



URBANITE

Supporting the decision-making in urban transformation with
the use of disruptive technologies

Deliverable D3.8

Data aggregation and storage module implementation v2

Editor(s):	TEC, ENG
Responsible Partner:	TECNALIA
Status-Version:	Final – v2.0
Date:	30.09.2022
Distribution level (CO, PU):	PU

Project Number:	GA 870338
Project Title:	URBANITE

Title of Deliverable:	Data aggregation and storage module implementation v2
Due Date of Delivery to the EC:	30.09.2022

Workpackage responsible for the Deliverable:	WP3–Data Management Platform
Editor(s):	TEC
Contributor(s):	TEC, ENG
Reviewer(s):	Fraunhofer FOKUS
Approved by:	All Partners
Recommended/mandatory readers:	WP4, WP5

Abstract:	This deliverable is the second version of the software implementation for the data aggregation and storage module. This deliverable is the result of Task3.3.
Keyword List:	Storage, Aggregation, Fusion, Catalogue, Software
Licensing information:	This document is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v1.0	04/09/2021	First version of D3.8	TECNALIA
V1.1	21/06/2022	Update of components, APIs and chapter describing changes in v2	TECNALIA
V1.2	29/07/2022	Content revision. Extension of data aggregation and fusion sections	TECNALIA
V1.3	12/09/2022	Description of new data models (noise, census, etc.-) and update of the installation instructions	TECNALIA
V1.4	16/09/2022	Version ready for internal review	TECNALIA
V1.5	27/09/2022	Internal review	FRAUNHOFER
V1.6	30/09/2022	Final version ready for submission	TECNALIA

DRAFT VERSION

Table of Contents

Table of Contents	4
List of Figures	6
List of Tables.....	6
Terms and abbreviations.....	8
Executive Summary	9
1 Introduction	10
1.1 About this deliverable	10
1.2 Document structure	10
1.3 Updates with respect to v1	10
2 Implementation.....	12
2.1 Functional description.....	12
2.1.1 Fitting into overall URBANITE Architecture.....	12
2.2 Technical description	13
2.2.1 Data Aggregation.....	13
2.2.1.1 Traffic Flow Data Aggregation.....	14
2.2.1.2 Bike Data Aggregation.....	15
2.2.1.3 Design & Implementation	17
2.2.2 Data Fusion.....	19
2.2.3 Data Storage & Retrieval.....	20
2.2.3.1 Challenges and architectural design	21
2.2.3.2 Storage Layer.....	22
2.2.3.3 Access Layer.....	26
2.2.3.4 Technical specifications.....	27
2.2.4 Data Catalogue	27
3 Data Storage & Retrieval & Aggregation -delivery and usage	29
3.1 Installation instructions.....	29
3.2 User Manual	31
3.3 Licensing information.....	31
3.4 Download	31
4 Data Catalogue -delivery and usage.....	31
4.1 Installation instructions.....	31
4.2 User Manual	31
4.3 Licensing information.....	35
4.4 Download	35
5 Conclusions	36
6 References.....	37

7	APPENDIX: Data models	38
7.1	Traffic Flow Observed	38
7.2	Air QualityObserved	38
7.3	WeatherObserved	39
7.4	Calendar and Day Specification	39
7.4.1	DaySpecification	40
7.4.2	Calendar	40
7.5	Event.....	41
7.6	Transport Station.....	42
7.7	Point of Interest	43
7.8	GtfsShape	43
7.9	Tourist Trip	44
7.10	Origin Destination Matrix.....	45
7.11	Population and household models	46
7.11.1	CensusObserved	46
7.11.2	PopulationObserved.....	47
7.12	ElectroMagneticObserved.....	51
7.13	NoiseLevelObserved.....	51
7.14	MapLayer.....	52
7.15	Metadata.....	53
8	APPENDIX: Storage & Retrieval API	55
8.1	Storage	55
8.1.1	insertTData (POST)	55
8.1.2	updateTData (PUT).....	57
8.1.3	deleteTData (DELETE).....	60
8.2	Retrieval	61
8.2.1	getTData (GET)	61
8.2.2	getTData (single record) (GET)	64
8.2.3	getTDataRange (GET)	66
8.2.4	getSupportedDataModels (GET)	69
8.2.5	getDistinct (GET).....	71
8.3	Metadata.....	74
8.3.1	dataset (PUT).....	74
8.3.2	dataset (DELETE)	77
8.3.3	getDataset (GET)	78
8.3.4	getCatalogueDatasets (GET).....	80
8.3.5	searchDatasets (GET)	82

9	APPENDIX: Data Aggregation API.....	84
9.1	Types of Aggregators.....	84
9.2	aggregate (GET).....	85

List of Figures

FIGURE 1 - URBANITE ARCHITECTURE	13
FIGURE 2 - TIME SERIES STRUCTURE FOR TRAFFIC DATA AT A GIVEN ROAD LOCATION CORRESPONDING TO A SENSOR LOCATION.....	14
FIGURE 3 - INFORMATION RELATED TO THE START AND END POINTS OF THE BIKE TRAJECTORIES.....	15
FIGURE 4 - VISUALIZATION OF ONE OF THE DIRECT TESSELLATIONS PROVIDED IN URBANITE FOR THE DISTRICTS OF THE CITY OF BILBAO. THE NUMBER SHOWN FOR EACH DIVISION CORRESPONDS TO THE VALUE OF THE ZONE_ID.....	16
FIGURE 5 - VISUALIZATION OF VORONOI AREAS TESSELLATION OBTAINED BY URBANITE. THE INPUTS FOR THE GENERATION ARE THE LIMITING RECTANGLE AND THE SET OF RED POINTS MARKED IN THE MAP.	17
FIGURE 6 - OPENTSDDB ARCHITECTURE.....	18
FIGURE 7 - DATAPOINTS WITHOUT DOWNSAMPLING	18
FIGURE 8 - DATAPOINTS WITH DOWNSAMPLING	19
FIGURE 9 - DASHBOARD VISUALIZATION IN GRAFANA	19
FIGURE 10 – GET METHOD IN THE DATA AGGREGATION API.....	19
FIGURE 11 - DATA STORAGE & RETRIEVAL REPOSITORIES	20
FIGURE 12 - TECHNOLOGY STACK	22
FIGURE 13 – METHODS IN THE DATA STORAGE API TO INSERT, DELETE AND UPDATE DATA.....	26
FIGURE 14 - METHODS IN THE DATA RETRIEVAL API TO RETRIEVE DATA	27
FIGURE 15 - METHODS IN THE DATA STORAGE API TO MANAGE METADATA	27
FIGURE 16 - SCHEMA ABOUT URBANITE COMPONENTS (UI, DATA CATALOGUE AND CONNECTORS, DS&R). ..	28
FIGURE 17 - DATA CATALOGUE MANAGEMENT OF FEDERATED ODMS.....	32
FIGURE 18 - DATA CATALOGUE CONFIGURATION MANAGEMENT	32
FIGURE 19 - DATA CATALOGUE FEDERATED METADATA SEARCH BY TAG	32
FIGURE 20 - DATA CATALOGUE FEDERATED METADATA SEARCH	33
FIGURE 21 - INFORMATION FIELDS OF DATASET' METADATA SEARCH LIST	33
FIGURE 22 - DATA CATALOGUE FEDERATED METADATA DATASET DETAIL VIEW.....	34
FIGURE 23 - DATA CATALOGUE- DISTRIBUTION - INFORMATION ICON	34
FIGURE 24 - DATA CATALOGUE -DETAILS OF A DISTRIBUTION	34

List of Tables

TABLE 1: STATUS OF DATA FUSION, STORAGE AND RETRIEVAL/CATALOGUE REQUIREMENTS FROM D5.1	12
TABLE 2: STRUCTURE FOR DAY SPECIFICATION	40
TABLE 3: STRUCTURE FOR A CALENDAR SPECIFICATION	40
TABLE 4: API FOR DATA INSERTION.....	55
TABLE 5: API FOR DATA UPDATE	57

TABLE 6: API TO DELETE DATA	60
TABLE 7: API FOR DATA RETRIEVAL	61
TABLE 8: API FOR DATA RETRIEVAL (SPECIFIC RECORD).....	64
TABLE 9: API FOR DATA RETRIEVAL (TIME RANGE)	66
TABLE 10: API FOR THE RETRIEVAL OF THE DATA MODELS INFORMATION	69
TABLE 11: API FOR THE RETRIEVAL OF DISTINCT VALUES.....	71
TABLE 12: API FOR THE INSERT AND UPDATE OF METADATA	74
TABLE 13: API FOR THE DELETION OF METADATA.....	77
TABLE 14: API FOR THE RETRIEVAL OF DATASET METADATA	80
TABLE 15: API FOR THE SEARCH AND RETRIEVAL OF DATASET METADATA	82
TABLE 16: TYPES OF AVAILABLE AGGREGATORS.....	84

DRAFT VERSION

Terms and abbreviations

ACID	Atomic, Consistent Isolated, Durable
API	Application Programming Interfaces
BI	Business Intelligence
CKAN	Comprehensive Knowledge Archive Network
CSV	Comma Separated Values
DCAT	Data CATalog Vocabulary
DCAT-AP	Data CATalog Vocabulary - Application Profile
DKAN	Drupal-based Knowledge Archive Network
EC	European Commission
GNU	GNU's Not Unix
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
JAR	Java ARchive
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
JVM	Java Virtual Machine
NGSI-LD	Next Generation Service Interfaces- Linked Data
NO	Nitric Oxide
NoSQL	Not Only SQL
NO ₂	Nitrogen Dioxide
NO _x	Nitrogen Oxides
OD	Origin Destination
ODMS	Open Data Management Systems
PM ₁₀	Particulate Matter
RDBMS	Relational DataBase Management System
REST	Representational State Transfer
SO ₂	Sulfur Dioxide
SQL	Structured Query Language
SSL	Secure Socket Layer
TSD	Time Series Daemon
TSDB	Time Series Data Base
URL	Uniform Resource Locator

Executive Summary

This deliverable contains an overview of the software components that are related to the tasks of data aggregation, fusion, storage, retrieval and its catalogue. This refers to the process of mapping, aggregation, fusion, storage and retrieval of the curated data. A common model (based on Fiware’s data model) for storage of the information and knowledge extraction has been defined, handling the semantic processing of the curated data as well as the aggregation and deduplication of the data that originate from distinct sources. Finally, the storage strategy and implementation of a set of APIs for retrieval were developed. For each existing module described in this deliverable, an overview along with a description is given. Where applicable, details on configuration, installation and usage are provided. The components are implemented following a microservice approach, so they fit well with the global docker-based architecture.

