SCOREwater

Smart City Observatories implement REsilient Water Management

DELIVERABLE D3.1

FUNCTIONAL AND TECHNICAL ANALYSIS OF EXISTING SYSTEMS AND APPLICATIONS WITH DESCRIPTION OF STANDARDS, CONNECTIONS AND DATA

Dissemination level	Public
Type	Report
Issued by	Civity
Contributing project partners	Eurecat, Future City Foundation, Hydrologic
Author(s)	Hof, A., Vanmeulebrouk, B.
Reviewed by	Corominas, L., de Bruin, B., Hallgren, F.
Keywords	platform, architecture, software components, FIWARE, standards, data models
Number of pages	56
Number of annexes	1
Date:	2020-05-26
Version:	V 1
Deliverable number	D3.1
Work Package number:	WP 3
Status:	Delivered
Approved by coordinator (IVL)	2020-05-27

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Copyright notices

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

REVISION HISTORY

Version	Reason for changes	Name	Date
1	Original release to EU	Arjen Hof	2020-05-29

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

CONTENT

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

LIST OF FIGURES

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

LIST OF TABLES

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

ABBREVIATIONS

Abbreviation	Definition
CKAN	Comprehensive Kerbal Archive Network
GE	Generic Enabler
ICT	Information and Communications Technology
IoT	Internet of Things
SDG	Sustainable Development Goals
SME	Small and Medium-sized Enterprise

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

PROJECT ABSTRACT

SCOREwater focuses on enhancing the resilience of cities against climate change and urbanization by enabling a water smart society that fulfils SDGs 3, 6, 11, 12 and 13 and secures future ecosystem services. We introduce digital services to improve management of wastewater, stormwater and flooding events. These services are provided by an adaptive digital platform, developed and verified by relevant stakeholders (communities, municipalities, businesses, and civil society) in iterative collaboration with developers, thus tailoring to stakeholders' needs. Existing technical platforms and services (e.g. FIWARE, CKAN) are extended to the water domain by integrating relevant standards, ontologies and vocabularies, and provide an interoperable open-source platform for smart water management. Emerging digital technologies such as IoT, Artificial Intelligence, and Big Data is used to provide accurate real-time predictions and refined information.

We implement three large-scale, cross-cutting innovation demonstrators and enable transfer and upscale by providing harmonized data and services. We initiate a new domain "sewage sociology" mining biomarkers of community-wide lifestyle habits from sewage. We develop new water monitoring techniques and data-adaptive storm water treatment and apply to water resource protection and legal compliance for construction projects. We enhance resilience against flooding by sensing and hydrological modelling coupled to urban water engineering. We will identify best practices for developing and using the digital services, thus addressing water stakeholders beyond the project partners. The project will also develop technologies to increase public engagement in water management.

Moreover, SCOREwater will deliver an innovation ecosystem driven by the financial savings in both maintenance and operation of water systems that are offered using the SCOREwater digital services, providing new business opportunities for water and ICT SMEs.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

EXECUTIVE SUMMARY

The goal for developing and implementing the SCOREwater platform, is that it needs to be based on existing open source software components, standards and data models. A prerequisite is to use FIWARE where possible and applicable. This document evaluates and selects the FIWARE-components and possible alternatives for the implementation of the SCOREwater platform.

Initiatives in other EU-projects are investigated. Important for SCOREwater is the taskforce of similar EU-projects focusing on FIWARE-models for water management. SCOREwater is involved and collaborating in this taskforce, which is important to create widely supported solutions and a reference implementation of the SCOREwater platform.

The SCOREwater platform needs to facilitate the achievement of business goals, as described in the demonstration projects for each city. This document takes the different user stories for the SCOREwater-demonstration projects to select the appropriate datamodels, standards and software needed to implement the SCOREwater Platform.

Chapter 2 describes the user stories for the three demonstration projects as derived from the requirements each city has mentioned.

Chapter 3 investigates applicable standards and data models to support the user stories for each demonstration project. These models are used to describe the (sensor-)data in a standardized way. First, there has been investigated if a FIWARE-model is available and usable. If no standardized data models exist, collaboration is sought with other water related initiatives and standardisation bodies to investigate alternative solutions.

Chapter 4 describes the software alternatives for implementing the user stories in accordance with the defined data models and standards. The first option for open source software components are FIWARE Generic Enablers. They have been qualified based on the ISO-model for software quality. For some functions/components within the SCOREwater platform alternative open source solutions have been selected. The conclusions summarise the selection of components, standards and models that are proposed for the next step of the implementation of the SCOREwater Platform.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

1. INTRODUCTION

Purpose of this deliverable is to select data models, open standards and software used to implement the SCOREwater platform. What data models, standards and software are needed for the platform depends on the tasks the users of the platforms want to accomplish using the platform. These are described in the form of user stories in chapter 2. Chapter 3 investigates applicable data models and standards for the different user stories from the 3 cities. These models and standards need to be implemented in the SCOREwater platform. To be able to compare the different candidates using relevant characteristics, a simple software quality model is used. Chapter 4 focuses software and maps the SCOREwater software architecture on FIWARE software architecture. The initial candidate for a component is the FIWARE Generic Enabler (GE). If these GE´s do not pass the software quality analysis, alternatives are described and selected.

1.1. RELATION TO OTHER H2020 PROJECTS

Within the ICT4Water-cluster (ICT4water, sd) there are several projects with similar goals and setup as SCOREwater. The ICT4Water site is a hub for EU-funded research projects on ICT and Water Management, with a strong focus on interoperability between water information systems.

There are 5 EU-projects, with a focus on water, interested in cooperation on standard data models: Fiware4Water, DigitalWater City, SCOREwater, NAIADES and aqua3S. A taskforce is created that focuses on FIWARE and ontologies. The overall objective is to end-up with interoperable water models, covering the water lifecycle, deployed over NGSI-LD (FIWARE). SCOREwater is participating in this taskforce.

The generic principles underlying the SCOREwater platform are also implemented in EU-Lighthouse projects for energy transition. The European Innovation Partnership on Smart Cities and Communities (EIP-SCC) (EIP-SCC, sd) is bringing together cities, industries, SMEs, investors, researchers and other smart city actors. Their marketplace offers an overview of other projects and valuable guides, toolkits and blueprints. Relevant information on data models and API-standards is still limited.

2. SCOREWATER PLATFORM USER STORIES

User stories describe a software feature from an end-user perspective. They should describe who can do what with a certain feature and – most importantly – why. A user story should describe the added value of a feature for the end user. User stories may be refined and/or split up in multiple smaller user stories. Prioritization of user stories will take place after stakeholder consultation, where the stakeholders are supposed to represent the end users.

The user stories for the SCOREwater platform have been defined using requirements from the three different cases (Barcelona, Gothenburg and Amersfoort). The user stories for the different cities all share some common denominators. All cases want to be able to:

- Upload data from various sources to the SCOREwater platform
- Run data driven models on the SCOREwater platform
- Provide 3rd parties with access to data and models on the SCOREwater platform using different methods (data market)

2.1. UPLOAD DATA TO THE SCOREWATER PLATFORM

2.1.1. PUSH SENSOR DATA TO THE SCOREWATER PLATFORM

As a sensor data provider

I want to be able to push my data to the SCOREwater platform

So that I can publish them to a larger audience

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Acceptance criteria

- Minimum interval of data
- Maximum number of datapoints per minute
- Maximum number of records in one timeseries
- Maximum time to publication to customer

Background information

Sensors usually use some highly optimized data format to push the data they recorded to a data logger. To bring these data into the SCOREwater platform, they must be converted from the proprietary sensor format to a standardized format, a FIWARE format in this case. This is usually done using a so called IoT agent or Translator.

2.1.2. HARVEST 3RD PARTY API'S

Actual data

As a data provider

I want to be able to publish my data in the SCOREwater platform

So that I can publish them to a larger audience

Acceptance criteria

- Data is copied from my API to the SCOREwater platformMetadata is published on the SCOREwater platform
- Data is harmonized
- Automatic, periodic harvester
- Shortest interval available should be configurable

Background information

It is best practice to maintain data as close to the source as possible, but that is not possible in all situations, for example:

- Original platform cannot comply with sufficient quality of service requirements
- Harmonization/integration of data requires more processing than feasible in real time

Metadata only

As a data provider

I want to be able to publish my metadata in the SCOREwater platform

So that I can publish them to a larger audience

Acceptance criteria

- Metadata is copied from my API to the SCOREwater platform
- Data is not copied to the SCOREwater platform
- End users can access harmonized data, i. e. translate proprietary data to harmonized data (in this case NGSI)
- Automatic, periodic harvester

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Background information

When it is possible to maintain the data at the source only, the SCOREwater platform should refer to the data at its original location, without copying it to the SCOREwater platform. In this case, only a metadata record is kept at the SCOREwater platform.

2.1.3. MANUAL UPLOAD OF DATA

As a data provider

I want to be able to manually upload data (files) to the SCOREwater platform

So that I can publish them to a larger audience

Acceptance criteria

File formats to be supported:

CSV, XLS, SHP, GeoPackage, MDB, doc(x) , PDF, …

2.1.4. MANAGE ACCESS TO DATASETS

As a data provider

I want to be able to manage access restrictions to the datasets I publish on the SCOREwater platform

So that I can control who has access to my data sets and can prevent unauthorized access

Acceptance criteria

Rate limiting, plans, authentication and authorization

- Open data sets
- Data sets which require authentication and authorization
- Data sets for which the first no. of calls is free, after a certain amount of calls a subscription fee is required
- Data sets for which a fee must be paid before access is granted

2.1.5. LOGGING

As a data provider

I want to receive statistics on the usage of my datasets

So that I can better tailor my data sets to the needs of the users

Acceptance criteria

- Time of usage
- Location of usage (indication)
- User details
- Query parameters

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

2.2. RUN DATA DRIVEN MODELS ON THE SCOREWATER PLATFORM

Within the frame of SCOREwater data driven models will be developed. These can either be run on the platform (this paragraph) or outside of the platform. In that case, the resulting data can be uploaded to the platform using the functionality described by the user stories in paragraph 2.1. Figure 1 depicts the different options for running data driven models on the SCOREwater platform using the Barcelona case with the S::CAN sensors as an example.

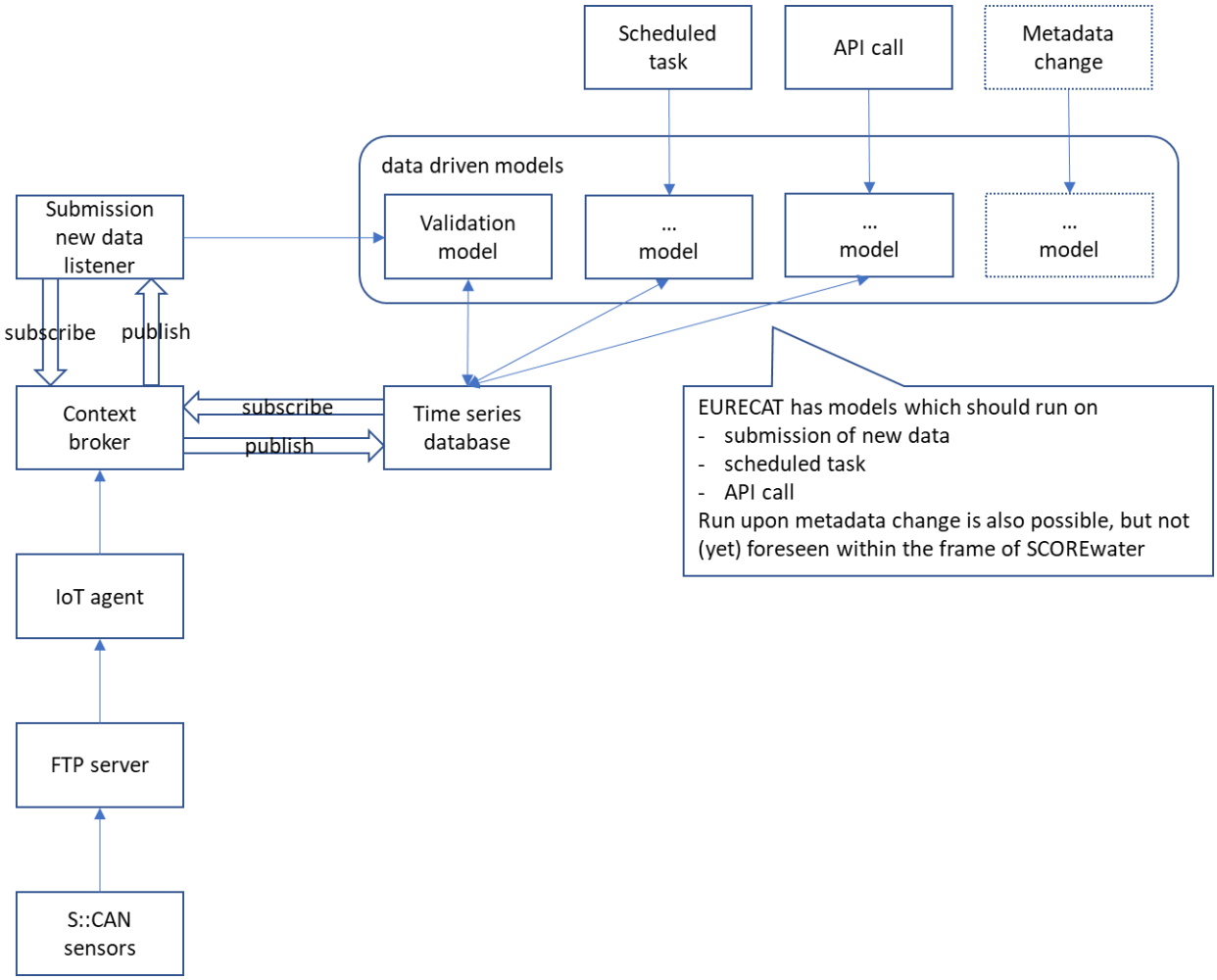


Figure 1 Options for running data driven models on the SCOREwater platform

The information from the S::CAN sensors is sent to the SCOREwater platform using FTP. The data is picked up by an IoT agent which translates the data to NGSI and submits it to a context broker. Multiple listeners are connected to this context broker: a listener to create a time series database and a listener to start a model which validates the incoming data (2.2.1). Other options for running data driven models are via a scheduled task, to periodically run a model (2.2.2) and to run a model upon API call (2.2.3). It is also possible to run a model upon metadata change, (2.2.4), but this option is not foreseen for SCOREwater yet.

