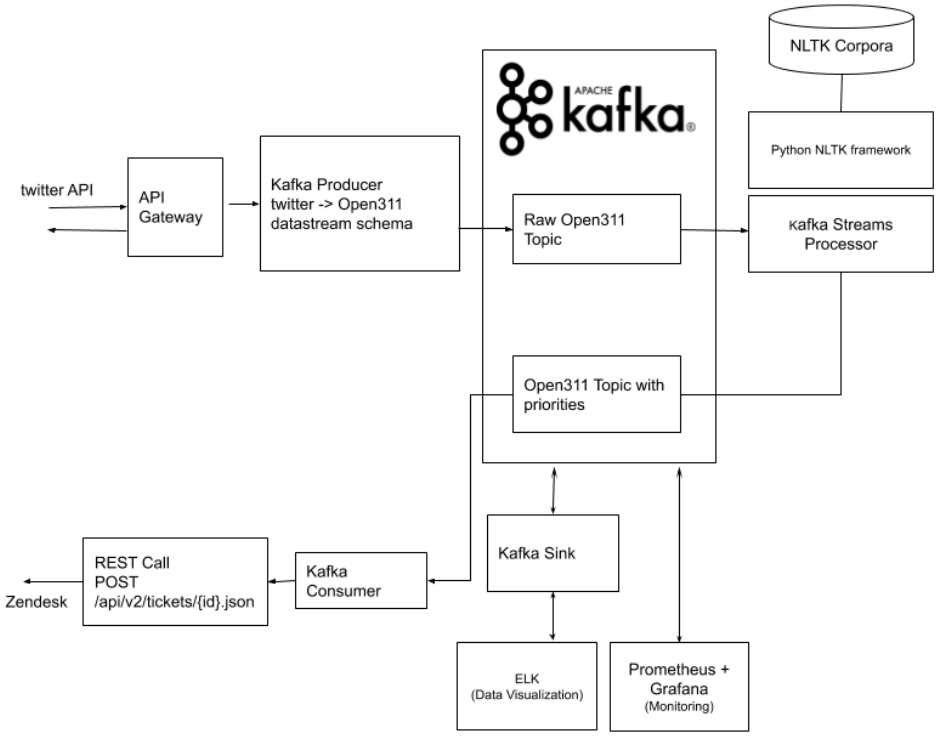


FIGURE 33: UC10 NATURAL LANGUAGE PROCESSING

The incoming data is a stream of tweets requested from twitter API which reaches the platform first at the API Gateway

The API Gateway serves many purposes some of which are

- having a common endpoint for multiple backend services of the platform
- authorizing the access to the platform using API Keys
- setting a rate limit of requests to the backend HTTPS end-point
- manage access tokens of third-party service (twitter API)

The HTTPS end-point receives the measurement data as a POST request in twitter protocol format to the corresponding URL from the API gateway. The POST data is parsed and converted into Open311 API compliant format and fed(produced) into Kafka.

A Kafka streams application supported by the Python Natural Language Toolkit (NLTK) obtains the raw data topic in Open311 schema.

At first, the natural language processing methods will be tested using twitter data feed. Preprocessing, feature extraction as well as many other steps will be performed using the NLTK, but also other Python packages such as Scikit-learn and Tensorflow will be used when needed. A wide collection of tools for data processing are included in NLTK.

Later, the focus of the study will be moved on analysis of other citizen feedback channels the cities or CoE has: as an example, emails or web form results. Similar preprocessing and feature extraction methods will be used as with the twitter data. However, the language used in the customer feedback forms can be identified based on the website where the form is, but in twitter the messages can be of any language. Sentiment analysis of the feedback forms will be one of the main topics of the Use Case.

Text mining and sentiment classification have been well researched in the past two decades. For more information about the background and methods, a study by Hu and Liu on mining and summarizing customer reviews is a good starter.[6]

The analytical tool adds a priority attribute to the Open311 message and creates a new stream back into Kafka. The necessary context information if required in the processing should be possible to be obtained in the stream processor application using a GET request to the relevant URL of the asset data store for the stream in question.

The required streams (the generic one and the transformed one (with priorities) are sinked (via Kafka Connectors) into the PostgreSQL (PostGIS) database as a store of historical data.

An outgoing endpoint that processes GET requests to obtain events in a way that is compliant with the Open311 API shall be served by another backend service that would query the PostGIS database.

The required streams are also sinked via Kafka Connector to the Elastic Search and visualized with Kibana. The measurements info like summaries etc. can be viewed here.

Prometheus and Grafana shall be used to monitor the health of the Kafka streaming platform.

# APPENDIX C. LIST OF REFERENCES

[1] L. Heuser, J. Scheer, P. den Hamer, and B. de Deathouwer, “Reference Architecture & Design Principles,” 2017.

[2] O. Lassila, “Setting Your Data Free: Thoughts on Information Interoperability,” no. June, pp. 1–24, 2007.

[3] J. Ruohomäki, Timo; Airaksinen, Enni; Huuska, Petteri; Kesäniemi, Outi; Martikka, Mikko; Suomisto, “Smart City Platform Enabling Digital Twin,” 2018.

[4] G. Agugiaro, J. Benner, P. Cipriano, and R. Nouvel, “The Energy Application Domain Extension for CityGML: enhancing interoperability for urban energy simulations,” *Open Geospatial Data, Softw. Stand.*, vol. 3, no. 1, p. 2, Dec. 2018.

[5] P. Nummi, “Sähköinen osallistuminen alueidenkäytön suunnittelussa,” *Ympäristöministeriön Julk.*, p. 16, 2018.

[6] M. Hu and B. Liu, “Mining and Summarizing Customer Reviews,” 2004.

# APPENDIX D. DOCUMENT METADATA

| Version | Date | Editors | Description |
|---|---|---|---|
| **0.1** | 14 May 2020 | Timo Ruohomäki | First draft, document structure and appendixes |
| **0.2** | 19 May 2020 | Noora Reittu | First draft of chapter four on geographical systems |
| **0.3** | 20 May 2020 | Andreas Andra | Chapter three on data models |
| **0.4** | 20 May 2020 | Sheena Puthanpurayil | Chapters on stream processing and operations |
| **0.5** | 21 May 2020 | Andreas Andra | Updates on Knowledge Management |
| **0.6** | 25 May 2020 | Timo Ruohomäki | FinEst Twins UoP Platform Architecture |
| **0.7** | 26 May 2020 | Timo Ruohomäki | Updates after review by Kalle Toiskallio (Aalto) |
| **0.8** | 26 May 2020 | Sheena Puthanpurayil, Kimmo Raivio | Use case definitions and data flow diagrams |
| **0.9** | 29 May 2020 | Henry Patzig | Final review by TalTech |
| **1.0** | 29 May 2020 | Timo Ruohomäki | Final version, ready for submission |

TABLE 2: DOCUMENT REVISION HISTORY

## STATEMENT OF ORIGINALITY

This deliverable document contains original, previously unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.