DRAFT VERSION

1 Introduction

The term Data Management Platform stands for a variety of distinct software components that work together to deliver the key functionalities, that are data harvesting, data curation, and data aggregation and storage. The three deliverables D3.3, D3.6, and D3.8, focus on these core features respectively. Due to the interaction between these modules, the aforementioned deliverables should be understood as a collection of documents related to the same overarching concept that is the Data Management Platform. These three deliverables describe the final version of these components.

1.1 About this deliverable

Within the Data Management Platform, this deliverable focuses on the components related to data fusion and aggregation, data storage and retrieval and data catalogue. It presents fusion and aggregation techniques of interest for urban mobility, the challenges involved in storing and retrieving big volumes of data as well as managing heterogeneous formats, and a solution for managing datasets and related metadata. A dataset is a combination of data and metadata.

1.2 Document structure

Section 2.1 covers the functionalities provided by the four components related to data fusion and aggregation, data storage and retrieval and data catalogue, and their fitting to the general architecture defined in WP5. Section 2.2 describes the technical details of the different components in the Data Management Platform. Then, for each of the main modules, dedicated sections 3 and 4, present their installation instructions, a brief user manual, licensing information and the repository URL for downloading the source code. The document wraps up with a conclusion and references.

1.3 Updates with respect to v1

The main updates with respect to v1 of this document consist of the extension of the Data Storage component to support additional data models, improvements in the Data Retrieval Component API and the implementation of the Data Aggregation API.

- Extension of the Data Storage component: The following new models are now supported in order to meet the requirements of WP4: Event, GtfsShape, TouristTrip, PointOfInterest, TransportStation, CensusObserved, PopulationObserved, ElectroMagneticObserved, NoiseLevelObserved and OriginDestination Matrix. A new model named MapLayer has been included to fulfil the requirements of WP5. The detailed description of these models has been included in 7 APPENDIX: Data models.
- Implementation of the Data Aggregation API: In order to improve performance and reduce time response in the retrieval of big volumes of data, e.g. for traffic models, new time series storage has been implemented in OpenTSDB. The data aggregation API allows the retrieval of data aggregated by customized time periods, as well as operations on the data, such as the calculation of minimum and maximum values, etc.
- The Data Fusion section has been extended to include concrete examples of how data is fused in URBANITE and for which purposes.
- The Data Retrieval API has been extended to support the new data models and to provide some additional methods adapted to the needs of WP4, WP5 and WP6.
- In order to guarantee that only authorized users can store and retrieve data, the Data Storage component will be deployed without external access. This means that only components in the same network will have access to the endpoints. Regarding data

retrieval, a new API has been implemented, i.e. OpenDataRetrieval, that provides access only to the datasets that are offered openly. This way, access to proprietary data through the Data Retrieval component will be restricted and users will only have access through the OpenDataRetrieval to public data, aggregated data or to analysis and simulation results based on the data, but not to the raw data.

DRAFT VERSION

2 Implementation

2.1 Functional description

As mentioned in the introduction, this deliverable focuses on the 4 components related to data fusion and aggregation, data storage and retrieval and data catalogue.

The functional requirements for these components were listed in deliverable D5.1 and a detailed design was provided in deliverable D5.4. We present here a short summary and the status of development. All the requirements are fulfilled.

Table 1: Status of Data fusion, storage and retrieval/catalogue requirements from D5.1

Component	Requirements in D5.1 + current status
Data fusion/aggregation	<ul style="list-style-type: none"> • DF.01. Aggregation. The component should allow to aggregate curated data coming from different data sources if needed. (<i>fulfilled</i>) • DF02. Deduplication. The component should allow the deduplication of the data (<i>fulfilled</i>). • DF03. Data mapping. The data should be mapped into EU vocabularies (<i>fulfilled</i>). • DF.04. MetaData mapping. The metadata should be mapped into DCAT-AP metadata (<i>fulfilled</i>).
Data Storage	<ul style="list-style-type: none"> • DS.01. Big data storage. The harvested data should be persisted to a big data capable storage solution (<i>fulfilled</i>). • DS02. DCAT-AP compliance. The data storage component should be able to process and store DCAT-AP compliant metadata (<i>fulfilled</i>).
Data Retrieval / Data Catalogue	<ul style="list-style-type: none"> • DR.01. Data Retrieval. The data retrieval component must expose API to retrieve and query the data stored in the different repositories (<i>fulfilled</i>). • DR.02. Data Hub. The metadata stored in the repositories should be accessible through a data hub in a uniform way taking advantage of DCAT-AP standard and related profile (<i>fulfilled</i>).

2.1.1 Fitting into overall URBANITE Architecture

The four components described in this deliverable have been implemented following a microservice approach, so they fit well with the docker-based architecture designed in WP5. Besides, the Data Catalogue offers a user interface to manage and search through the datasets.

The components described in this deliverable are high-lighted in green in the architecture diagram from deliverable D5.4 below:

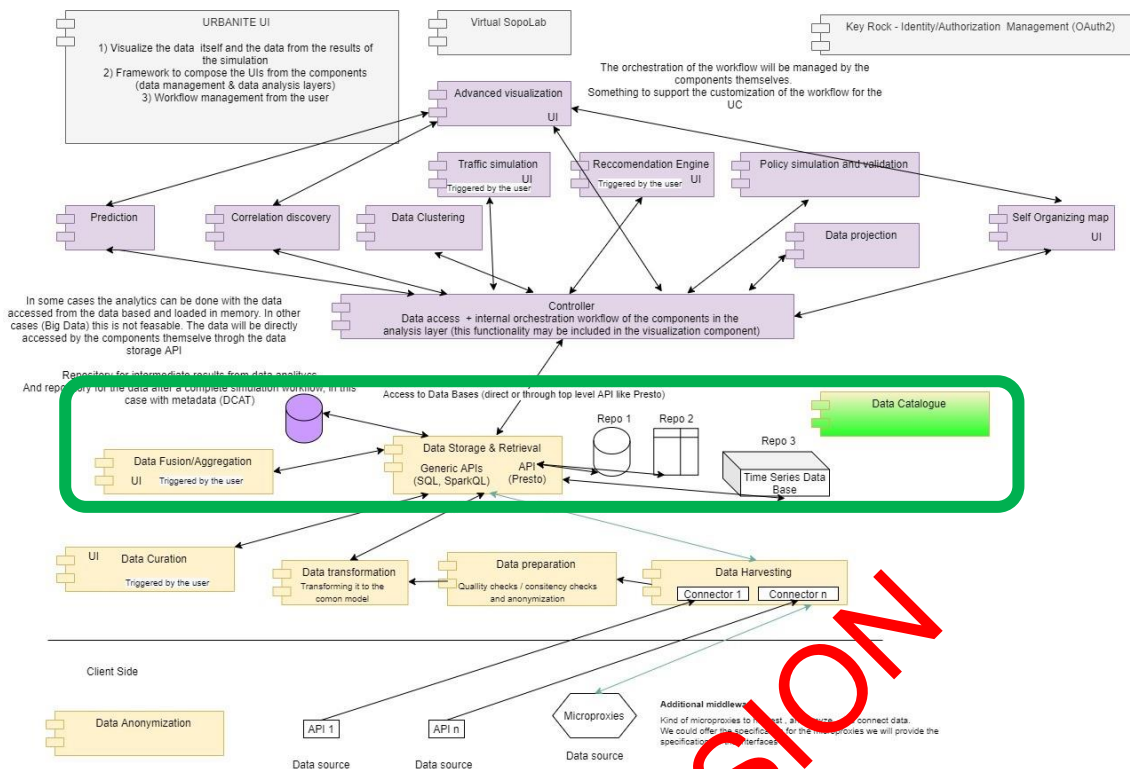


Figure 1 - URBANITE architecture

2.2 Technical description

This section describes the technical details of the different components in the Data Management Platform dedicated to data fusion & aggregation, data storage & retrieval and data cataloguing.

Data fusion and aggregation functionalities have been integrated into the different functional modules related to the analysis of traffic and mobility by bicycles.

2.2.1 Data Aggregation

Data aggregation is the process of gathering data and presenting it in a summarized format. Data aggregation is useful e.g. to remove personal information or to provide information in a synthetic form or to train the models in WP4.

In the case of **Traffic Data**, the following aggregated functions are calculated before performing the training of the Artificial Intelligence models in WP4:

- Initial date and time of the data, typically the Unix timestamp.
- End date and time of data, typically the Unix timestamp.
- Temporal aggregation period, typically 5 or 15 minutes.
- Maximum value of the traffic flow.
- Minimum value of the traffic flow.
- Number of holes within the data.
- Average period of each hole.
- Standard Deviation

Next, we provide a more detailed description of traffic flow data aggregation and bike data aggregation computed in URBANITE.

2.2.1.1 Traffic Flow Data Aggregation

Traffic flow data measured by sensors can be provided in different ways. One common way (as provided by the Bilbao use case) is to provide an aggregated time series structure describing the number of vehicles that have gone through a particular road at a given sensor location.

```

...
1631523300    123
1631523600    107
1631524200    144
1631524500    208
...

```

Figure 2 - Time series structure for traffic data at a given road location corresponding to a sensor location

The structure can be seen in Figure 2. Each line in the time series has two values being the first one, the timestamp (in seconds) in Epoch format and the second one, the number of vehicles that have gone through that specific location for the last 5 minutes.

This period, 5 minutes or 300 seconds, is the temporal aggregation rate used for this time series. The period can vary from time series to time series and depending on the application. In the case of URBANITE, temporal aggregation rates of 5 and 15 minutes are considered. The 15 minutes aggregation is computed from the values obtained in the 5 minutes aggregation. The computation of time series with a longer temporal aggregation period implies the loss of information. For this reason, data is usually stored with the finer resolution available.

As it is mentioned above, the resolution to be used depends on the specific application but also on the actual data measured. It could be thought that the best would be to always use the finer resolution, but, in practice, using the finer resolution could imply working with a time series with a lot of noise, whereas increasing the temporal aggregation period averages the noise producing a smoother time series. Hence, these two aspects must be balanced, on one hand the time series should have the most possible information, and on the other, with the least possible noise. The periods chosen in URBANITE, 5 and 15 minutes, correspond to values in which the noise and the amount of information are compensated, producing time series that are appropriated to train AI model for prediction.