2.2.1. UPON SUBMISSION OF NEW DATA

As a data scientist

I want to be able to run a model when new data is published on the platform

So that I receive new model results on the latest data input

Acceptance criteria

- Real time

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

- Subscribe to context broker
- Python
- Maximum runtime, maximum resources (CPU, RAM, Storage)
- Model logging is stored and available with authorization
- Location where progress of a model run can be found (scheduled, in progress, done)
- Model output is stored and available with authorization
- Meta-data is automatically updated

Background information

Eurecat is creating a model to sensor validate data as it comes in. This model takes the new measurement and compares it to data recorded earlier in the timeseries database.

2.2.2. RUN PERIODICALLY USING A SCHEDULED TASK

As a data scientist

I want to be able to periodically run a model using the data in the timeseries database

So that I get periodically updated model results

Acceptance criteria

- Configurable schedule
- Python
- Maximum runtime, maximum resources (CPU, RAM, Storage)
- Model logging is stored and available with authorization
- Location where progress of a model run can be found (scheduled, in progress, done)
- Model output is stored and available with authorization
- Meta-data is automatically updated

Background information

Eurecat, BCASA

2.2.3. VIA AN API CALL

As a software developer for a sewer maintenance company

I want to be start a model run model which uses the data in the timeseries database

So that I can create dashboards using the information the model produces

Acceptance criteria

- Synchronous or asynchronous?
- Python
- API endpoint to start model and set model parameters for model run
- Maximum runtime, maximum resources (CPU, RAM, Storage)
- Model logging is stored and available with authorization
- Location where progress of a model run can be found (scheduled, in progress, done)
- Model output is stored and available with authorization

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

- Meta-data is automatically updated

Background information

Eurecat, BCASA, predictive maintenance

2.2.4. UPON METADATA MODIFICATION

As a data scientist

I want to be able to run a model when metadata of a dataset is modified on the platform

So that I receive new model results on the changed dataset metadata

Acceptance criteria

- Real time
- Subscribe to context broker
- Python
- Maximum runtime, maximum resources (CPU, RAM, Storage)
- Model logging is stored and available with authorization
- Location where progress of a model run can be found (scheduled, in progress, done)
- Model output is stored and available with authorization
- Meta-data is automatically updated

Background information

Eurecat is creating a model to sensor validate data as it comes in. This model takes the new measurement and compares it to data recorded earlier in the timeseries database.

2.3. DATA MARKET

The Data Market offers the one stop shop for "data consumers" to available data from different data owners/providers. Data consumers can search for data, subscribe to data, test and learn about data (and their API´s). Although the Data Market focuses on the users of data, this can only be achieved if prerequisites for data owners/providers and the platform owners are met.

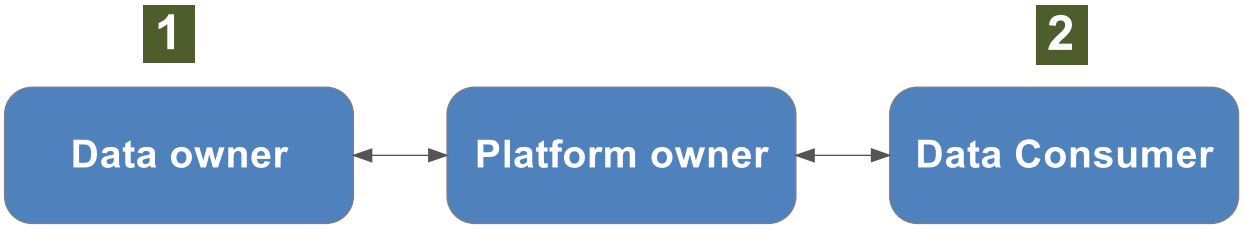


Figure 2 Different stakeholders/roles for a Data market

2.3.1. PUBLISH DATA

As a Data Owner

I want to publish data on the SCOREwater platform and decide how and to whom I make this data available

So that I a) can increase the exposure of my work and b) control who has access to my work

Acceptance criteria

- a data owner must be able to determine/set usage rights and licences

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

- a data owner must be able to add different price models to my data
    - Based on API-usage
    - Based on one time download
- the platform must insight in usage of my data
    - API-calls
    - Downloads

2.3.2. CONSUME DATA (1) – FIND DATA

As a data consumer

I want an overview of available data and the conditions on how to use them

So that I can decide if and how to use the data

Acceptance criteria

- the business case of each API, with suggestions about the type of problems it can solve

2.3.3. CONSUME DATA (2) - SUBSCRIBE TO DATASETS

As a data consumer

I want to be able to access datasets

So that I can use them in my work

Acceptance criteria

- API to use data sets in programs
- Download data sets

2.3.4. CONSUME DATA (3) – INSIGHT IN USAGE OF DATASETS

As a Data Consumer

I want insight in my usage of a dataset

So that I can determine if the subscription I took out is in line with my usage of the dataset

Acceptance criteria

- API-calls
- Downloads

2.3.5. CONSUME DATA (4) – SHOW CASE

As a data consumer

I want to promote my solution I created with the data/API and expose it in the Data Market (show case)

So that I can use it for reference purposes

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

2.3.6. CONSUME DATA (5) - SUBSCRIBE TO ALERTS

With the model results and measurements coming into the SCOREwater platform, it can function as an early warning system in an operational setting. Therefore, push messages are desired to send out alerts to specific events.

Push messages (SMS)

As a maintenance engineer/crisis response unit

I want to be notified when action is required

To prevent and resolve problems

Acceptance criteria

- Message send via SMS (should be country independent)
- Threshold exceedance triggers sending of SMS
- Content of message prescribed to take variables (known in portal) to change the message
- Threshold comparison interval set by admin
- Admin can configure Text to be send
- Admin can configure threshold comparison and interval of the comparison
- Admin can configure receivers of SMS notification
- SMS text may not exceed length of text message

Background information

Subscriber receives a warning if a certain configurable threshold is reached.

Push messages (email)

As a maintenance engineer/crisis response unit

I want to be notified when action is required

To prevent and resolve problems

Acceptance criteria

- Message send via Email
- Threshold exceedance triggers sending of message
- Content of message prescribed to take variables (known in portal) to change the message
- Threshold comparison interval set by admin
- Admin can configure Text to be send
- Admin can configure threshold comparison and interval of the comparison
- Admin can configure receivers of email notification
- Email must be signed with a DKIM key (to prevent it from being classified as junk email)

Background information

Subscriber receives a warning if a certain configurable threshold is reached.

Configuration of push messages

Configuration of content of push messages

As a crisis response manager

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

I want to configure what is send in a push message

So that I can tailor the content to the recipients

Acceptance criteria

- Message can use variables:
  - From model
  - From threshold
- Resulting total message length is shown
- Admin can choose the message type(s) to be used
- Thresholds can be used as variables in message

Configuration of trigger of push messages

As a crisis response manager

I want to configure when a push message is send

So that I can tailor the trigger to send a message to the recipients

Acceptance criteria

- Triggers can be set on thresholds in relation to model variables:
  - Exceedance
  - Falls below
- Thresholds can be set for model variables

Configuration of recipients of push messages

As a crisis response manager

I want to configure to who a message is send

So that I can reach the appropriate recipients of the message

Acceptance criteria

- Recipients can be CRUD (name, email address and phone number)
- For each trigger a list of recipients can be chosen

For each trigger the type of push message can be chosen (email/SMS/both)

2.3.7. PLATFORM OWNER

As a platform owner

I want to be able to engage/invite partners, developers, data providers to use the platform with its API-products, dataproducts and solutions

So that I can further increase to usage of the platform

Acceptance criteria

- current and future stakeholders must be able to learn/experience our API´s, data, solutions and our project
- the user experience must be elegant and simple, both for data provider and data consumers

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

- documentation must be top notch. Documentation that is as easily consumed by beginners as it is by experts
- help developers begin sampling and working with the API as quickly as possible ("time to hello API")
- establish trust in the API´s provided and offer confidence in the security and reliability of the API´s
- attractive to partners that help to create better end-to-end customer experiences

3. STANDARDS AND DATA MODELS FOR SCOREWATER USER STORIES

3.1. INTRODUCTION

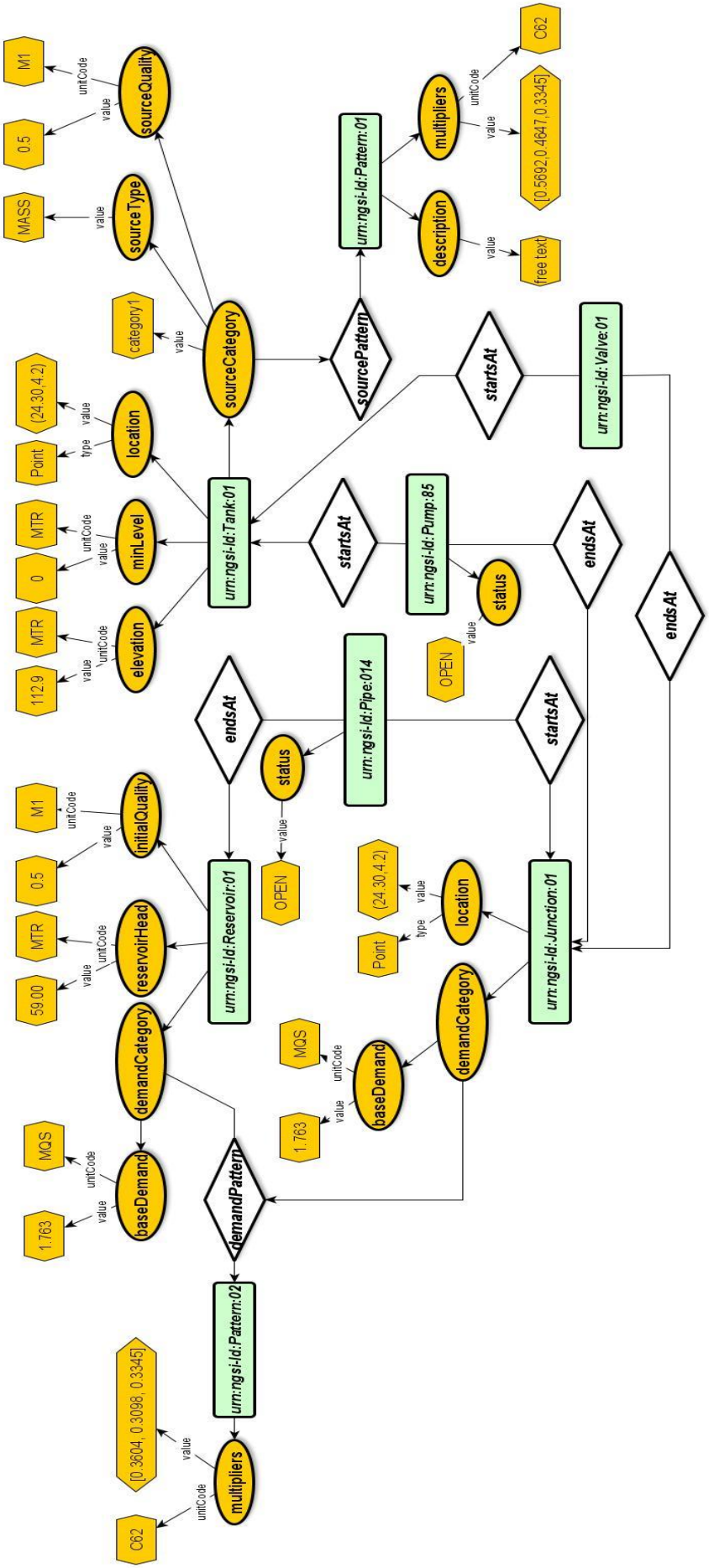
When it concerns "standards", there a lot of different entries to handle this topic. Within SCOREwater the focus is primarily on API-standards and standard data models. Following the description of the user stories, this chapter will look at common API´s and specific, standardized data models for the different demonstration projects in SCOREwater. Whenever possible, available FIWARE-models will be used. If there is no appropriate FIWARE data model available, open, standard models from outside the FIWARE ecosystem will be recommended. These models might be added to the FIWARE-list of data models in a later stadium.