Another important aspect to mention is that, although we use a 5-minute aggregation rate, that does not imply that we can count on the fact that the sensors will always provide data every 5 minutes. For example, in Figure 2 there is no data available for timestamp 1631523900. This usually occurs when working with real data because there are always situations where the sensor has gone offline due to issues related to the data capturing process, power outages, connectivity problems or other obstacles that do not allow to obtain the correct data.

However, the traffic data is not always provided by the city in an aggregated time series format. For example, in the Helsinki use case, a timestamp is stored every time a vehicle goes through the sensor location. This implies that in order to transform this information into an aggregated time series format, some calculations must be performed. The aggregation component of the Data Management platform is in charge of performing these calculations

periodically. Given an aggregation period and a starting time stamp, a division of the timeline can be performed and the trips that lie in each slot can be added to produce the aggregated time series format.

In certain cases, other aggregations need to be performed, for example, some traffic sensors divide the information depending on the type of vehicle (motorcycle, regular car or long vehicle, i.e., a bus or a truck). In the traffic flow prediction procedure performed in URBANITE, the traffic flow is not distinguished according to the vehicle type. Therefore, vehicle type aggregation needs to be done. Other sorts of aggregations include the aggregation of vehicles measured in different lanes of the road.

2.2.1.2 Bike Data Aggregation

The harvested data related to bikes is the GPS data of the individual bike rentals, in the case of Bilbao and Helsinki use cases, rented using the city's public service. These data are used in URBANITE for the computation of Origin-Destination (OD) matrixes and for the analysis of trajectories. For the first case, some aggregations need to be computed by the Data Management Platform. For the second case, data cleaning processes need to be carried out (see D3.5 for more detailed information about *Cleaning Trajectory Data*).

In the case of the computation of the OD matrixes, a reduced set of data is used and only the initial and end points of each rental are considered (see Figure 3). The data in each row represents a rental where the three first values correspond to the starting point of the trajectory (timestamp in seconds in Epoch format when the rental started, and GPS latitude and longitude coordinates), and the last three values correspond to the end of the trajectory (timestamp and GPS coordinates).

```

...
1631524500 43.265315 -2.913067 1631824230 43.263796 -2.925268
1631522503 43.254280 -2.912948 1631622503 43.251823 -2.927679
1631524523 43.262615 -2.917710 1631534523 43.256140 -2.903499
...

```

Figure 3 - Information related to the start and end points of the bike trajectories

There are two different aggregation processes to be performed in the computation of the OD matrixes: **temporal** and **spatial aggregation**. The temporal aggregation corresponds to the same type of aggregation performed in the case of traffic flow data, i.e. trips are summed up for a given time period. Typically, the aggregation periods for the computation of the OD matrixes are longer than for the prediction of traffic data, being a typical value equal to 1 hour. The reason for choosing such a large value, in comparison to the traffic flow case, is due to the fact that the number of trips is notably lower, and a longer period is needed in order to obtain enough statistics to reduce the noise.

The spatial aggregation implies adding together all the trips within the area of interest. For this purpose, a tessellation of the map of the area of interest, i.e. a division of the map that fills all the zone of interest, needs to be provided.

In the case of the OD matrix computation, two different mechanisms are provided within the URBANITE project: a direct method by means of a geojson object, and by defining a set of points to produce a Voronoi tessellation¹.

The direct method needs a geojson² that represents a “FeatureCollection” object which contains a set of features where the geometry should be of type Polygon. Each of these features should have a property of namezone_id that will be later used to identify each of the areas.

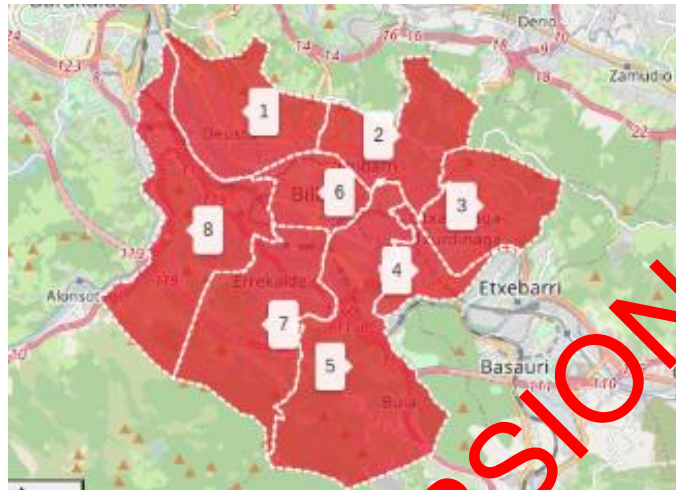


Figure 4 - Visualization of one of the direct tessellations provided in URBANITE for the districts of the city of Bilbao. The number shown for each division corresponds to the value of the zone_id.

Alternatively, in order to generate the Voronoi areas, two inputs need to be provided: a set of points and a rectangle containing all the previous points and that set a limit to all the Voronoi generated areas. These and the resulting areas generated by the tool are shown in Figure 5.

¹ <https://mathworld.wolfram.com/VoronoiDiagram.html>

² <https://geojson.org/>



Figure 5 - Visualization of Voronoi areas tessellation obtained by URBANITE. The inputs for the generation are the limiting rectangle and the set of red points marked in the map.

The resulting Voronoi areas that can be used in the process of the spatial aggregation in the computation of the OD matrix are defined as the set of points that are closer to the given input points. This definition implies that the Voronoi areas are polygons. A possible set of points to define the spatial aggregation area typically is the set of locations where the bicycles can be rented, i.e., the bike stations.

2.2.1.3 Design & Implementation

To optimize queries for large volumes of data, we have opted for the OpenTSDB framework oriented for time series. It is based on a big data database called HBase. The OpenTSDB layer mounted on top of it provides fast access for large queries.

OpenTSDB consists of a Time Series Daemon (TSD). To interact with it, it is necessary to run one or more TSDs. Each TSD uses the open-source database HBase. The data schema is highly optimized for fast aggregations of similar time series to minimize storage space. The communication with the TSD can be via telnet-style protocol or HTTP API. Figure 6 shows the OpenTSDB architecture.

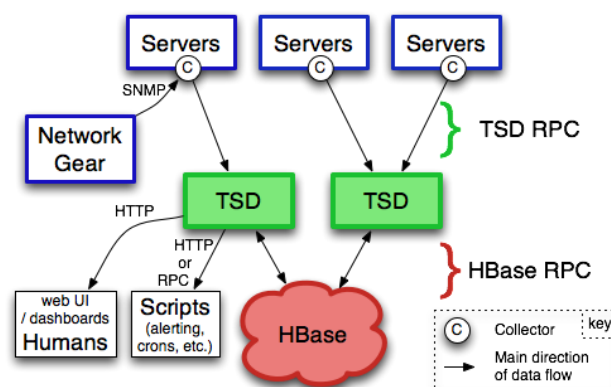


Figure 6 - OpenTSDB architecture.

A powerful feature of OpenTSDB is the ability to perform on-the-fly aggregations of multiple time series into a single set of data points. Aggregation functions are means of merging two or more data points for a single timestamp into a single value. For more information about aggregators, refer to the section aggregators in the appendix “Data aggregation API”.

Besides, OpenTSDB can ingest a large amount of data. Hence queries may return a large number of data points and eat up bandwidth (see Figure 7). Downsampling can be used to reduce the number of data points and therefore, decrease the bandwidth.

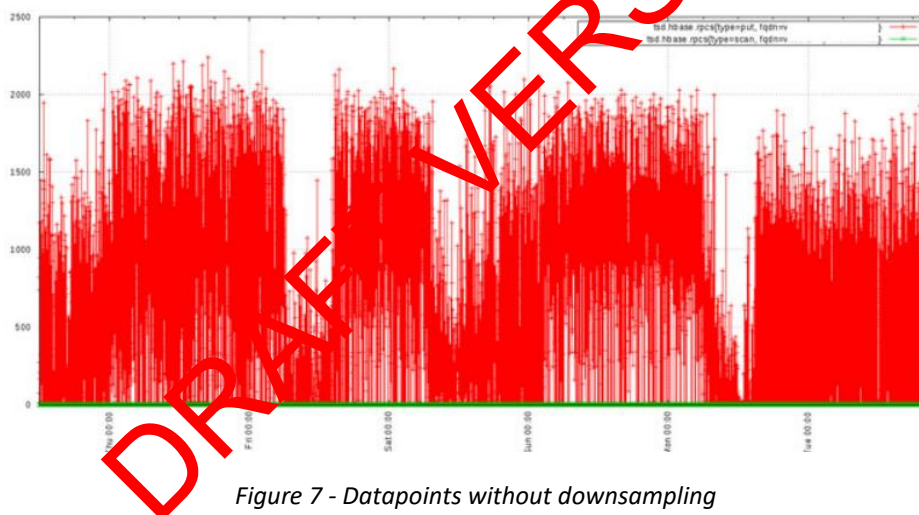


Figure 7 - Datapoints without downsampling

In the image above, we can see a large number of data points. Using downsampling, we can reduce the number of data points with an aggregation function, as we can see in the image below. Downsampling always requires an aggregation function and a time interval.

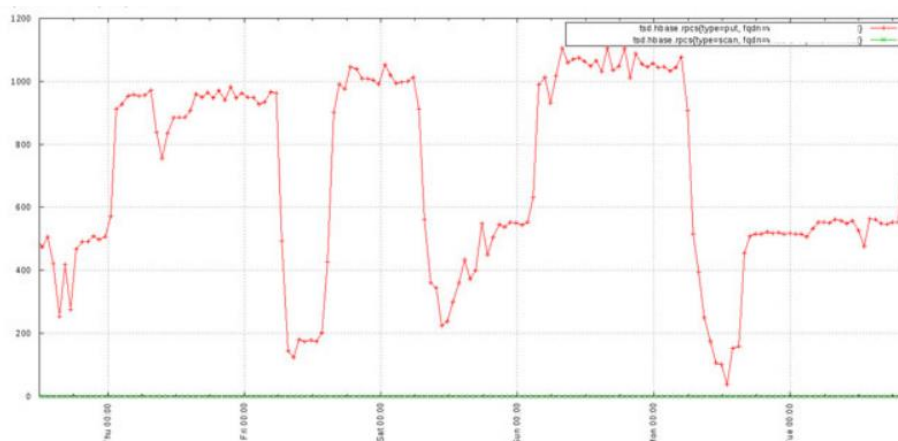


Figure 8 - Datapoints with downsampling

Another feature of OpenTSDB is fast integration with Grafana, which is a tool for real-time metrics visualization through a data source connector.



Figure 9 - Dashboard Visualization in Grafana

To access the data, we have exposed a REST API called “Data Aggregation API”. It contains a ‘get’ resource to obtain the data through the OpenTSDB connection.