Many standards for data models are already described in D1.1. The EU-projects mentioned in paragraph 1.1 of this deliverable investigate the implementation of FIWARE-standards and data models, while benefitting from existing initiatives such as ETSI SAREF (**S**mart **A**pplications **REF**erence Ontology).

3.2. STANDARDS AND MODELS FOR ALL CASES

Specific FIWARE data models for water are still in their early stages. There is a beginning of Water Network Management Harmonized Data Models, mainly derived from the EPANET platform. The main entities currently described in this model are: Curve, Junction, Pattern, Pipe, Pump, Reservoir, Tank and Valve. The picture below shows the first draft of this model. Some of these entities can be useful for SCOREwater cases.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751



This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Eurecat is involved in the SAREF4WATER. There is DRAFT version of this document on the ETSI-portal. A mapping between SAREF4WATER and NGSI LD is one of the tasks at hand. The next steps are to consolidate the SAREF4WATER/NGSI-LD mappings and to publish the final JSON-LD contexts.

3.2.1. FIWARE NGSI-V2 AND NGSI-LD

The FIWARE NGSI (Next Generation Service Interface) API is one of the standards to be implemented in SCOREwater. This API defines[1]:

- a **data model** for context information, based on a simple information model using the notion of context entities
- a **context data interface** for exchanging information by means of query, subscription, and update operations
- a **context availability interface** for exchanging information on how to obtain context information (whether to separate the two interfaces is currently under discussion).

The main elements in the NGSI data model are context entities, attributes and metadata, as shown in the figure below.

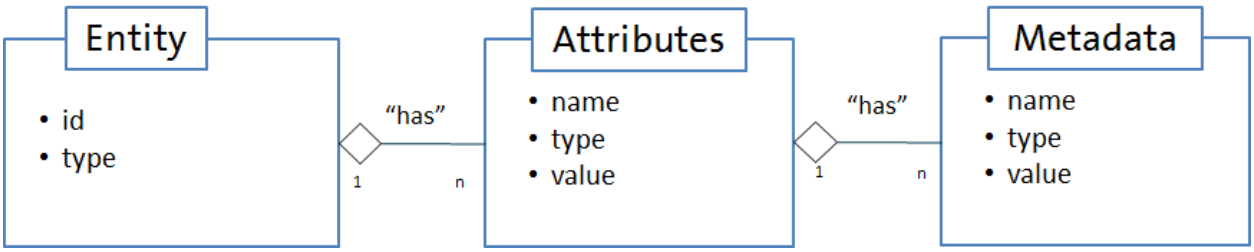


Figure 3 NGSI data model

There are two options: NGSI V2 and NGSI LD. The main differences between the two is that the underlying Data Model for NGSI-LD is based upon the Property Graph Data Model, as shown in Figure 4 Property Graph Data Model for NGSI-LD. Furthermore the Entity IDs in NGSI V2, are URIs (URLs or URNs) in NGSI-LD, some metadata is standardized and a few properties/attributes have changed. Overall both REST API´s are quite similar.

[1] https://fiware.github.io/specifications/ngsiv2/stable/

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

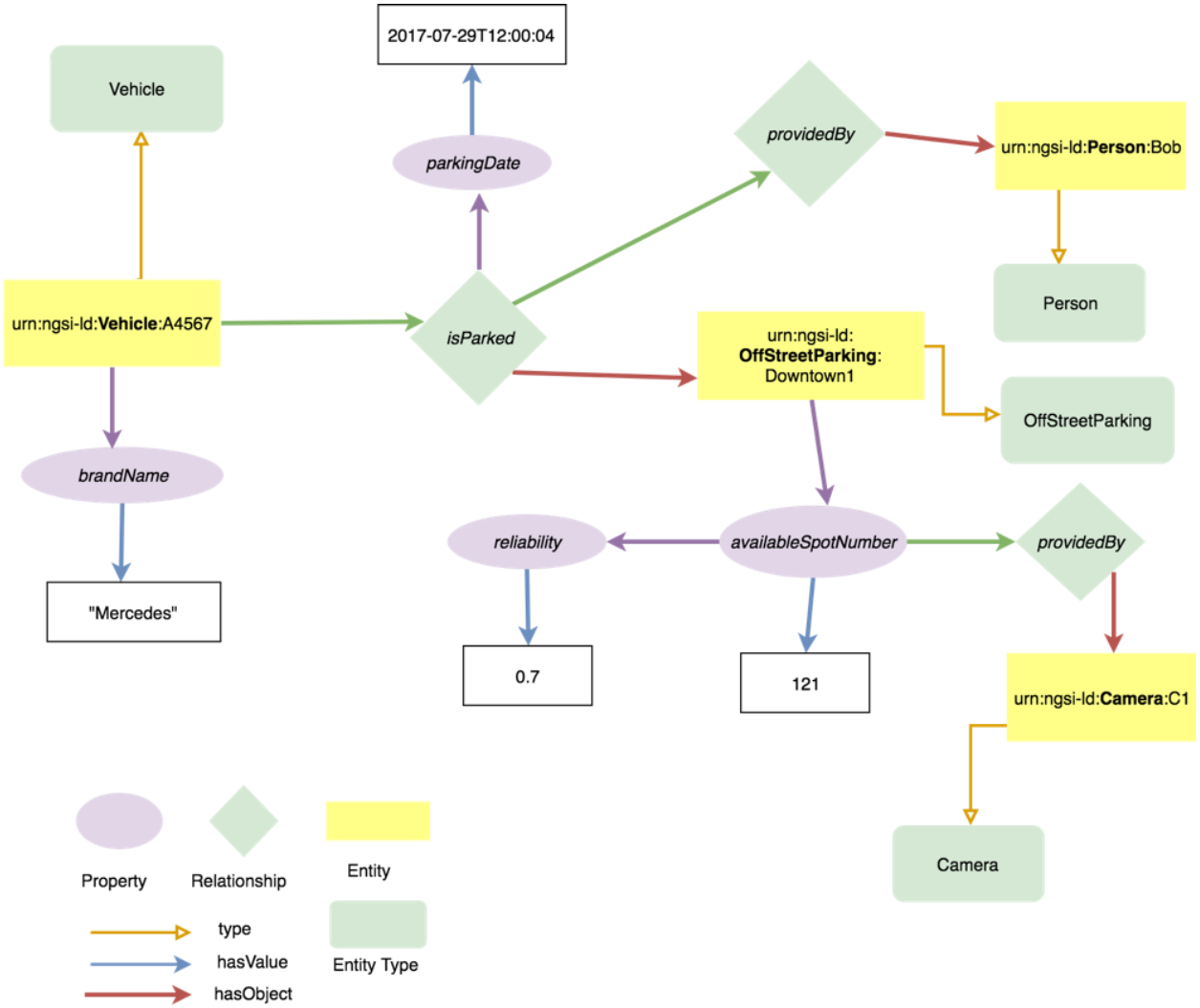


Figure 4 Property Graph Data Model for NGSI-LD

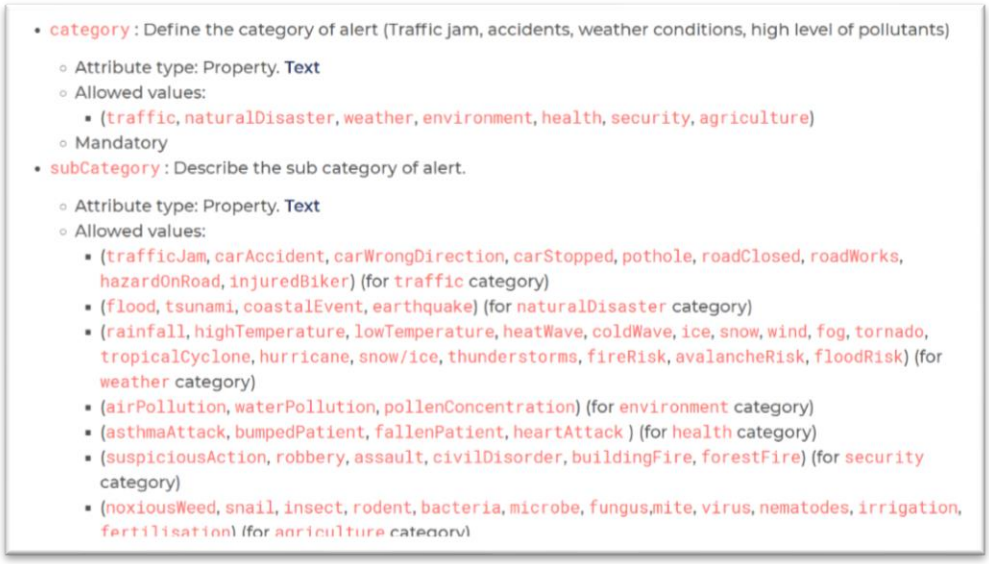
FIWARE has a catalogue with so called Generic Enablers (GE´s). These buildings blocks are software that can be used within the SCOREwater platform. These GE´s are described in paragraph 4.2.

3.2.2. FIWARE DATA MODELS FOR DEMONSTRATION PROJECTS

This paragraph will look at the available an applicable data model for the demonstration projects. Besides the project-specific needs, they share their need for an “alerting” model. Alerting is the option to notify a user or device of a specific condition/status or trigger a specific action, based on the alert. For alerting there is a FIWARE-model that will be used (FIWARE, 2020). This model has been developed to be used/implemented in the SmartSDK-project. The goal of this (expired) EU-project was to provide a number of ready to use models for the creation of Smart Services based on NGSI version 2. The results of this project have been incorporated in the existing FIWARE-models.

The JSON-schema for the “alert” FIWARE data model describes categories and subcategories that are applicable (see Figure 5). During the implementation of the SCOREwater demonstration projects it will become clear whether or not these (sub-) categories are sufficient. This alerting-model will therefore be applied for all use cases.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751



- category : Define the category of alert (Traffic jam, accidents, weather conditions, high level of pollutants)
  - Attribute type: Property. Text
  - Allowed values:
    - (traffic, naturalDisaster, weather, environment, health, security, agriculture)
  - Mandatory
- subCategory : Describe the sub category of alert.
  - Attribute type: Property. Text
  - Allowed values:
    - (trafficJam, carAccident, carWrongDirection, carStopped, pothole, roadClosed, roadWorks, hazardOnRoad, injuredBiker) (for traffic category)
    - (flood, tsunami, coastalEvent, earthquake) (for naturalDisaster category)
    - (rainfall, highTemperature, lowTemperature, heatWave, coldWave, ice, snow, wind, fog, tornado, tropicalCyclone, hurricane, snow/ice, thunderstorms, fireRisk, avalancheRisk, floodRisk) (for weather category)
    - (airPollution, waterPollution, pollenConcentration) (for environment category)
    - (asthmaAttack, bumpedPatient, fallenPatient, heartAttack) (for health category)
    - (suspiciousAction, robbery, assault, civilDisorder, buildingFire, forestFire) (for security category)
    - (noxiousWeed, snail, insect, rodent, bacteria, microbe, fungus,mite, virus, nematodes, irrigation, fertilisation) (for agriculture category)

Figure 5 FIWARE alerting data model[2]

3.2.3. DATA MARKET API´S AND STANDARDS

The Data Market is a combination of technology components, such as API-management and identity management and Open API´s for managing the provisioning of data.

FIWARE offers the Biz-ecosystem Generic Enabler, which is a joint component made up of the FIWARE Business Framework and a set of APIs (and its reference implementations) provided by the TMForum. “This component allows the monetization of different kind of assets during the whole service life cycle, from offering creation to its charging, accounting and revenue settlement and sharing. The Business API Ecosystem exposes its complete functionality through TMForum standard APIs; concretely, it includes the catalog management, ordering management, inventory management, usage management, billing, customer, and party APIs.”[3] The schema below shows the architecture for the FIWARE-TM Forum Biz Ecosystem

The Business API Ecosystem is not a single software repository, but it is composed of different projects which work together to provide the complete functionality. Within SCOREwater we will implement the necessary and appropriate components.

[2] https://fiware-datamodels.readthedocs.io/en/latest/Alert/doc/spec/index.html#alert-data-model
[3] https://github.com/FIWARE-TMForum/Business-API-Ecosystem

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

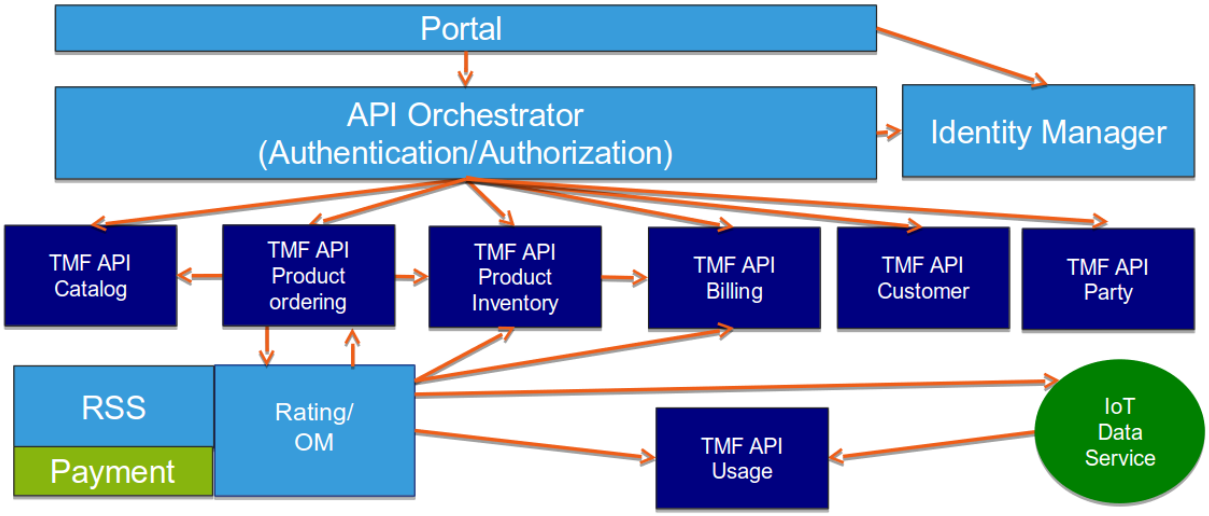


Figure 6 The FIWARE-TM Forum Business API Ecosystem

3.2.4. DATA MODELS FOR AMERSFOORT CASE

Within the frame of the Amersfoort case, the SCOREwater platform will have to be able to deal with data regarding air quality, weather and water.

Air quality

For air quality there is a FIWARE-model "Air quality observed" (FIWARE, 2020). Civity has extensive experience with this model from other projects, such as Snifferbike. This model will be used to harmonize the air quality data collected within the frame of the Amersfoort case.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

DATA MODEL

A JSON Schema corresponding to this data model can be found **here**.

- `id` : Unique Identifier.
- `type` : Entity type. It must be equal to `AirQualityObserved`.
- `dataProvider` : Specifies the URL to information about the provider of this information
  - Attribute type: Property. **URL**
  - Optional
- `dateModified` : Last update timestamp of this entity.
  - Attribute type: Property. **DateTime**
  - Read-Only. Automatically generated.
- `dateCreated` : Entity's creation timestamp.
  - Attribute type: Property. **DateTime**
  - Read-Only. Automatically generated.
- `location` : Location of the air quality observation represented by a GeoJSON geometry.
  - Attribute type: GeoProperty. `geo:json`.
  - Normative References: **https://tools.ietf.org/html/rfc7946**
  - Mandatory if `address` is not defined.
- `address` : Civic address of the air quality observation. Sometimes it corresponds to the air quality station address.
  - Normative References: **https://schema.org/address**
  - Mandatory if `location` is not present.
- `dateObserved` : The date and time of this observation in ISO8601 UTCformat. It can be represented by an specific time instant or by an ISO8601 interval.
  - Attribute type: Property. **DateTime** or an ISO8601 interval represented as **Text**.
  - Mandatory
- `areaServed` : Higher level area to which this air quality measurement belongs to. It can be used to group air quality measurements per district, neighbourhood, etc.
  - Attribute type: Property. **Text**
  - Normative References: **https://schema.org/areaServed**
  - Optional
- `source` : A sequence of characters giving the source of the entity data.
  - Attribute type: Property. **Text** or **URL**
  - Optional
- `airQualityLevel` : Overall qualitative level of health concern corresponding to the air quality observed.
  - Attribute type: Property. **Text**
  - Example values defined by the **USA EPA Agency**: (`good`, `moderate`, `unhealthyForSensitiveGroups`, `unhealthy`, `veryUnhealthy`, `hazardous`). As this can be different between countries, regulations or implementations, the set of allowed values will depend on the reference specification used. It is recommended that implementations use the same naming conventions as exemplified above (lower case starting words, camel case when compound terms are used)
  - Attribute metadata:
    - `referenceSpecification` : Specification that must be taken as reference when interpreting the supplied qualitative value.
      - Type: **Text** or **URL**
      - Mandatory
  - Optional
- `airQualityIndex` : Air quality index corresponding to the air quality observed.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `referenceSpecification` : Specification that must be taken as reference when interpreting or calculating the supplied air quality index.
      - Type: **Text** or **URL**
      - Optional
  - Optional
- `reliability` : Reliability (percentage, expressed in parts per one) corresponding to the air quality observed.
  - Attribute type: Property. **Number**
  - Allowed values: Interval [0,1]
  - Optional
- `refDevice` : A reference to the device(s) which captured this observation.
  - Attribute type: Relationship. Reference to an entity of type `Device`
  - Optional
- `refPointOfInterest` : A reference to a point of interest (usually an air quality station) associated to this observation.
  - Attribute type: Relationship. Reference to an entity of type `PointOfInterest`
  - Optional

REPRESENTING AIR POLLUTANTS

In order to enable a proper management of the concentrations of the different pollutants, *for each* pollutant (measurand) there must be an attribute which name *MUST* be exactly equal the chemical formula (or mnemonic) of the measurand, ex. CO. The structure of such an attribute will be as follows:

- Attribute name: Equal to the name of the measurand, for instance `CO`.
- Attribute type: Property. **Number**
- Attribute value: corresponds to the value for the measurand as a number.
- Attribute metadata:
  - `timestamp` : optional timestamp for the observed value in ISO8601 format. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
    - Type: **DateTime**
  - `unitCode` : The unit code (text) of measurement given using the **UN/CEFACT Common Code** (max. 3 characters). For instance, `GP` represents milligrams per cubic meter and `GQ` represents micrograms per cubic meter.
    - Type: **Text**
    - Mandatory
  - `description` : short description of the measurand
    - Attribute type: Property. **Text**
    - Normative References: `https://uri.etsi.org/ngsi-ld/description` equivalent to **description**
    - Optional

REPRESENTING QUALITATIVE LEVELS OF THE DIFFERENT AIR POLLUTANTS

In order to enable a proper management of the qualitative levels of the different pollutants, *for each* pollutant (measurand) there might be an attribute which name *MUST* be exactly equal to the concatenation of the chemical formula (or mnemonic) of the measurand with the string `_Level`, ex. `CO_Level`. To be more precise, the structure of such an attribute will be as follows:

- Attribute name: Equal to the name of the measurand plus the suffix `_Level`, for instance `CO_Level`.
- Attribute type: Property. **Text**
- Attribute value: Example values defined by the USA EPA Agency:(`good`, `moderate`, `unhealthyForSensitiveGroups`, `unhealthy`, `veryUnhealthy`, `hazardous`). As this can be different between countries, regulations or implementations, the set of allowed values will depend on the reference specification used. It is recommended that implementations use the same naming conventions as exemplified above (lower case starting words, camel case when compound terms are used)
- Attribute metadata:
  - `description` : short description of the measurand and its related qualitative level
    - Type: **Text**
    - Optional
  - `referenceSpecification` : Specification that must be taken as reference when interpreting the supplied qualitative value.
    - Type: **Text** or **URL**
    - Mandatory

REPRESENTING AIRQUALITY-RELATED WEATHER CONDITIONS

Certain weather conditions have an influence over the observed air quality. There are two options for representing them:

- A/ Through a linked entity of type `WeatherObserved` (attribute named `refWeatherObserved`).
- B/ Through a group of weather-related properties already defined by **WeatherObserved**.

Below is the description of the attribute to be used for option A/.

- `refWeatherObserved` : Weather observed associated to the air quality conditions described by this entity.
  - Attribute type: Relationship. Reference to a **WeatherObserved** entity.
  - Optional

Note: JSON Schemas are intended to capture the data type and associated constraints of the different Attributes, regardless their final representation format in NGSI(v2, LD).

Figure 7 FIWARE Air quality observed data model

Weather

One of the goals of the City of Amersfoort is to reduce (the feeling of) heat stress within the city. For this purpose, data regarding the actual and perceived weather will be collected, amongst others by the citizen science-project Meetjestad (measure your city) whose data will be made available through the SCOREwater platform. This initiative started in 2015 and citizens measure temperature and humidity in their own neighborhood to research the effects of climate change in Amersfoort. For collecting and harmonizing data on temperature and humidity the FIWARE datamodel “weather observed” (FIWARE, 2020) will be applied.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

DATA MODEL

A JSON Schema corresponding to this data model can be found **here**.

- `id` : Unique identifier.
- `type` : Entity type. It must be equal to `WeatherObserved`.
- `dataProvider` : Specifies the URL to information about the provider of this information
  - Attribute type: Property. **URL**
  - Optional
- `dateModified` : Last update timestamp of this entity.
  - Attribute type: Property. **DateTime**
  - Read-Only. Automatically generated.
- `dateCreated` : Entity's creation timestamp.
  - Attribute type: Property. **DateTime**
  - Read-Only. Automatically generated.
- `name` : Name given to the weather observed location.
  - Attribute type: Property. **Text**
  - Normative References: `https://uri.etsi.org/ngsi-ld/name` equivalent to **name**
  - Optional
- `location` : Location of the weather observation represented by a GeoJSON geometry.
  - Attribute type: Property. `geo:json`.
  - Normative References: **https://tools.ietf.org/html/rfc7946**
  - Mandatory if `address` is not defined.
- `address` : Civic address of the weather observation. Sometimes it corresponds to a weather station address.
  - Attribute type: Property. **Address**
  - Normative References: **https://schema.org/address**
  - Mandatory if `location` is not present.
- `dateObserved` : The date and time of this observation in ISO8601 UTCformat. It can be represented by an specific time instant or by an ISO8601 interval.
  - Attribute type: Property. **DateTime** or an ISO8601 interval represented as **Text**.
  - Mandatory
- `source` : A sequence of characters giving the source of the entity data.
  - Attribute type: Property. **Text** or **URL**
  - Optional
- `refDevice` : A reference to the device(s) which captured this observation.
  - Attribute type: Relationship. Reference to an entity of type `Device`
  - Optional
- `refPointOfInterest` : A reference to a point of interest (usually a weather station) associated to this observation.
  - Attribute type: Relationship. Reference to an entity of type `PointOfInterest`
  - Optional
- `weatherType` : The observed weather type. It is represented by a comma separated list of weather statuses, for instance `overcast, lightRain`.
  - Attribute type: Property. **Text**
  - Allowed values: A combination of (`clearNight`,`sunnyDay`, `slightlyCloudy`, `partlyCloudy`, `mist`, `fog`, `highClouds`, `cloudy`, `veryCloudy`, `overcast`, `lightRainShower`, `drizzle`, `lightRain`, `heavyRainShower`, `heavyRain`, `sleetShower`, `sleet`, `hailShower`, `hail`, `shower`, `lightSnow`, `snow`, `heavySnowShower`, `heavySnow`, `thunderShower`, `thunder`) or any other extended value.
  - Optional
- `dewPoint` : The dew point encoded as a number.
  - Attribute type: Property. **Number**
  - Default unit: Celsius degrees.
  - See also: **https://en.wikipedia.org/wiki/Dew_point**
  - Optional
- `visibility` : Visibility reported.
  - Attribute type: Property. **Text**
  - Allowed values: One of (`veryPoor`, `poor`, `moderate`, `good`, `veryGood`, `excellent`)
  - Optional
- `temperature` : Air's temperature observed.
  - Attribute type: Property. **Number**
  - Default unit: Degrees centigrades.
  - Attribute metadata:
    - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
      - Type: **DateTime**
  - Optional
- `relativeHumidity` : Air's relative humidity observed (percentage, expressed in parts per one).
  - Attribute type: Property. **Number**
  - Allowed values: A number between `0` and `1`.
  - Attribute metadata:
    - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
      - Type: **DateTime**
  - Optional
- `precipitation` : Precipitation level observed.
  - Attribute type: Property. **Number**
  - Default unit: Liters per square meter.
  - Attribute metadata:
    - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
      - Type: **DateTime**
  - Optional
- `windDirection` : The wind direction expressed in decimal degrees compared to geographic North (measured clockwise), encoded as a Number. Range 0 to 360.
  - Attribute type: Property. **Number**
  - Default unit: Decimal degrees
  - Attribute metadata:
    - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
      - Type: **DateTime**
  - Optional
- `windSpeed` : The observed wind speed in m/s, encoded as a Number.
  - Attribute type: Property. **Number**
  - Default unit: meters per second
  - Attribute metadata:
    - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
      - Type: **DateTime**
  - Optional
- `atmosphericPressure` : The atmospheric pressure observed measured in Hecto Pascals.
  - Attribute type: Property. **Number**
  - Default unit: Hecto Pascals
  - Attribute metadata:
    - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
      - Type: **DateTime**
  - Optional
- `pressureTendency` : Is the pressure rising or falling? It can be expressed in quantitative terms or qualitative terms.
  - Attribute type: Property. **Text** or **Number**
  - Allowed values, if expressed in quantitative terms: one Of (`raising`, `falling`, `steady`)
  - Optional
- `solarRadiation` : The solar radiation observed measured in Watts per square meter.
  - Attribute type: Property. **Number**
  - Default unit: Watts per square meter
  - Attribute metadata:
    - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
      - Type: **DateTime**
  - Optional
- `illuminance` : The **illumninance** observed measured in lux (lx) or lumens per square metre (cd·sr·m−2).
  - Attribute type: Property. **Number**
  - Default unit: Lux
  - Attribute metadata:
  - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
    - Type: **DateTime**
  - Optional
- `streamGauge`: The water level surface elevation observed by Hydrometric measurement sensors, namely a **Stream Gauge**, expressed in centimeters.
  - Attribute type: Property. **Number**
  - Default unit: Centimeters (cm)
  - Attribute metadata:
    - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
  - Optional
- `snowHeight`: The snow height observed by generic snow depth measurement sensors, expressed in centimeters.
  - Attribute type: Property. **Number**
  - Default unit: Centimeters (cm)
  - Attribute metadata:
    - `timestamp` : optional timestamp for the observed value. It can be omitted if the observation time is the same as the one captured by the `dateObserved` attribute at entity level.
  - Optional