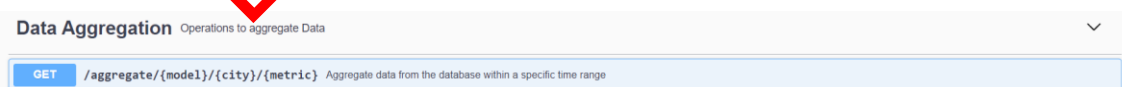


Figure 10 – Get method in the Data Aggregation API.

The complete specification is described in 9 APPENDIX: Data Aggregation API.

2.2.2 Data Fusion

According to [1] “data fusion techniques combine data from multiple sensors and related information from associated databases to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone.”

The integration of data and knowledge from the same or several different sources is called data fusion. The terms information fusion and data fusion are commonly used as synonyms, but there is a little difference. The term data fusion refers to the integration of raw data, i.e. directly obtained from sensors, whereas the term information fusion is used to refer to the integration of already processed data [2]. In the case of URBANITE, the data management platform deals mostly with information fusion.

The purpose of using data/information fusion is to improve data quality, i.e. to reduce error probability and to have higher reliability of the data being used in the algorithms for analytics and decision-making.

There are different data fusion techniques and the approach to use depends on the intended usage. E.g. images from different types of cameras can be fused to obtain more information, or data from complementary data sources can be fused to create improved datasets.

In URBANITE, information fusion is done on aggregated data. Several of the analytic modules developed within URBANITE project require the different data to be fused to be appropriately consumed by the algorithms. Specifically, the regression methods that are being used by the Data Analysis modules expect the times at which the different values are measured to be aligned. Not only aligned but also the aggregation period needs to be the same for all the fused quantities for the whole process to make sense. Next, we describe some examples of datasets that are fused in URBANITE and for which purpose:

- **Traffic Prediction:** In this case, the four datasets that are fused are traffic counts, weather measurements (temperature and precipitation), festivities information (from calendar dataset) and events (football games and ferry arrivals). The resulting fused data stream has an aggregation period of 15 minutes.
- **Bike OD Matrix:** Here, the data that is fused is the same as in the previous case but instead of traffic counts, bike rental information is used. The final data stream has an aggregation period of 1 hour.
- **Bus OD Matrix:** To calculate these OD matrices, Data Fusion of the ticketing information dataset and the routing information is carried out. In this case, the final algorithm is the trip chaining algorithm instead of regression methods.

2.2.3 Data Storage & Retrieval

The Data Storage & Retrieval component provides the means to store and retrieve datasets, DCAT-AP compliant metadata, and related data. Hence, this component needs to have repositories to store both DCAT-AP compliant metadata and transformed data, as depicted in Figure 11.

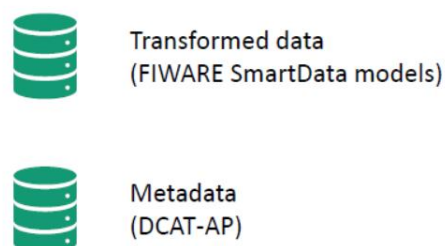


Figure 11 - Data Storage & Retrieval repositories

2.2.3.1 Challenges and architectural design

When deciding about the technologies to use to implement the Data Storage & Retrieval component, several factors were taken into account. One of the main factors to consider was related to the volume of data to be managed, as the number of records to be ingested can be very high in some cases due to the update frequencies (for example, traffic flows can reach around 800.000 monthly registrations). Another important factor was related to the diversity of available data. Although a specific data model is common to several use cases, they may not all have the same information available, making the records, within the same common model, quite heterogeneous. For these cases, the JSON-LD format is quite suitable.

Therefore, it was necessary to design a storage system which could be capable of handling large volumes of data while offering considerable flexibility in terms of the structure of the data, characteristics that fit very well with the NoSQL MongoDB database.

On the other hand, it was also taken into consideration that there may be other types of data sources with information that does not present a high update or a large volume of records, for which solutions such as the MySQL database can be used.

Even so, although these two databases (MongoDB and MySQL) were chosen for the first version, the system is flexible and remains open to adding other databases that may be more appropriate to the needs that may arise in the future or to changing the current ones. As the volume of data increased, especially for traffic flow related data, there was the need to optimize the performance of the queries to retrieve big volumes of data and reduce the response time, so the big data time series database called OpenTSDB was included. This fact demonstrated that the design is open to adding new databases.

However, the existence of several databases is transparent for the end user or for the consumer of data, as the main access point to the storage and retrieval system is through a REST API. This API provides a set of services that are in charge of accessing the most appropriate database, depending on the type of data it is managing, in a totally transparent way to the modules that interact with it.

Besides, the Retrieval API was extended with an OpenDataRetrieval API. The functionality is very similar. In the case of the Retrieval API, it provides access to all data sets in the databases, but it can only be accessed from the internal network where the Data Platform is deployed. On the other hand, the OpenDataRetrieval API provides access only to public or open data and can be accessed from any network. This way, we can guarantee that only authorized users have access to proprietary or sensitive data.

A high-level architecture of this component is shown in Figure 12:

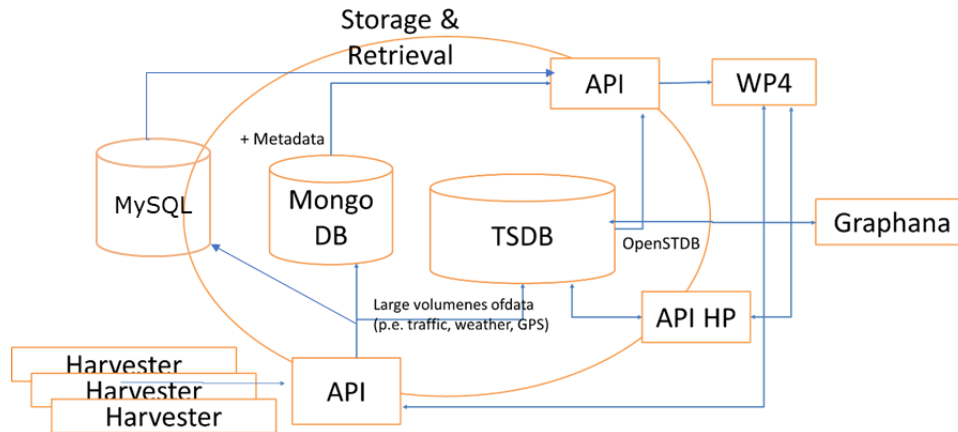


Figure 12 - Technology stack

2.2.3.2 Storage Layer

As mentioned before, the chosen Data Storage system is a combination of SQL (MySQL), NoSQL (MongoDB) and time series (OpenTSDB) databases, to cover all the needs that may arise from the analysis and prediction processes developed in WP4. The data is stored in MongoDB according to the models described in 7 APPENDIX: Data models and formatted in JSON-LD format as key-values.

MongoDB³ is an open source, document-oriented NoSQL database, which means that it stores data in the form of JSON-like documents, thus, it supports arrays and nested objects as values. Being based on documents and Schema Less, the documents within a collection (table) may not have the same fields, thus avoiding having fields with empty or null values that make the size of the database grow unnecessarily.

In addition, MongoDB does not require large computing resources, and it can be used in a decentralized environment in a distributed way. This allows scalability not only vertically (CPU and RAM) but also horizontally (creating more nodes).

On the other hand, MySQL⁴ is one of the world's most popular relational database (RDBMS), and it is based on a client-server model. Among its main characteristics, we can find:

- Low cost in hardware and software requirements for its execution.
- Offers high speed and good performance.
- Capable of handling a large volume of data.
- Ease of installation and configuration, supported in almost 100% of current operating systems.
- High stability and low probability of data corruption.
- Supports security through SSL (Secure Socket Layer) and data encryption.
- Possibility of using different storage mechanisms that offer different operating speeds, physical support, capacity, geographical distribution, transactions ...
- Use of ACID transactions (Atomic, Consistent Isolated, Durable), through commit, rollback, crash recovery and record blocking, and Distributed Transactions.
- Supports replication.

³ <http://www.mongodb.com/>

⁴ <https://www.mysql.com/>

- Supports ANSI SQL⁵
- Ease of data import and export processes.

Finally, all harvested data are stored in MongoDB and timeseries data are also stored in OpenTSDB. Next, we explain how the collections have been organized in MongoDB for each of the data models depending on their volume.

- **Traffic Flow Observed:** the number of records (traffic status measurements) can be too high to store all of them in a single collection. For example, in Bilbao Use Case we can find approximately 800,000 records per month, which would make almost 10 million records per year.

Due to performance issues, and although the logic of storing and retrieving the data is somewhat complicated, the data is divided into one collection for each month and year, so handling a volume of about 800.000 records per collection (Bilbao use case) is fast and efficient.

The names of the collections follow the pattern `trafficflowobserved_<city>_<year>_<month>`. For example:

```
trafficflowobserved_bilbao_2021_08
trafficflowobserved_helsinki_2021_05
...
```

To speed up data queries, collections also have an index on the `dateObserved` field, in descending order, in addition to the index that MongoDB generates on the unique record identifier.

- **Air Quality Observed:** For this data model, the number of records per use case is estimated to be around 50.000 per month, so a volume of close to 600.000 records per year allows us to have a single collection per year and use case.

The names of the collections follow the pattern `airqualityobserved_<city>_<year>`. E.g.:

```
airqualityobserved_bilbao_2021
airqualityobserved_messina_2020
airqualityobserved_amsterdam_2022
...
```

To speed up data queries, collections also have an index on the `dateObserved` field, in descending order, in addition to the index that MongoDB generates on the unique record identifier.

- **Day Specification:** In this case, there should be one record for each day of a year, so the total number of records is small enough to be able to have all the data in a single collection per use case, without compromising performance.

The collections are named `dayspecification_<city>`:

```
dayspecification_amsterdam
```

⁵ https://docs.oracle.com/database/121/SQLRF/ap_standard_sql001.htm

dayspecification_bilbao
dayspecification_helsinki
dayspecification_messina

- **Calendar:** For this data model, there should be one record for each year, so all the data will be stored in just one collection per use case. The collections are named calendar_<city>:

calendar_amsterdam
calendar_bilbao
calendar_helsinki
calendar_messina

- **Event:** To store relevant events in the city that can have an important impact on traffic prediction. E.g. in Bilbao, the model is used to store the calendar of football matches. In Helsinki, the model is used to store information about ferry arrivals and departures. In general, there should be few records per year but for clarity and to facilitate queries we have decided to have a dataset per year. The event model is used and the collections' ids follow the pattern: event_<city>_<year>. E.g.:

event_bilbao_2014
event_helsinki_2022

- **GtfsShape:** Model to store geographic areas. Normally, this dataset should have a low number of records, so we store the data in a single dataset. In the case of Bilbao, 8 records are stored for the districts and 21 for the Wi-Fi zones. In the case of Amsterdam, there are 14 regions in the North Neighborhood. In Messina, there are 6 districts. The collection id follows the pattern: gtfsshape_<city>. E.g.:

gtfsshape_bilbao
gtfsshape_amsterdam
gtfsshape_messina

- **TouristTrip:** Model to store itineraries given a set of waypoints. For this data model, the number of records per use case is estimated to be around 50.000 per month, so a volume close to 600.000 records per year.

The names of the collections follow the pattern touristtrip_<city>_<item>_<year>_<month>. E.g.:

touristtrip_bilbao_bikes_2018_10
touristtrip_bilbao_bikes_2021_02
touristtrip_helsinki_bikes_2017_05

- **TransportStations:** Model to store the Public Transport Stations. Normally, this dataset should have a low number of records, so we store the data in a single dataset. In the case of Bilbao 40 records are saved for Bilbao's public bike stations. In the case of Helsinki we have 457 records of bike stations. In the case of Messina there are 998

records for tram and bus stations. In the case of Amsterdam, there are 244 records for tram and metro stations. The collection's id follows the pattern: transportstation_<city>. E.g.:

transportstation_bilbao
transportstation_helsinki
transportstation_messina
transportstation_amsterdam

- **PointOfInterest:** Model to store the points of interest. Normally, this dataset should have at most a few thousands of records, so we store the data in a single dataset. In the case of Bilbao, 40 records are saved for Bilbao's public bike stations. In the case of Messina, 1153 records of different categories are available. In the case of Amsterdam, we have 1430 records for sport locations and schools (primary and secondary education). The collection's id follows the pattern: pointofinterest_<city>. E.g.:

pointofinterest_bilbao
pointofinterest_messina
pointofinterest_amsterdam

- **OriginDestinationMatrix:** Model to store an Origin Destination Matrix. Normally this dataset should have at most a hundred of records. These matrices are calculated with the modules developed in WP4. The collection's id follows the pattern: origendestinationmatrix_<city>. E.g.:

origendestinationmatrix_bilbao

- **CensusObserved:** Model to store household related information from Eurostat data on European Union Statistics on Income and Living Conditions (EU-SILC). The collection's id follows the pattern: censusobserved_<city>. E.g.:

censusobserved_bilbao

- **PopulationObserved:** Model to store population statistics by age, gender, districts, income and employment. The collection's id follows the pattern: populationobserved_<city>. E.g.:

populationobserved_messina

- **ElectroMagneticObserved:** The data model is intended to measure excessive electric and magnetic fields (EMFs), or radiation in a work or public environment according to the level of exposure to electromagnetic fields on the air. The frequency of the hertzian waves is conventionally lower than 300 GHz, propagating in space without artificial guide. Messina has historical data from 2019.

electromagneticobserved_messina_2019

- **NoiseLevelObserved:** model to store an observation of those acoustic parameters that estimate noise pressure levels at a certain place and time. Messina has historical data from 2018 and 2019:

noiselevelobserved_messina_2018

noiselevelobserved_messina_2019

- **MapLayer:** model to store map layer (GeoJSON files) of the use cases. Scripts that run on AirFlow store the produced map layer on the Data Storage. A section of the UI allows retrieving the available map layers and presents them to the user, who can select the ones to be displayed on the map and access their description.
- **Metadata:** All metadata information is stored in a single collection named *metadata*, since we estimate that the number of records to be handled is not excessively large, so there is no need to divide them into different collections. Storing them in a single collection also facilitates operations on the metadata, both the insertion and the search by tags. To perform these searches by tags in the text fields, a text index⁶ has been created on the collection.

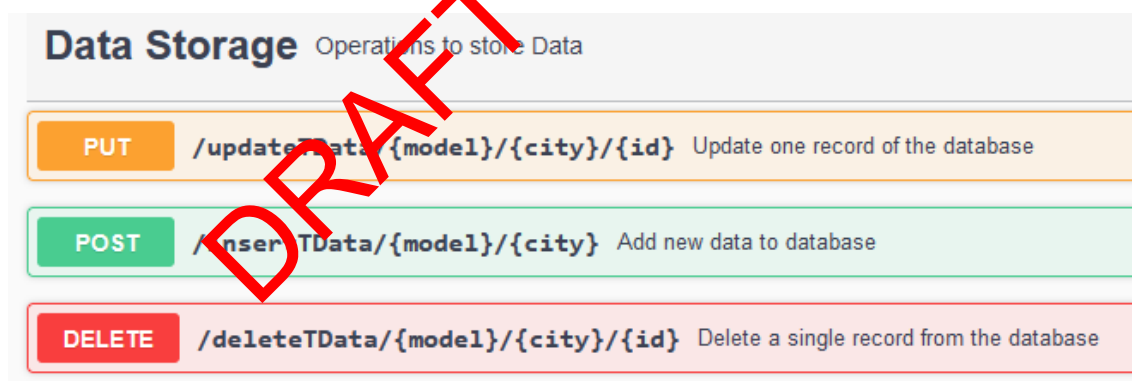
2.2.3.3 Access Layer

The access layer contains a REST API with predefined methods for inserting or accessing data and metadata and a JDBC connection, through the Presto software, to the different databases.

2.2.3.3.1 API

The API is the main access point to the datasets and metadata, providing methods to store and retrieve both of them. It offers a set of REST Web Services, so all the accesses are made through HTTP/HTTPS requests.

This section provides a list of these methods, divided into storage, retrieval, and metadata related REST services. The complete description (method, input and output parameters, etc.) is described in 8 APPENDIX: Storage & Retrieval API.



Data Storage Operations to store Data	
PUT	<code>/updateData/{model}/{city}/{id}</code> Update one record of the database
POST	<code>/insertData/{model}/{city}</code> Add new data to database
DELETE	<code>/deleteData/{model}/{city}/{id}</code> Delete a single record from the database

Figure 13 – Methods in the Data Storage API to insert, delete and update data

⁶ <https://docs.mongodb.com/manual/core/index-text/>

⁷ <https://urbanite.esilab.org:8443/data/swagger-ui/index.html?configUrl=/data/v3/api-docs/swagger-config>

Data Retrieval Operations to retrieve Data	
GET	<code>/getTData/{model}/{city}/{id}</code> Get a record from the database
GET	<code>/getTData/{model}/{city}</code> Get data from the database
GET	<code>/getTDataRange/{model}/{city}</code> Get data from the database within a specific time range
GET	<code>/getSupportedDataModels</code> Get available Data Models
GET	<code>/getDistinct/{model}/{city}</code> Get the different values for a specific field.

Figure 14 - Methods in the Data Retrieval API to retrieve data

Metadata Operations to store and retrieve metadata	
PUT	<code>/dataset</code> Add new metadata of a dataset or update an existing one
DELETE	<code>/dataset</code> Delete the metadata of a dataset
GET	<code>/getCatalogueDatasets</code> Get the metadata of all datasets.
GET	<code>/searchDatasets</code> Searches among the metadata of the existing dataset
GET	<code>/getDataset</code> Get metadata of a dataset

Figure 15 - Methods in the Data Storage API to manage metadata

2.2.3.4 Technical specifications

The Data Storage & Retrieval components have been developed in Java using the Spring Framework⁸ and Spring Boot⁹.

2.2.4 Data Catalogue

The Data Catalogue component provides access to the metadata of the different data sources and their data using a federated approach. In particular, the Data Catalogue component is based on Idra¹⁰, which is a platform able to aggregate Open Data Management Systems (ODMS) based on different technologies providing a unique access point. It harmonizes the metadata coming from federated data sources following DCAT-AP standard.

⁸ <https://spring.io/>

⁹ <https://spring.io/projects/spring-boot>

¹⁰ <https://idra.eng.it>

More specifically, the Data Catalogue provides functionalities to discover and access the datasets' metadata collected and managed by the components of URBANITE Ecosystem for data acquisition, aggregation and storage, such as the Data Storage & Retrieval component.

The Data Catalogue makes use of a specific connector, developed ad hoc for URBANITE ecosystem, to interact with the Data Storage & Retrieval. Following the guidelines for the development of Idra connectors, the new connector implements the operations to allow the interaction between the Data Catalogue and the Data Storage & Retrieval.

The URBANITE user interface already integrates the Data Catalogue user interface as depicted in Figure 16. In this schema is underlined also the usage of the Data Storage & Retrieval connector between the Data Catalogue and the Data Storage & Retrieval components (a part of other available connector types, e.g. CKAN, DKAN etc.).

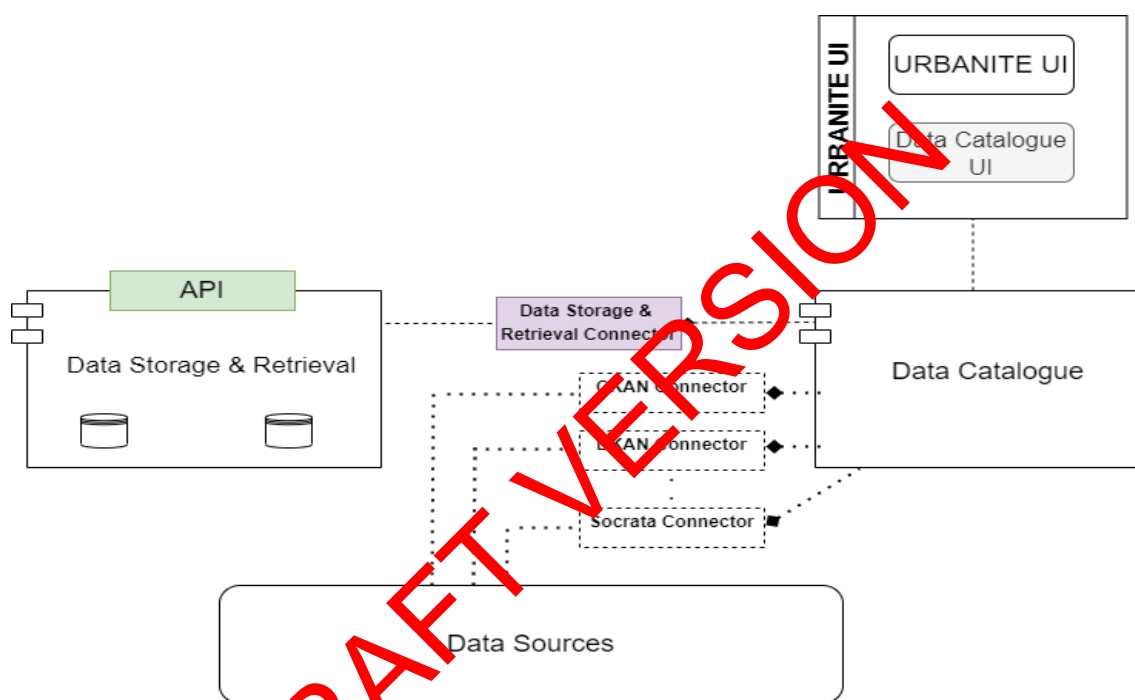


Figure 16 - Schema about URBANITE components (UI, Data Catalogue and connectors, DS&R)

A connector is the basic building block used by Idra to interact and harmonize, following DCAT-AP standard, the metadata of the datasets managed by the federated Open Data Management System. Within Idra, a specific connector is provided for any of the supported data sources typologies (indeed, Idra already includes connectors to federate ODMS based on CKAN, DKAN, Socrata, etc.).