Note: JSON Schemas are intended to capture the data type and associated constraints of the different Attributes, regardless their final representation format in NGSI(v2, LD).

Figure 8 FIWARE weather observed data model

Water

For water there are no FIWARE data models. We will use existing models and collaborate in the FIWARE taskgroup to develop appropriate models to be used in the Amersfoort-case.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

3.2.5. DATA MODELS FOR BARCELONA CASE

The Barcelona case deals with sewage system information.

Sewer systems

At the time of writing, there is no existing FIWARE data model for sewer systems. We have to look for existing other models that can be used within the SCOREwater platform. In the Netherlands, a foundation called Rioned has developed a dictionary for water management systems. This is an open standard, according to linked data principles. It´s recognized by Dutch government as a mandatory open standard. It looks like the most comprehensive model available for exchanging and provisioning of data, but the text is in Dutch. This standard is connected to other (inter)national standards. Whether this data model is applicable to the Barcelona use case is to be determined. As long as other data models, dictionaries and linked data models are lacking, this standard might be the right starting point.

3.2.6. DATA MODELS FOR GOTHENBURG CASE

The Gothenburg case deals with waste water quality.

Waste water (and Sewer systems)

For the waste water case in Gothenburg we suggest to use the FIWARE-model “water quality observed” (FIWARE, 2020). This Water Quality data model is intended to represent water quality parameters at a certain water mass section. The waste water that is measured by the sensors send their data to the SCOREwater platform. The common water quality parameters and the concentration of chemical agents are part of the model. For the Barcelona-case we will use the same model.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

DATA MODEL

A JSON Schema corresponding to this data model can be found **here**.

- `id` : Unique identifier.
- `type` : Entity type. It must be equal to `WaterQualityObserved`.
- `dataProvider` : Specifies the URL to information about the provider of this information
  - Attribute type: Property. **URL**
  - Optional
- `dateModified` : Last update timestamp of this entity.
  - Attribute type: Property. **DateTime**
  - Read-Only. Automatically generated.
- `dateCreated` : Entity's creation timestamp.
  - Attribute type: Property. **DateTime**
  - Read-Only. Automatically generated.
- `location` : Location where measurements have been taken, represented by a GeoJSON Point.
  - Attribute type: GeoProperty. `geo:json`.
  - Normative References: **https://tools.ietf.org/html/draft-ietf-geojson-03**
  - Mandatory if `address` is not present.
- `address`: Civic address where the Water Quality measurement is taken.
  - Attribute type: Property. **Address**
  - Normative References: **https://schema.org/address**
  - Mandatory if `location` is not present.
- `refPointOfInterest` : A reference to a point of interest associated to this observation.
  - Attribute type: Property. Reference to an entity of type `PointOfInterest`
  - Optional
- `dateObserved` : The date and time of this observation in ISO8601 UTCformat. It can be represented by an specific time instant or by an ISO8601 interval.
  - Attribute type: Property. **DateTime** or an ISO8601 interval represented as **Text**.
  - Mandatory
- `source` : A sequence of characters giving the source of the entity data.
  - Attribute type: Property. **Text** or **URL**
  - Optional

CONCENTRATIONS OF CHEMICAL AGENTS

This data model is flexible enough to accommodate different chemical agents present in water and which can be measured. Applications MUST declare the list of chemical agents which concentration is being measured. The `measurand` attribute must be used for such purpose.

- `measurand` : An array of strings containing details (see format below) about extra measurands provided by this observation.
  - Attribute type: Property. List of **Text**.
  - Allowed values: Each element of the array must be a string with the following format (comma separated list of values): `<measurand>, <observedValue>, <unitcode>, <description>`, where:
    - `measurand` : corresponds to the chemical formula (or mnemonic) of the measurand, ex. CO.
    - `observedValue` : corresponds to the value for the measurand as a number.
    - `unitCode` : The unit code (text) of measurement given using the **UN/CEFACT Common Code** (max. 3 characters). For instance, `M1` represents milligrams per liter.
    - `description` : short description of the measurand.
    - Examples: `"NO3,0.01, M1, Nitrates"`
  - Optional

Below there is a list of typical chemical agents measured when analysing water quality. If such chemical agents are measured data providers MUST use the property names expressed by the following list. Nonetheless, the `measurand` attribute MUST be used to declare them, so that applications can discover what extra measurands are available as part of an observation.

- `O2` : Level of free, non-compound oxygen present.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: milligrams per liter (mg/L).
  - Optional
- `Chla` : Concentration of chlorophyll A.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: micrograms per liter.
  - Optional
- `PE` : Concentration of pigment phycoerythrin which can be measured to estimate cyanobacteria concentrations specifically.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: micrograms per liter.
  - Optional
- `PC` : Concentration of pigment phycocyanin which can be measured to estimate cyanobacteria concentrations specifically.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: micrograms per liter.
  - Optional
- `NH4` : Concentration of ammonium.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: milligrams per liter (mg/L).
  - Optional
- `NH3` : Concentration of ammonia.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: milligrams per liter (mg/L).
  - Optional
- `Cl-` : Concentration of chlorides.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: milligrams per liter (mg/L).
  - Optional
- `NO3` : Concentration of nitrates.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: milligrams per liter (mg/L).
  - Optional

Note: JSON Schemas are intended to capture the data type and associated constraints of the different Attributes, regardless their final representation format in NGSI(v2, LD).

COMMON WATER QUALITY PARAMETERS

- `temperature` : Temperature.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: Celsius Degrees.
  - Optional
- `conductivity` : Electrical Conductivity.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: Siemens per meter (S/m).
  - Optional
- `conductance` : Specific Conductance.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: Siemens per meter at 25 °C (S/m).
  - Optional
- `tss` : Total suspended solids.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: milligrams per liter (mg/L).
  - Optional
- `tds` : Total dissolved solids.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: milligrams per liter (mg/L).
  - Optional
- `turbidity` : Amount of light scattered by particles in the water column.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: Formazin Turbidity Unit (FTU).
  - Optional
- `salinity` : Amount of salts dissolved in water.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: Parts per thousand (ppt).
  - Optional
- `pH` : Acidity or basicity of an aqueous solution.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: Negative of the logarithm to base 10 of the activity of the hydrogen ion.
  - Optional
- `orp` : Oxidation-Reduction potential.
  - Attribute type: Property. **Number**
  - Attribute metadata:
    - `timestamp`: Timestamp when the last update of the attribute happened.
      - Type: **DateTime**
  - Default unit: millivolts (mV).
  - Optional

Figure 9 FIWARE water quality observed data model

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

3.3. ALTERNATIVE SOLUTIONS: SENSORTHINGS API

Another standard in the OGC Sensorthings API (Open Geospatial Consortium, 2020). This API provides an open, geospatial-enabled way to interconnect the Internet of Things (IoT) devices, data, and applications over the Web (Open Geospatial Consortium, 2020). OGC SensorThings API is an official OGC standard specification. It was approved by OGC Technical Committee in February 2016. The underlying data model of the OGC Sensorthings API is shown in Figure 10.

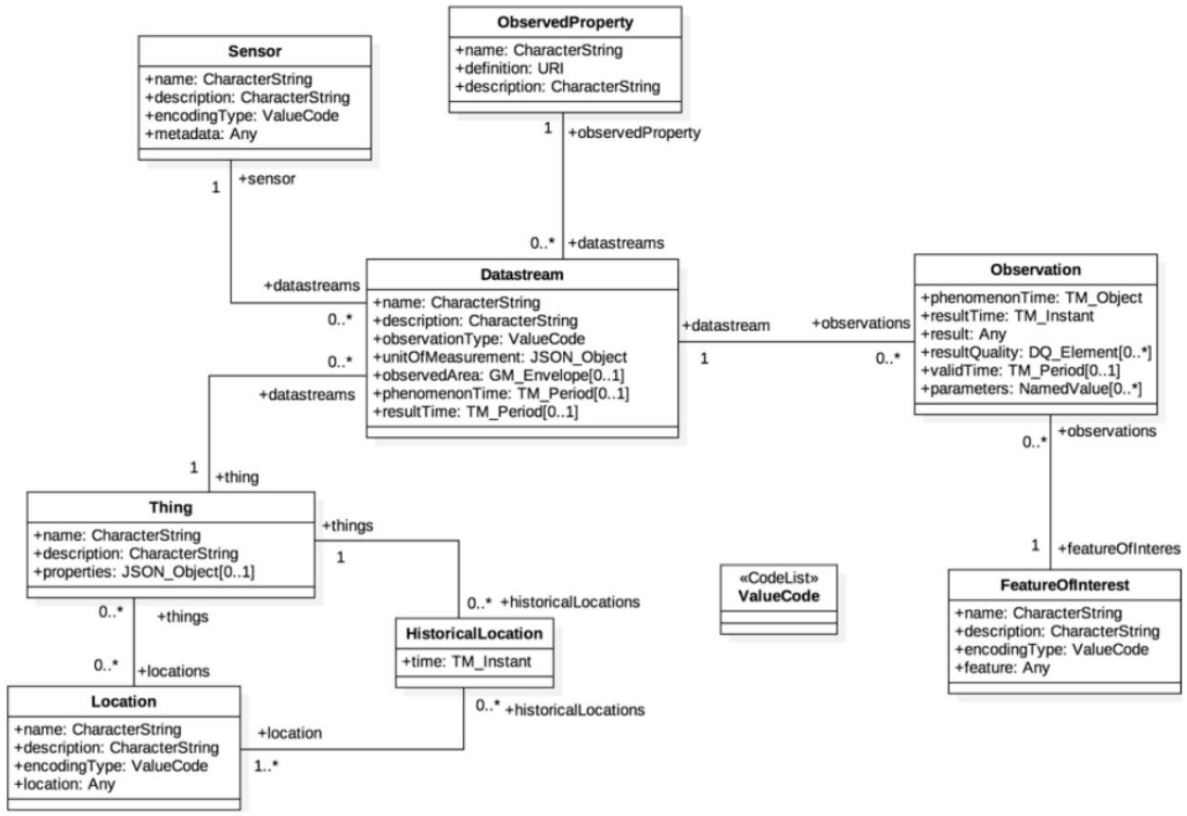


Figure 10 OGC Sensorthings API data model

Relevant uses cases and examples of the SensorThings API that are useful for SCOREwater have not been found. The FIWARE NGSI-LD and the OGC Sensorthings API are partly competing standards. Within SCOREwater we will first focus on the FIWARE NGSI API for the different demonstration projects.