The newly created connector allows to access and manage the datasets provided by the Data Storage & Retrieval by exploiting its exposed Rest API `"/data/getCatalogueDatasets"`.

The datasets' metadata returned by this API are finally added to Idra and made available for all functionalities provided by the tool for searching (using Idra APIs). More technical details about the Data Catalogue are reported in the deliverable D5.4.

3 Data Storage & Retrieval & Aggregation -delivery and usage

3.1 Installation instructions

In order to integrate well into the URBANITE platform, all components are available as Docker¹¹ images. However, before building the Docker images, the corresponding JAR files need to be created. A JAR file is an executable that runs on the JVM.

The storage and retrieval and the aggregation components are composed of two Docker groups:

➤ APIs

The APIs of the three components (storage, retrieval, aggregation) are bundled together and rely on a build tool called Maven for dependency management and generation of the JARs. As such, the deployment of a service can be achieved using the commands below. Note that curly brackets indicate that applicable values need to be substituted.

```
$>mvn clean package
```

```
$> docker build -t urbanite/dataStorage .
```

```
$> docker run -p {PORT}:80 urbanite/dataStorage
```

For this docker component, a certain configuration is needed to be applied using environment variables, in this case, the setup parameters to connect to MongoDB and to OpenTSDB. If any of these environment variables are not present, the default values will be used:

VARIABLE	DESCRIPTION	DEFAULT VALUE
MONGO_HOST	Host where MongoDB is installed	Mongoddb
MONGO_PORT	Port where MongoDB is listening	27017
MONGO_DBNAME	Name of the MongoDB Database to insert or retrieve data	urbanite
OPENTSDB_URL	Url where is deployed the database OpenTSDB	http://opentsdb:4242

These environment variables can be passed to Docker containers, for example:

```
$>docker run -it -p 80:80 -e MONGO_HOST=172.26.41.138 -e MONGO_PORT=27018 -e OPENTSDB_URL=http://opentsdb_4242 urbanite/datastorage
```

➤ Databases

The APIs of the 3 components (storage, retrieval and aggregation) have to connect to the different databases (MongoDB, MySQL and OpenTSDB). A single Docker-Compose¹² configuration file will be used to create all the images at once.

¹¹ <https://www.docker.com/>

For this docker component, it is also necessary to provide a previous configuration. Also, a user with access to MySQL database is needed. In the creation of MySQL Docker image, these variables will be set:

VARIABLE		DESCRIPTION
MYSQL_ROOT_PASSWORD		The password for ROOT user, to create this user in MYSQL configuration
MYSQL_USER		A user to be use
MYSQL_PASSWORD		The password for MYSQL_USER

Below is the docker-compose file:

```
---
# brings up the dependencies
version: '2'
services:

mysql:
  container_name: urbanite_mysql
  image: mysql:8.0.24
  environment:
    MYSQL_ROOT_PASSWORD: '{*****}'
    MYSQL_USER: 'presto'
    MYSQL_PASSWORD: '{*****}'
  ports:
    - "3306:3306"
  volumes:
    - /opt/mysql_data:/var/lib/mysql

mongodb:
  container_name: urbanite_mongodb
  image: mongo:4.0.24
  ports:
    - "27017:27017"
  volumes:
    - /opt/mongo_data:/data/db

opentsdb:
  image: petergrace/opentsdb-docker:latest
  restart: always
  volumes:
    - hbase_data:/data/hbase
    - tsdb_tmp:/tmp
  command:
    - /bin/bash
    - --
    - echo "tsd.storage.fix_duplicates = true" >>
    /etc/opentsdb/opentsdb.conf.sample ) && ( echo "tsd.http.query.allow_delete =
true" >> /etc/opentsdb/opentsdb.conf.sample ) && ( echo "tsd.query.timeout =
80000" >> /etc/opentsdb/opentsdb.conf.sample ) && /entrypoint.sh
```

This will:

- Create a docker container for MySQL, from an official image, with a user and running at port 3306.
- Create a docker container for MongoDB, from an official image, and running at port 27017.
- Create a docker container for OpenTSDB, from an official image, and running at port 4242.

¹² <https://docs.docker.com/compose/>

3.2 User Manual

The only part of the component that a user can interact with is the API, a REST Web Service. This API is developed following the OpenAPISpecification¹³ and offers a human-readable version¹⁴.

This page shows the services that are implemented, giving information about the access URLs, parameters, return codes and values.

Also, a JSON file with the full specification is available¹⁵.

3.3 Licensing information

The software is licensed under Affero General Public License (AGPL) version 3¹⁶.

3.4 Download

All source code resides in the GitLab maintained by Tecnalia¹⁷.

4 Data Catalogue -delivery and usage

4.1 Installation instructions

This section covers the steps needed to properly install the Data Catalogue. As described before, the Data Catalogue is an extension of Idra which is an Open Data Federation Platform developed as a Java EE (Enterprise Edition) application. This tool can be installed through Docker¹⁸. The installation instructions are detailed in the *Installation overview* section of the online manual. The detailed instruction to install or use the administration functionalities of Idra can also be found at the corresponding section on *Read The Docs*¹⁹.

4.2 User Manual

The Data Catalogue provides a set of Restful APIs to interact with the IDRA tool and its functionalities. These API are developed following the OpenAPI specification and in particular, Apiary²⁰. The official APIs are available on this official link²¹. These APIs are grouped into *Administration APIs*, *End User APIs* and *Federation APIs*. Further information about Idra APIs is reported in the deliverable D5.4 and available in the Idra documentation section²².

The Data Catalogue offers the functionalities to discover and access the datasets collected, managed and produced by the components of URBANITE Ecosystem for data acquisition, aggregation and storage. The Data Catalogue provides these main functionalities to let to the *administrator user* to *manage catalogues federation* and to *manage configuration* as depicted in Figure 17 and in Figure 18.

¹³ <https://swagger.io/specification/>

¹⁴ [http://\[server:port\]/data/swagger-ui/index.html?configUrl=/data/v3/api-docs/swagger-config](http://[server:port]/data/swagger-ui/index.html?configUrl=/data/v3/api-docs/swagger-config)

¹⁵ [https://\[server:port\]/data/v3/api-docs](https://[server:port]/data/v3/api-docs)

¹⁶ <https://www.gnu.org/licenses/agpl-3.0.en.html>

¹⁷ https://git.code.tecnalia.com/urbanite/public/-/tree/main/data_management_platform

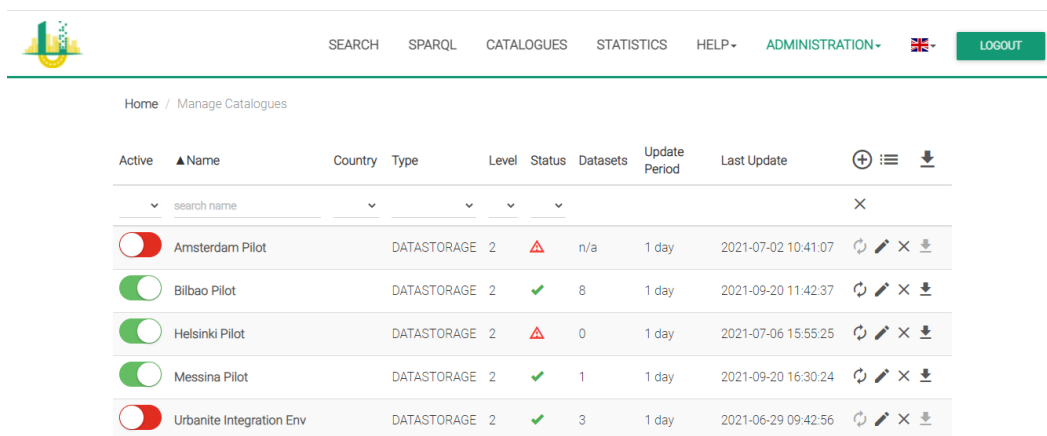
¹⁸ Install Idra on Docker - https://idra.readthedocs.io/en/latest/admin/install_docker/

¹⁹ Idra Installation - <https://idra.readthedocs.io/en/latest/admin/installation/>

²⁰ Apiary - <https://apiary.io/>

²¹ API description document - <https://idraopendata.docs.apiary.io/api-description-document>

²² Open API Idra - <https://idraopendata.docs.apiary.io/#>



Active	Name	Country	Type	Level	Status	Datasets	Update Period	Last Update	
<input type="checkbox"/>	Amsterdam Pilot		DASTORAGE	2		n/a	1 day	2021-07-02 10:41:07	
<input checked="" type="checkbox"/>	Bilbao Pilot		DASTORAGE	2		8	1 day	2021-09-20 11:42:37	
<input checked="" type="checkbox"/>	Helsinki Pilot		DASTORAGE	2		0	1 day	2021-07-06 15:55:25	
<input checked="" type="checkbox"/>	Messina Pilot		DASTORAGE	2		1	1 day	2021-09-20 16:30:24	
<input type="checkbox"/>	Urbanite Integration Env		DASTORAGE	2		3	1 day	2021-06-29 09:42:56	

Figure 17 - Data Catalogue management of federated ODMS

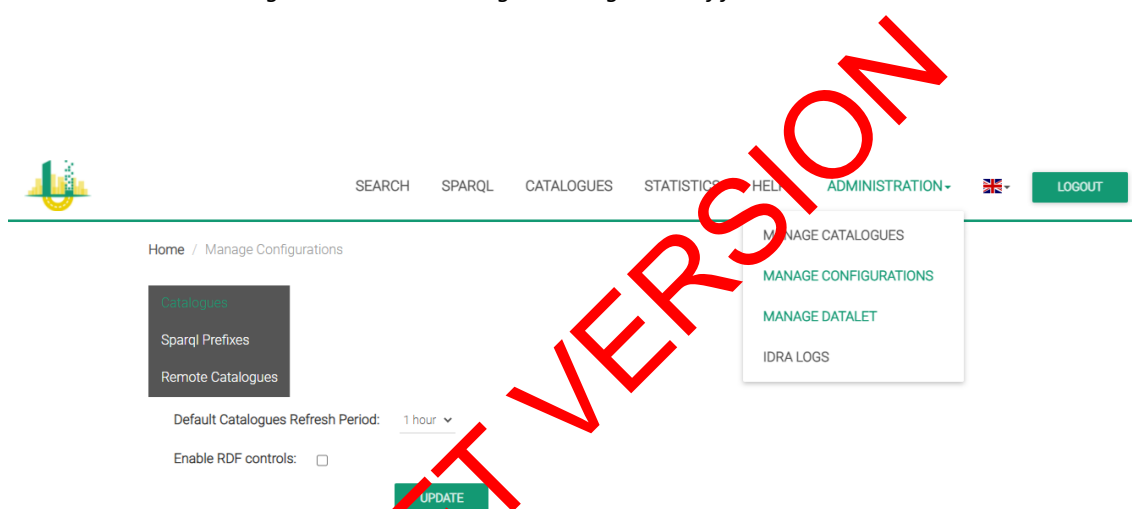


Figure 18 - Data Catalogue configuration management

The Data Catalogue also provides the *end users* the main functionalities to perform federated *metadata search* among federated catalogues (Figure 19).

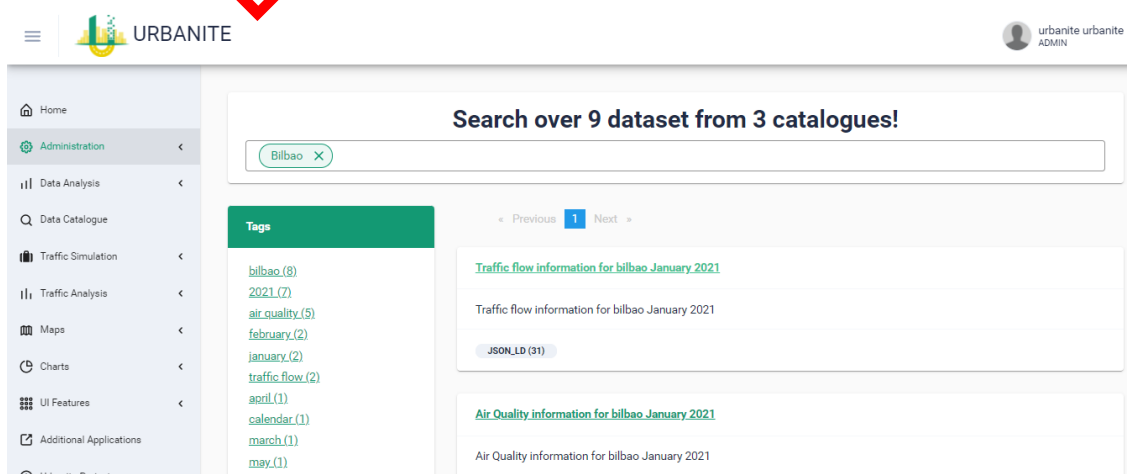


Figure 19 - Data Catalogue federated metadata search by tag

The Data Catalogue metadata search can filter the data using a facet approach. In particular, the following facets are available to filter on dataset metadata results by Tags, Formats, Licenses, etc., as depicted in Figure 20.

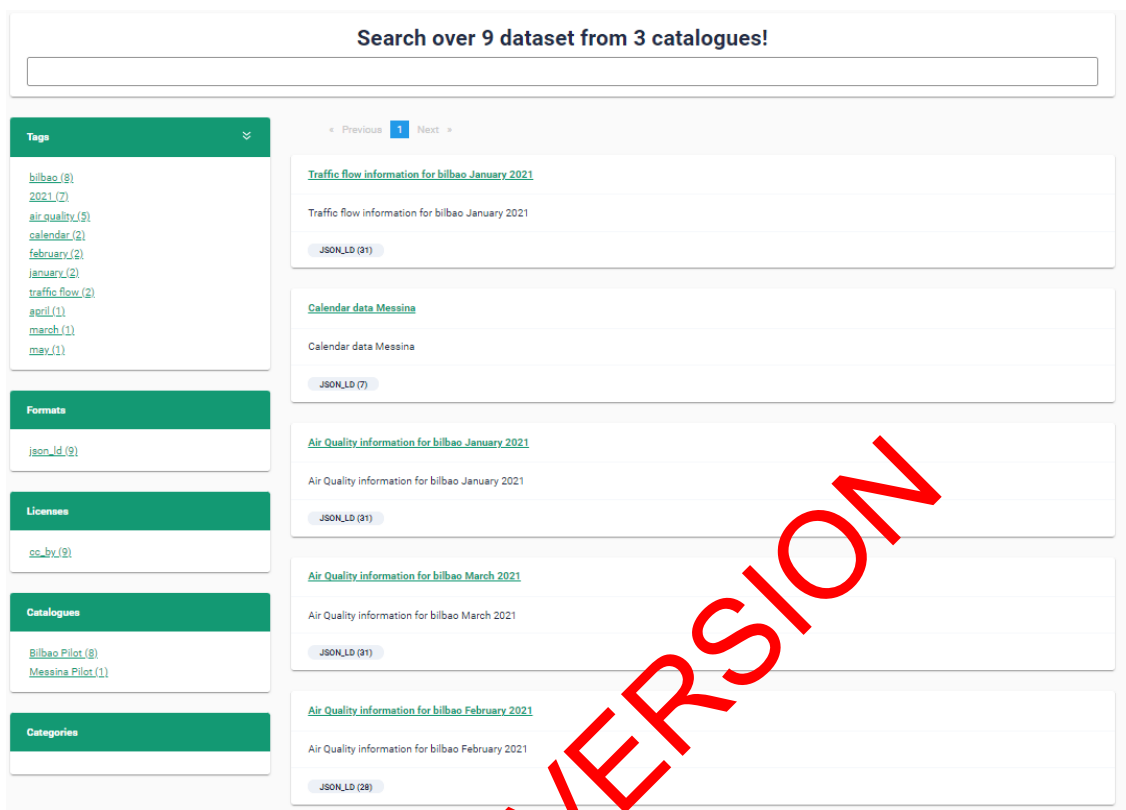


Figure 20 - Data Catalogue federated metadata search

The Data Catalogue metadata search page shows all metadata results, as default visualization, without any filtering. To perform a search, the user can insert one or more keywords into the search bar, as depicted in Figure 19.

Moreover, as previously reports, the user can filter the obtained results by selecting a tab, license, etc., reported in the panels located on the left (Figure 20).

The results are reported in a list of datasets matching with the selected filter and the submitted keywords; for each result the following information is reported: *title*, *description* and *all available distributions*.

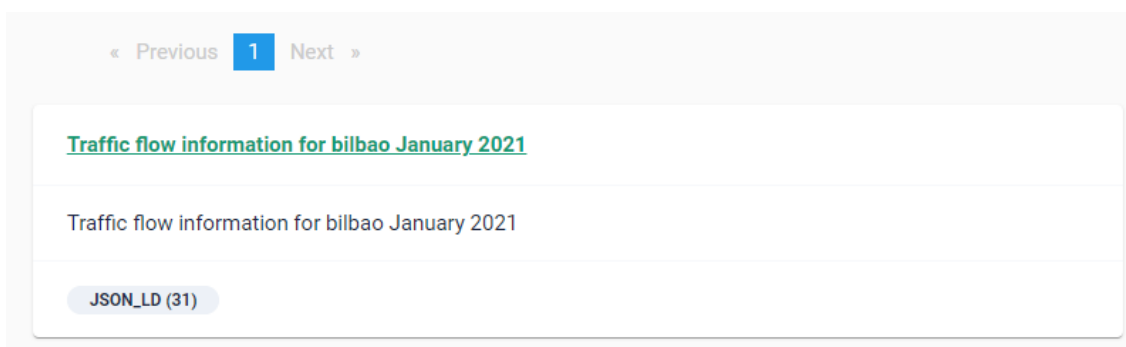


Figure 21 - Information fields of dataset's metadata search list

By clicking on its title, it is possible to access the detailed information of a dataset, as depicted in Figure 20. The detailed information includes the tags associated with the dataset, all the available distributions, the publisher name, the identifier of the dataset, etc.

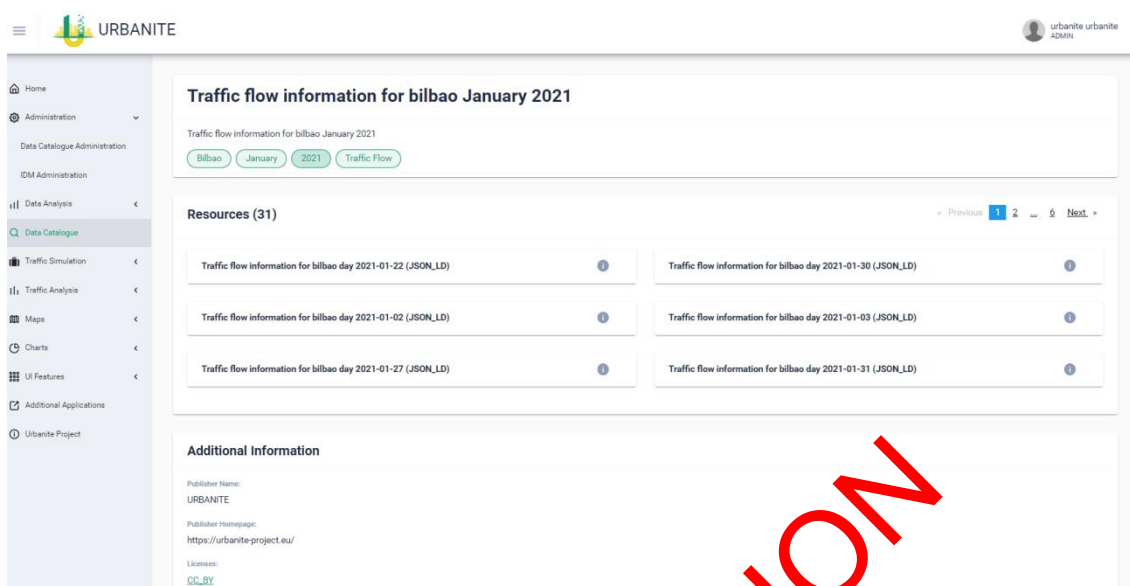


Figure 22 - Data Catalogue federated metadata - dataset detail view

Finally, by clicking on the information icon associated to each distribution, it is possible to access further details (Figure 24): the associated description, the format, the Access URL (for direct access to the distribution), the Download URL (to download the distribution), and the license.