4. SOFTWARE FOR SCOREWATER PLATFORM

4.1. INTRODUCTION

This chapter focuses on selecting software for the SCOREwater platform. Whenever possible, FIWARE generic enablers will be used. If that is not possible, other open source alternatives will be used. The chapter starts by describing the software architecture of the SCOREwater platform using the FIWARE catalogue. The SCOREwater platform will consist of a mixture of standard and tailor made software. To evaluate the different options a framework to evaluate the different components is needed. The ISO/IES25010 model for evaluation of software will be used to structure the selection of software for the SCOREwater platform. Once this model has been introduced, it will be used to discuss the candidate software to implement the SCOREwater platform.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

4.2. FIWARE AND SCOREWATER

Figure 11 is taken from the FIWARE web site. It is a high level overview of the FIWARE Catalogue. The FIWARE Catalogue is a curated framework of open source platform components which can be assembled together and with other third-party platform components to accelerate the development of smart, data driven platforms such as the SCOREwater platform. The main and only mandatory component of any "Powered by FIWARE" platform or solution is the FIWARE Orion Context Broker generic enabler, which brings a cornerstone function in any smart solution: the need to manage context information, enabling to perform updates and bring access to context. Building around the FIWARE Context Broker, a rich suite of complementary FIWARE components are available, dealing with:

- Interface with the Internet of Things (IoT), Robotics and third-party systems. This relates to capturing information from sensors and other third party systems. This component is also responsible for controlling actuators, but this has not come up in one of the SCOREwater cases;
- Core Context Management is at the heart of any FIWARE compatible solution. This is where the information from sensors and other third party systems is managed. To be FIWARE compatible, a context broker must be present in this components, but it also contains the component to create the time series database and API's to provide other components with access to the data;
- Context Processing, Analysis and Visualization uses the data managed in the core context management component to generate additional added value for the end user. This is where the SCOREwater data driven models will be running;
- Context Data/API management, publication, and monetization allows owners of the data to control who has access to the data available on the platform, under what conditions.

Deployment tools provides the hard- and software infrastructure the FIWARE platform is hosted on.

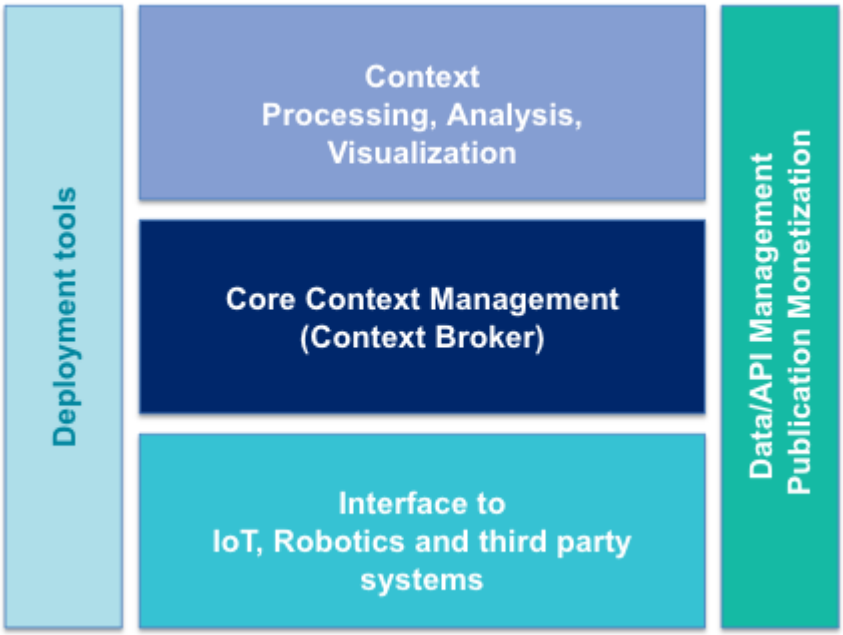


Figure 11 FIWARE catalogue (FIWARE, 2020)

Figure 12 contains the SCOREwater platform implementation of the FIWARE catalogue. It is still a high level overview which needs further detailing within the frame of implementation of the SCOREwater platform for the different cases.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

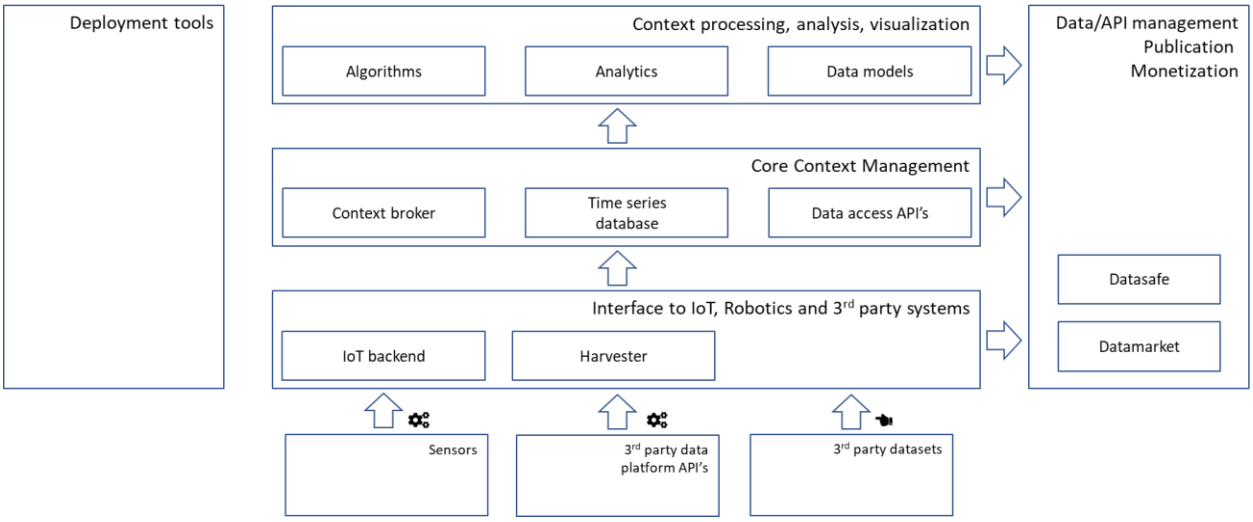


Figure 12 Translation of FIWARE architecture to SCOREwater platform architecture

For the "Interface to IoT, Robotics and 3rd party systems", several IoT agents and 3rd party data harvesters will be needed. Some sensors and 3rd party API's may be supported by standard software, others might have to be tailor made within the frame of SCOREwater. In the "Core Context Management" component one or more context brokers will be deployed, depending on the need for different NGSI versions. It will also contain the components which connect to the context broker(s) to create the time series databases. Either standard software will be used for this purpose, or custom components may be created in case of optimizations needed which are difficult to accomplish using standard software. The "Core Context Management" component will also include an open data platform allowing SCOREwater data providers to manage their datasets and others to actually use the datasets. "Context processing, analysis, visualization" data driven models, dashboards. "Data/API management Publication Monetization" Datasafe for safe access to secured data Data Market to grant access to API's. Standard software configured.

4.3. EVALUATION OF SOFTWARE QUALITY USING ISO/IEC 25000

The ISO/IEC 25000 standards provide a model for the evaluation of software product quality (International Standards Organization, 2020). This model will be applied to structure the requirements for the software used to implement the SCOREwater platform. ISO/IEC25000 consists of multiple divisions. ISO/IEC 25010 is part of the so called "Quality Model Division" and describes software quality using characteristics and sub characteristics. Sub characteristics can be measured using so called metrics. Creating an extensive model for each of the components used for the SCOREwater platform is beyond the scope of this deliverable, but the ISO/IEC 25010 does provide a framework for structuring the requirements. When selecting a component, it will be indicated whether a ISO/IEC 25010 characteristic is relevant for a component. Candidates will be compared using those relevant characteristics.

Figure 13 ISO 25010 characteristics and sub characteristics (International Standards Organization, 2020)

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Table 1 ISO25010 characteristics and their relevance for the SCOREwater platform

Functional Suitability	
Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. This characteristic is relevant for all the components of the SCOREwater platform, if a component cannot perform the functions it is intended to, there is no point in including it in the platform.	
Performance efficiency	
Performance relative to the amount of resources used under stated conditions. This characteristic is mostly relevant for server side components, they must be able to handle the expected load and process data within the expected time frame.	
Compatibility	
Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment. This is relevant for all components of the SCOREwater platform. They must a) run on the proper platforms (for server side components this means running on CentOS and on Docker, for client side components this means running on a specific list of devices - all modern browsers probably), b) support the proper standards (compatible with NGSI) and c) be compatible with the database systems avaiable.	
Usability	
Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. This characteristic is relevant for components with which end users must interact. It is less relevant for components only used by a limited number of administrator users since a limited number of users can be trained in using a certain software package (e. g. the configuration of a server side component).	
Reliability	
Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is relevant for mission critical server side components collecting data. They must be highly available, easy to recover and not fail when faulty data are inserted.	
Security	
Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. This characteristic is relevant for all components.	
Maintainability	
This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements.	

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Within the frame of the SCOREwater platform this means that software must be supported by a vibrant community. If such software is not available and custom software must be developed, it must adhere to software engineering best practices.
Portability
Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is not relevant. All the software will be available in the form of OCI (Open Container Interface) images, aka Docker images. These images can be run on a multitude of platforms.

The license under which software is released is not part of ISO/IEC 25010, but the software used for the SCOREwater platform should be available under an open source license. An open source license allows for the unlimited redistribution of software. Proprietary software will not be considered.

4.4. DEPLOYMENT TOOLS

For the duration of the project, the SCOREwater platform will be hosted using the Civity hard- and software infrastructure. Therefore, existing deployment tools such as the operating system, databases, web servers etcetera will be used as much as possible. Software to be installed within the frame of SCOREwater should be compatible with this infrastructure. For now focus is on researching other software components of the software stack needed to implement the SCOREwater platform.

4.4.1. OPERATING SYSTEM

The Civity software infrastructure is running on virtual machines on which the CentOS 7 operating system is installed. Therefore, all software installed within the frame of the SCOREwater platform should run on CentOS. To facilitate deployment of the SCOREwater platform after the project, Open Container Initiative (OCI) images (aka Docker images) will be created. These can be run on various platforms. Documentation and guides to deploy these Docker images will be supplied.

4.4.2. WEB SERVER

It is not that relevant to investigate different options in that domain. Civity currently uses the de facto industry standards HAProxy for load balancing and Apache httpd to serve http and https.

4.4.3. DATABASES

The same goes for databases, investigating different database options is not our initial focus. This may change though should we run into limitations using the currently used database software. At the time of writing, PostgreSQL and MongoDB databases are being hosted in the Civity infrastructure. Installation and configuration of additional database systems, specifically to support time series, is an option. Other (partially) open source database systems are for example CrateDB (Crate.io, 2020) or InfluxDB (Influxdata, 2020).

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

4.5. INTERFACE TO IOT, ROBOTICS AND THIRD PARTY SYSTEMS

4.5.1. IOT BACKEND

Characteristic	
Functional suitability	IoT sensors usually upload the data they produce in some highly optimized proprietary format. Purpose of the IoT backend is to receive those messages, convert the proprietary sensor data format to NGSI and pass the standardized data on to the context broker.
Performance efficiency	Processing should take place in near real time.
Compatibility	NGSI, NGSI version 2, NGSI LD.
Usability	n/a
Reliability	The IoT backend should lose as little data as possible.
Security	The IoT backend should not accept data from sensors which have not been properly registered (device/asset management)
Maintainability	n/a
Portability	Run on CentOS and in Docker, use PostgreSQL or MongoDB database

Generic enablers

IDAS	
Functional suitability	☒ The IDAS Generic Enabler consists of a number of IoT agents which interface with devices using some widely used IoT protocols (LWM2M over CoaP, JSON or UltraLight over HTTP/MQTT or OPC-UA)
Performance efficiency	☐ to be assessed
Compatibility	☒ IDAS supports both NGSI version 2 and NGSI LD
Usability	n/a
Reliability	☐ to be assessed
Security	☒ The FIWARE catalogue contains components to secure the IDAS Generic Enablers.
Maintainability	☒ Supported by FIWARE
Portability	☒ Written in NodeJS, runs on both CentOS and Docker

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Alternatives

Custom IoT agents	
Functional suitability	☒ Custom IoT agents are tailor made for a specific sensor and allow for a highly optimized translation from the proprietary sensor data format to NGSI.
Performance efficiency	☒ Capacity can be increased as needed
Compatibility	☒ Both NGSI version 2 and NGSI LD can be supported
Usability	n/a
Reliability	☒ Existing Snifferbike implementation demonstrates that a reliable set-up is possible
Security	☒ Secure using Keycloak
Maintainability	☒ Supported by SCOREwater
Portability	☒ Custom IoT agents will be developed in Java. Runs in both CentOS and Docker, can use any database for which a JDBC driver is available.

4.5.2. 3RD PARTY HARVESTERS

Characteristic	
Functional suitability	Purpose of the 3rd party harvesters is to periodically visit 3rd party API's and download and store metadata (and data if needed) which has been modified since the previous visit. The schedule should be configurable.
Performance efficiency	There is a tradeoff between the time the harvester needs to complete a task and the scheduler allows between runs. The harvester should be fast enough to be finished before the next run starts.
Compatibility	The 3rd party harvester should be compatible with different 3rd party API's. The list of API's to be harvested within the frame of SCOREwater is not yet complete. A lot of the API's which are known at the time of writing use proprietary API's.
Usability	n/a
Reliability	The 3rd party harvester should harvest all applicable metadata and data.
Security	The 3rd party harvester should respect security considerations of the 3rd party API.
Maintainability	n/a
Portability	Run on CentOS and Docker, use PostgreSQL or MongoDB database

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Generic enablers

There are no Generic Enablers available for this purpose. It is difficult to come up with a generic 3rd party data API harvester due to the lack of standardization and the variation in API's.

Alternatives

Custom harvesters	
Functional suitability	☒ Custom 3rd party harvesters are tailor made for a specific API and allow for a highly optimized translation from the proprietary API to NGSI. Civity has created several 3rd party harvesters in the past using 3rd party libraries such as OGR and FME.
Performance efficiency	☒ Implementation of initial harvesters shows that sufficient performance can be achieved by tailoring indexes in the database towards the queries needed
Compatibility	☒ Both NGSI version 2 and NGSI LD can be supported
Usability	n/a
Reliability	☒
Security	☒ Harvesters can be run a server which cannot be accessed from the internet
Maintainability	☒ Supported by SCOREwater
Portability	☒ Written in Java, runs in both CentOS and Docker, uses a PostgreSQL database

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

4.6. CORE CONTEXT MANAGEMENT

4.6.1. CONTEXT BROKER

Characteristic	
Functional suitability	The context broker is at the heart of any FIWARE compliant platform or solution. The context broker receives messages with sensor data from the IoT agents and publishes updates with those data. Components interested in those update subscribe to those updates (publish-subscribe).
Performance efficiency	Real time
Compatibility	The context broker should be able to receive and publish NGSI version 2 and NGSI-LD entities.
Usability	n/a
Reliability	No data loss
Security	Device management
Maintainability	Support from community
Portability	Run on CentOS and in Docker, use PostgreSQL or MongoDB database

Generic enablers

Orion Context Broker	
Functional suitability	☒ The "original" context broker.
Performance efficiency	☒ Messages are processed in real time
Compatibility	☐ NGSI (legacy) and NGSI version 2
Usability	n/a
Reliability	☒ Able to handle the expected load as demonstrated by for instance Snifferbike (Provincie Utrecht, Civity, SODAQ, RIVM, 2020)
Security	☒ The FIWARE catalogue contains components to secure the Orion Context Broker
Maintainability	☒ Support from FIWARE
Portability	☒ Written in C++, runs in both CentOS and Docker, uses a MongoDB database
	(FIWARE, 2020)

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Alternatives

Sensative Context Broker	
Functional suitability	☒ Alternative for Orion context broker.
Performance efficiency	☒ Messages are processed in real time
Compatibility	☐ NGSI version 2
Usability	n/a
Reliability	☐ To be evaluated
Security	☒ The Sensative context broker uses Keycloak secure access
Maintainability	☒ Support from Sensative and Civity
Portability	☒ Written in NodeJS, runs on both CentOS and Docker, uses a MongoDB database

Scorpio NGSI-LD Broker	
Functional suitability	☒ Context broker for NGSI-LD.
Performance efficiency	☒ Messages are processed in real time
Compatibility	☐ NGSI-LD
Usability	n/a
Reliability	☒
Security	☒ The FIWARE catalogue contains components to secure the Scorpio NGSI-LD Broker
Maintainability	☒ Support from FIWARE
Portability	☒ Written in Java, runs on both CentOS and Docker, uses a PostgreSQL database
	(NEC Laboratories Europe and NEC Technologies India, 2020)

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Orion-LD	
Functional suitability	☒ Alternative for Scorpio NGSI-LD Broker.
Performance efficiency	☒ Messages are processed in real time
Compatibility	☐ NGSI-LD
Usability	n/a
Reliability	☐
Security	☒ The FIWARE catalogue contains components to secure the Orion-LD context broker
Maintainability	☐ Orion-LD is still in alpha phase
Portability	☒ Written in C++, runs on both CentOS and Docker, uses a MongoDB database
	(FIWARE / context.Orion-LD, 2020)

4.6.2. TIME SERIES DATABASE

Characteristic	
Functional suitability	Purpose of the component to create the time series database is to subscribe to context broker updates and persist each update in a preferably configurable time series database back-end. At least PostgreSQL and/or MongoDB should be supported.
Performance efficiency	Updates should be processed near real time.
Compatibility	The time series database should be able to process NGSI version 2 and NGSI-LD updates.
Usability	n/a
Reliability	The time series database should lose as little data as possible. Each update sent out by the context broker should end up in the database.
Security	n/a (this component will not be accessible from internet)
Maintainability	n/a
Portability	Run on CentOS and Docker, use PostgreSQL or MongoDB database

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Generic enablers

Cygnus	
Functional suitability	☒ Cygnus subscribes to a context broker and persists updates published by this context broker in a so called data sink, for instance a PostgreSQL, CKAN or Hadoop data sink.
Performance efficiency	☐ Past experiences show that Cygnus performance depends on the data sink used. Performance of some of the sinks we might want to use might be insufficient.
Compatibility	☐ Cygnus only supports NGSI (legacy)
Usability	n/a
Reliability	☐ Cygnus uses an in memory queue. This queue has limited capacity, causing data loss when it fills up. Upon restart, records in the queue are lost as well.
Security	n/a
Maintainability	☒ Support from FIWARE
Portability	☒ Runs in both CentOS and Docker
	(FIWARE, 2020)

Draco	
Functional suitability	☒ Draco subscribes to a context broker and persists updates published by this context broker using a so called processor. Currently a PostgreSQL, MySQL, and MongoDB processor are available. Draco is based on Apache NIFI
Performance efficiency	☐ To be evaluated
Compatibility	☐ Draco supports NGSI version 2 and NGSI-LD
Usability	n/a
Reliability	☐ To be evaluated
Security	n/a
Maintainability	☒ Support from FIWARE
Portability	☒ Runs on both CentOS and Docker
	(FIWARE, 2020)

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

STH Comet	
Functional suitability	☐ STH (Short Term History) Comet is used to create short term time series databases in a MongoDB database
Performance efficiency	☐ To be evaluated
Compatibility	☐ To be evaluated
Usability	n/a
Reliability	☐ To be evaluated
Security	n/a
Maintainability	☒ Support from FIWARE
Portability	☒ Written in Node.js, runs on both CentOS and Docker, uses a MongoDB database
	(FIWARE, 2020)

Alternatives

Custom time series database	
Functional suitability	☒ Custom time series databases allow for very flexible, highly optimized data models and conversion from NGSI to this data model. Adding support for a new entity type requires a bit of coding (not just configuration).
Performance efficiency	☒ This approach was used by Civity for "Snuffelfiets". Whereas Cygnus cannot always keep track of "Snuffelfiets" updates, the tailor made time series database does.
Compatibility	☒ It is up to the developer implementing the conversion to use either NGSI, NGSI version 2 and/or NGSI-LD.
Usability	n/a
Reliability	☒
Security	n/a
Maintainability	☒ Support from SCOREwater
Portability	☒ Written in Java, runs in both CentOS and Docker, can use any database

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

4.6.3. OPEN DATA PLATFORM

Characteristic	

Functional suitability	The open data platform plays an important role in the SCOREwater platform. It is used for the following purposes:
	• Publish metadata on datasets
	• Publish metadata on services
	• Upload of datasets
	• Organize datasets according to
	  • Organization
	  • Theme
	• Harvesting of metadata
	All functions of the open data platform should be available via a user interface and an API.

Performance efficiency	
Compatibility	Metadata: • DCAT • ISO19139 Data: • CSV • XLS • Shape • Geopackage • JSON • GeoJSON
Usability	The end-user portal should be very user friendly. The maintenance portal should be user friendly as well, but those users can be trained so it is not as important as the user friendliness of the end-user portal.
Reliability	
Security	Prevent unauthorized access
Maintainability	n/a
Portability	Run on CentOS and Docker

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Generic enablers

CKAN	
Functional suitability	☒ CKAN allows organizations the publish metadata and data
Performance efficiency	☒ CKAN is used by numerous big organizations attracting a lot of traffic
Compatibility	☒ CKAN can support all the relevant standards
Usability	n/a
Reliability	☒ CKAN is used by numerous big organizations attracting a lot of traffic without reliability issues
Security	☒ Authorization and authentication using username and password, OAuth2
Maintainability	☒ Support form community
Portability	☒ Written in Python, runs in both CentOS and Docker, uses a PostgreSQL database
	(CKAN, 2020)

Alternatives

Geonetwork Open Source	
Functional suitability	☐ Main purpose of GeoNetwork Open Source is a metadata catalog focusing on spatially referenced resources.
Performance efficiency	☒ Geonetwork Open Source is used by numerous big organizations attracting a lot of traffic without performance issues
Compatibility	☒ Geonetwork Open Source can support all the relevant standards
Usability	n/a
Reliability	☒ Geonetwork Open Source is used by numerous big organizations attracting a lot of traffic without reliability issues
Security	☒ Authorization and authentication using username and password, OAuth2
Maintainability	☒Support from community
Portability	☒ Written in Java, runs in both CentOS and Docker, uses a PostgreSQL database
	(Open Source Geospatial Foundation, 2020)

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

4.6.4. CONTEXT PROCESSING

Data driven models are custom implementations by nature. Two programming languages will be supported for the implementation of data driven models: Python 3 and Java. Within the frame of the project, re-usable libraries will be created to be able to subscribe to NGSI messages.

4.6.5. AUTHENTICATION AND AUTHORIZATION

Characteristic	

Functional suitability	• Realms
	• Authentication and authorization using
	  ○ username and password
	  ○ API key

Performance efficiency	Must be able to handle the expected load
Compatibility	OAuth2
Usability	End user facing pages
Reliability	
Security	Prevent unauthorized access
Maintainability	Supported by community or large organization
Portability	Run on CentOS and in Docker

Generic enablers

Keyrock	
Functional suitability	☒
Performance efficiency	☒
Compatibility	☒
Usability	n/a
Reliability	☒
Security	n/a
Maintainability	n/a
Portability	☒

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Alternatives

Keycloak	
Functional suitability	☐
Performance efficiency	☒
Compatibility	☒
Usability	n/a
Reliability	☒
Security	n/a
Maintainability	n/a
Portability	☒

4.6.6. API MANAGEMENT

Characteristic	
Functional suitability	Twofold purpose: gateway and configuration
Performance efficiency	Be able to handle expected load
Compatibility	OpenAPI specification
Usability	n/a
Reliability	
Security	Play well with authentication and authorization tools to prevent unauthorized access
Maintainability	n/a
Portability	Run on CentOS and Docker, use PostgreSQL or MongoDB database

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Generic enablers

APinf	
Functional suitability	☐ Limited functionality
Performance efficiency	☐ Not evaluated
Compatibility	☐ Not evaluated
Usability	☐ Not evaluated
Reliability	☐ Not evaluated
Security	☐ Not evaluated
Maintainability	☐ Supported by FIWARE, small community
Portability	☐ Not evaluated

Most of the characteristics have not been evaluated because the functionality of APinf is based on API-umbrella, and there is limited community-activity for APinf. The site https://apinf.com/ is not working.

Alternatives

3Scale	
Functional suitability	☒
Performance efficiency	☒
Compatibility	☒
Usability	☐
Reliability	☒
Security	☒
Maintainability	☒ Supported by RedHat
Portability	☒

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Apiman	
Functional suitability	☒ From past experience it is known that Apiman supports the required functionality.
Performance efficiency	☐ Not evaluated
Compatibility	☐ Not evaluated
Usability	☐ Not evaluated
Reliability	☐ Not evaluated
Security	☐ Not evaluated
Maintainability	☐ Apiman is no longer maintained
Portability	☐ Not evaluated

Most of the characteristics have not been evaluated since Apiman is no longer actively maintained. Using Apiman for a new project would not be a wise decision.

Kong	
Functional suitability	☐ Limited functionality in the open source variant. Proprietary extensions needed to
Performance efficiency	☐ Not evaluated
Compatibility	☐ Not evaluated
Usability	☐ Not evaluated
Reliability	☐ Not evaluated
Security	☐ Not evaluated
Maintainability	☐ Not evaluated
Portability	☐ Not evaluated

Most of the characteristics of Kong have not been evaluated because of the insufficient functionality in the open source variant of the software.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Gravitee	
Functional suitability	☒ Functionality rich open source solution with two components: API-management (API Gateway, Management API and API Portal) and Access Management (Authorization Gateway, Management Portal). Good documentation. Installation guide for CentOS. Can use MongoDB and Elastic. Each Gravitee.io component provides its own technical API.
Performance efficiency	☐
Compatibility	☒
Usability	☒
Reliability	☒
Security	☒
Maintainability	☒
Portability	☒

WSO2	
Functional suitability	☐
Performance efficiency	☐
Compatibility	☐
Usability	☐
Reliability	☐
Security	☐
Maintainability	☐
Portability	☐

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

4.7. CONCLUSION

Figure 14 contains the FIWARE catalogue again (and as such a copy of Figure 12), but this time the software which will be used to implement the different components has been added in *italic text*.

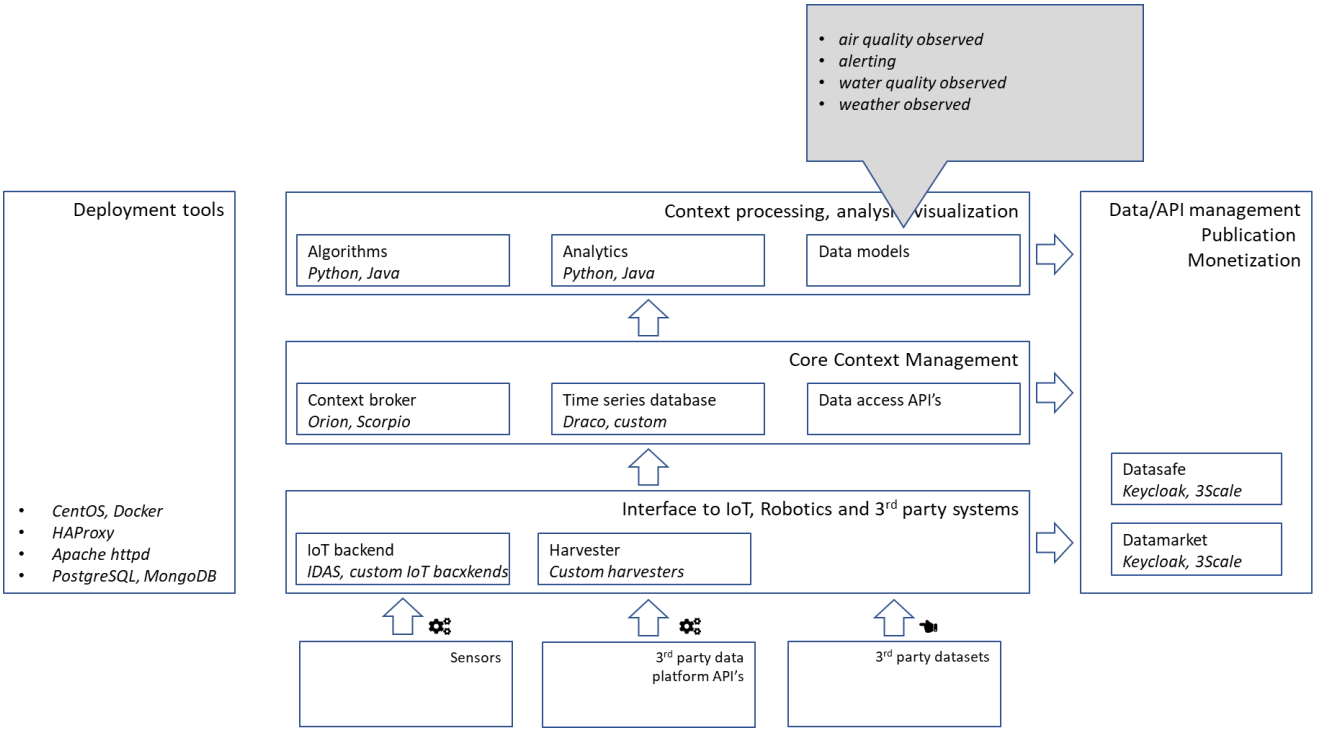


Figure 14 FIWARE catalogue with selection of software for SCOREwater platform

4.7.1. DEPLOYMENT TOOLS

The existing Civity infrastructure will be used.

4.7.2. INTERFACE TO IOT, ROBOTICS AND 3RD PARTY SYSTEMS

Whenever possible IoT agents available from IDAS will be used. If this is not possible (for instance because of a proprietary protocol or optimizations which are difficult to implement using off the shelf software), custom IoT agents will be developed.

Harvesters for 3rd party API's will be custom made since these 3rd party API's are usually proprietary. Existing software (e. g. software to generate client implementations from API documentation automatically) will be used as much as possible.

4.7.3. CORE CONTEXT MANAGEMENT

Since there is not one context broker available which supports both NGSI version 2 and NGSI-LD, two context brokers will be deployed within the frame of the project: an Orion context broker to support NGSI version 2 and a Scorpio context broker to support NGSI-LD.

Time series databases will be created using Draco whenever possible. In case of for instance optimizations needed we might have to resort to custom components to create a time series database.

CKAN will be used to provide access to the data API's: manual upload of 3rd party datasets, access to metadata and data, harvesting of 3rd party metadata API's

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

4.7.4. CONTEXT PROCESSING, ANALYSIS AND VISUALIZATION

Implementation of data driven models will take place using Java or Python 3. Re-usable modules to subscribe to context broker updates will be developed for both Java and Python 3.

4.7.5. DATA/API MANAGEMENT, PUBLICATION, MONETIZATION

Keycloak for authorization and authentication

3Scale for API management

With the combination of these tools, any kind of API-communication can be managed in a secure and reliable way. The API-management solution is a proxy between the source systems and the API-users. It is used to describe policies, plans and access-methods to API´s.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

5. REFERENCES

(n.d.). Retrieved from ICT4water: https://www.ict4water.eu

CKAN. (2020, 05 01). *CKAN, the world's leading Open Source data portal platform*. Retrieved from https://ckan.org/

Crate.io. (2020, 05 01). *Crate.io - The #1 database for IoT-scale*. Retrieved from https://crate.io/

EIP-SCC. (n.d.). Retrieved from EIP-SCC: https://eu-smartcities.eu/

FIWARE / context.Orion-LD. (2020, 04 09). Retrieved from FIWARE / context.Orion-LD: https://github.com/FIWARE/context.Orion-LD

FIWARE. (2020, 05 04). *Air quality observed*. Retrieved from FIWARE-datamodels: https://fiware-datamodels.readthedocs.io/en/latest/Environment/AirQualityObserved/doc/spec/index.html

FIWARE. (2020, 05 04). *Alert data model*. Retrieved from FIWARE-datamodels: https://fiware-datamodels.readthedocs.io/en/latest/Alert/doc/spec/index.html

FIWARE. (2020, 04 09). *Cygnus*. Retrieved from FIWARE-Cygnus: https://fiware-cygnus.readthedocs.io/en/latest/

FIWARE. (2020, 05 01). *FIWARE catalogue*. Retrieved from https://www.fiware.org/developers/catalogue/

FIWARE. (2020, 04 09). *FIWARE Draco*. Retrieved from FIWARE-Draco: https://fiware-draco.readthedocs.io/en/latest/

FIWARE. (2020, 05 04). *Water quality*. Retrieved from FIWARE-datamodels: https://fiware-datamodels.readthedocs.io/en/latest/Environment/WaterQualityObserved/doc/spec/index.html

FIWARE. (2020, 05 04). *Weather observed*. Retrieved from FIWARE-datamodels: https://fiware-datamodels.readthedocs.io/en/latest/Weather/WeatherObserved/doc/spec/index.html

FIWARE. (2020, 04 09). *Welcome to Orion Context Broker*. Retrieved from FIWARE-Orion: https://fiware-orion.readthedocs.io/en/master/

FIWARE. (2020, 04 09). *Welcome to the FIWARE Short Term Historic (STH) - Comet documentation*. Retrieved from FIWARE-STH-Comet: https://fiware-sth-comet.readthedocs.io/en/latest/

Influxdata. (2020, 05 01). *Real-time visibility into stacks, sensors and systems*. Retrieved from https://www.influxdata.com/

International Standards Organization. (2020, 02 07). *ISO/IEC 25010*. Retrieved from ISO 25000 Portal: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&limitstart=0

NEC Laboratories Europe and NEC Technologies India. (2020, 04 09). *ScorpioBroker / ScorpioBroker*. Retrieved from ScorpioBroker / ScorpioBroker: https://github.com/ScorpioBroker/ScorpioBroker

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Open Geospatial Consortium. (2020, 05 01). *OGC SensorThings API*. Retrieved from https://www.ogc.org/standards/sensorthings

Open Geospatial Consortium. (2020, 05 01). *opengeospatial / sensorthings*. Retrieved from Github: https://github.com/opengeospatial/sensorthings

Open Source Geospatial Foundation. (2020, 04 10). *GeoNetwork opensource*. Retrieved from GeoNetwork opensource: https://geonetwork-opensource.org/

Provincie Utrecht, Civity, SODAQ, RIVM. (2020, 05 01). *Snuffelfiets - Naar een fietsend meetnetwerk*. Retrieved from https://snuffelfiets.nl/

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

ANNEX 1 - STOCKTAKING

A final Annex of stocktaking was included in all Deliverables of SCOREwater produced after the first half-year of the project. It provides an easy follow-up of how the work leading up to the Deliverable has addressed and contributed to four important project aspects:

1. Strategic Objectives
2. Project KPI
3. Ethical aspects
4. Risk management

STRATEGIC OBJECTIVES

Table 2 lists those strategic objectives of SCOREwater that are relevant for this Deliverable and gives a brief explanation on the specific contribution of this Deliverable.

Table 2. Stocktaking on Deliverable's contribution to reaching the SCOREwater strategic objectives.

Project goal	Contribution by this Deliverable
The SCOREwater platform will be based on existing open source software components, standards and data models.	This deliverable selected the components, standards and models that will be used for the SCOREwater platform.
Identify existing systems and applications, and provide a functional and technical analysis of these systems and applications, including relevant standards, connections and data.	This deliverable has identified and described solutions and alternatives for the SCOREwater platform, based on a software quality model.
A prerequisite of the project is to base the SCOREwater platform on FIWARE	This deliverable has identified gaps in FIWARE components, standards and data models

PROJECT KPI

Table 3 lists the project KPI that are relevant for this Deliverable and gives a brief explanation on the specific contribution of this Deliverable.

Table 3. Stocktaking on Deliverable's contribution to SCOREwater project KPI's.

Project KPI	Contribution by this deliverable
Open source software by default	This deliverable has selected open source components, models and standards to be implemented in the SCOREwater platform.
FIWARE as prerequisite	This deliverable has identified and selected FIWARE components that are suitable for the SCOREwater platform

ETHICAL ASPECTS

Table 4 lists the project's Ethical aspects and gives a brief explanation on the specific treatment in the work leading up to this Deliverable. Ethical aspects are not relevant for all Deliverables. Table 4 indicates "N/A" for aspects that are irrelevant for this Deliverable.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

Table 4. Stocktaking on Deliverable's treatment of Ethical aspects.

Ethical aspect	Treatment in the work on this Deliverable
Justification of ethics data used in project	N/A
Procedures and criteria for identifying research participants	N/A
Informed consent procedures	N/A
Informed consent procedure in case of legal guardians	N/A
Filing of ethics committee's opinions/approval	N/A
Technical and organizational measures taken to safeguard data subjects' rights and freedoms	In accordance with D9.x where applicable
Implemented security measures to prevent unauthorized access to ethics data	In accordance with D9.x where applicable
Describe anonymization techniques	In accordance with D9.x where applicable
Interaction with the SCOREwater Ethics Advisor	N/A

RISK MANAGEMENT

Table 5 lists the risks, from the project's risk log, that have been identified as relevant for the work on this Deliverable and gives a brief explanation on the specific treatment in the work leading up to this Deliverable.

Table 5. Stocktaking on Deliverable's treatment of Risks.

Associated risk	Treatment in the work on this Deliverable
Technical immaturity of FIWARE components	Selection of open source alternatives, collaboration with other EU-funded projects and FIWARE-foundation.
Missing of incomplete standards and data models	Collaboration with other EU-funded projects, FIWARE-foundation and other standardization bodies to develop open standards and data models.

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751

scoreWATER

This project hase recieved funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no 820751