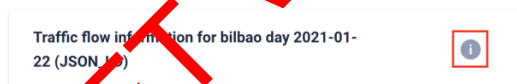


Figure 23 - Data Catalogue- Distribution - Information Icon



Figure 24 - Data Catalogue -details of a distribution

4.3 Licensing information

The Data Catalogue is licensed under Affero General Public License (AGPL) version 3. For further information, read license²³ section on official Github of the project.

4.4 Download

Detail about this extension and about this new connector is provided in the Data Catalogue section of this same document. The source code of the Data Catalogue is available on Github²⁴.

DRAFT VERSION

²³IDRA License - <https://github.com/OPSILab/Idra#license>

²⁴Data Catalogue Github - <https://git.code.tecnalia.com/urbanite/private/wp3-data-management/urbanite-data-catalogue-src>

5 Conclusions

This document describes the technical details of the components associated with the data fusion, aggregation, storage and retrieval integrated on the URBANITE Data Management Platform. The last three components are offered through APIs, whereas the Data Fusion techniques are implemented inside the code of the WP4 algorithms. Data Fusion is referred to the integration of multiple raw data from the same or several different sources gathered from sensors is called data fusion. Data Aggregation is the process of gathering data and presenting it summarized or anonymized. Finally, on a hand, the Data Storage & Retrieval capacities allow storing and retrieving datasets and associated metadata; on the other hand, the functionalities provided by the Data Catalogue allow to discover the datasets managed by the Data Storage & Retrieval and federate additional data sources, offering a unique point to access datasets coming from scattered sources.

The document provides the updated and final technical description of the components as well as installation instructions. Thus, these components are part of the data processing chain, integrating with those defined in deliverables D3.3 and D3.6, associated with data collection and curation.

DRAFT VERSION

6 References

- [1] D. L. Hall y J. Llinas, «An introduction to multisensor data fusion,» *Proceedings of the IEEE*, vol. 85, nº 1, p. 6–23, 1997.
- [2] F. Castanedo, «A Review of Data Fusion Techniques,» *The Scientific World Journal*, vol. 2013.

DRAFT VERSION

7 APPENDIX: Data models

Deliverable D3.4 described the common data models to be used in URBANITE. This section provides more detailed and updated information about those models and how they are being used by the Data Management Platform components.

Besides, it includes the description of new models that are now supported in v2 in order to meet the requirements of WP4: Event, GtfsShape, TouristTrip, PointOfInterest, TransportStation, OriginDestination Matrix, CensusObserved, PopulationObserved, NoiseLevelObserved and ElectroMagneticObserved .

7.1 Traffic Flow Observed

This model is based on the FIWARE's TrafficFlowObserved²⁵ data model, and contains information about an observation of traffic flow conditions at a certain place and time, such as the number of vehicles detected during the observation, the occupancy of the lanes, the location of the detection device, etc.

The model is very complete and collects a lot of information, but not all of it is of interest for WP4 or available in the cities' data sources. The subset of fields that are harvested are: id, address, averageVehicleSpeed, dateObserved, intensity, location and occupancy.

Example:

```
{
  "id": "urn:ngsi-ld:TrafficFlowObserved:59d4b0d0-5fe3d67a6_-2ca4",
  "address": {
    "addressCountry": "ES",
    "addressLocality": "Bilbao"
  },
  "averageVehicleSpeed": 22.0,
  "dateObserved": "2021-08-01T00:25:00.000",
  "intensity": 190.0,
  "location": {
    "coordinates": [
      [505545.0202764, 4709451.73590806],
      [505554.84865274, 4789477.77248377]
    ],
    "type": "Polygon"
  },
  "occupancy": 0.06,
  "type": "TrafficFlowObserved",
  "@context": [
    "https://smartdatamodels.org/context.jsonld",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ]
}
```

7.2 Air QualityObserved

Based on FIWARE's AirQualityObserved²⁶ data model, it contains information about an observation of air quality conditions at a certain place and time, such as the carbon monoxide, nitrogen dioxide, material particles, temperature, humidity, etc.

The subset of fields that are harvested are: id, dateObserved, location, NO, NO2, NOX, PM10 and SO2.

²⁵<https://github.com/smart-data-models/dataModel.Transportation/tree/master/TrafficFlowObserved>

²⁶ <https://github.com/smart-data-models/dataModel.Environment/tree/master/AirQualityObserved>

Example:

```
{
  "id": "urn:ngsi-ld:AirQualityObserved:62:290820210500",
  "dateObserved": "2021-08-29T05:00:00Z",
  "location": {
    "coordinates": [43.26750551179745, -2.935188110338201],
    "type": "Point"
  },
  "no": 2,
  "no2": 30,
  "nox": 33,
  "pm10": 13,
  "so2": 10,
  "type": "AirQualityObserved",
  "@context": [
    "https://smartdatamodels.org/context.jsonld",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ]
}
```

7.3 WeatherObserved

This model is based on FIWARE's WeatherObserved²⁷ data model, containing information about an observation of weather conditions at a certain place and time, such as precipitation, humidity, temperature, UV index, etc.

The subset of fields that are harvested are: id, dateObserved, location, atmosphericPressure, precipitation, relativeHumidity, temperature, windDirection and windspeed.

Example:

```
{
  "id": "urn:ngsi-ld:WeatherObserved:Brilhad-2021-06-30T07:00:00.00Z",
  "type": "WeatherObserved",
  "dateObserved": "2021-06-30T07:00:00.00Z",
  "temperature": 13.3,
  "precipitation": 0,
  "atmosphericPressure": 1024,
  "location": {
    "type": "Point",
    "coordinates": [-2.93419717987, 43.2641971992]
  },
  "relativeHumidity": 75,
  "windDirection": 120,
  "windSpeed": 10,
  "@context": [
    "https://smartdatamodels.org/context.jsonld",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ]
}
```

7.4 Calendar and Day Specification

These two models have been defined within the scope of the URBANITE project to collect information on city calendars. The daySpecification model collects information about a specific day, such as whether it is a holiday, school day, the day of the week, etc. The Calendar model collects information for a whole year, including a list of references (ID) of each of its days (daySpecification).

²⁷ <https://smart-data-models.github.io/dataModel.Weather/WeatherObserved/doc/spec.md>

7.4.1 DaySpecification

Table 2: Structure for day specification

Field	Description	Type	Mandatory
id	Unique identifier of the entity	Text	Y
type	Entity type	Text, must be 'DaySpecification'	Y
date	The date of this entity	Text in ISO8601 format	Y
description	A description of this item	Text	N
workingDay	If it is a working day (1) or not (0)	Numeric (0 or 1)	Y
schoolDay	If it is a school day (1) or not (0)	Numeric (0 or 1)	Y
publicHoliday	If it is a holiday (1) or not (0)	Numeric (0 or 1)	Y
weekDay	Day of the week	Numeric (1 to 7)	Y

Example

```
{
  "id": "urn:ngsi-ld:DaySpecification:Bilbao_2015_01_01",
  "type": "DaySpecification",
  "date": "2015-01-01",
  "description": "Año nuevo",
  "workingDay": 0,
  "schoolDay": 0,
  "publicHoliday": 3,
  "weekDay": 4,
  "createdAt": "2021-06-17T08:24:53.376Z",
  "modifiedAt": "2021-06-17T08:24:53.376Z",
  "@context": [
    "https://git.code.tecnalia.com/urbanite/public/-/raw/master/datamodels/calendar-ngsi.jsonld"
  ]
}
```

7.4.2 Calendar

Table 3: Structure for a calendar specification

Field	Description	Type	Mandatory
id	Unique identifier of the entity	Text	Y
type	Entity type	Text, must be 'Calendar'	Y
city	City of the calendar	Text	N
location	Location of the item	Geojson	N
year	Year of the calendar	Numeric	Y
days	Days that conform the calendar	List of Text	Y

Example (reduced to just 6 days, it could contain a whole year):

```
{
  "id": "urn:ngsi-ld:Calendar:Bilbao:2021",
  "type": "Calendar",
  "city": "Bilbao",
  "location": {
    "coordinates": [-2.93609619140625, 43.26345626603949],
    "type": "Point"
  },
  "year": 2021,
  "days": [
    "urn:ngsi-ld:DaySpecification:Bilbao:2021_01_01",
    "urn:ngsi-ld:DaySpecification:Bilbao:2021_01_02",
    "urn:ngsi-ld:DaySpecification:Bilbao:2021_01_03",
    "urn:ngsi-ld:DaySpecification:Bilbao:2021_01_04",
    "urn:ngsi-ld:DaySpecification:Bilbao:2021_01_05",
    "urn:ngsi-ld:DaySpecification:Bilbao:2021_01_06",
  ],
  "@context": [
    "https://git.code.tecnalia.com/urbanite/public/-/raw/master/datamodels/calendar-ngsi.jsonld"
  ]
}
```

7.5 Event

This model is based on FIWARE's Event²⁸ data model and is used to store relevant events in the city that can have an important impact on traffic prediction. E.g. football matches or ferry arrivals and departures. The type of event is stored in the category field.

Example:

```
{
  "_id": "urn:ngsi-ld:event:helsinki:ferry:VuosaariMuugacheck-in:1641582900000",
  "category": "ferry_arrival_departure",
  "startDate": {
    "$date": "2022-01-07T19:15:00.000Z"
  },
  "type": "Event",
  "passengerCount": 300,
  "departsFromHarbour": "Port of Helsinki",
  "arrivesToHarbour": "Muuga",
  "ship": "Finbo Cargo",
  "terminal": "Vuosaari Muuga check-in",
  "routeType": "ferry",
  "subCategory": "DEPARTURE",
  "@context": [
    "https://smart-data-models.org/context.jsonld",
    "https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/event-ngsi.jsonld"
  ],
  "dateCreated": {
    "$date": "2021-12-18T00:00:05.920Z"
  },
  "dateModified": {
    "$date": "2021-12-24T00:00:01.555Z"
  }
},
{
  "_id": "urn:ngsi-ld:event:bilbao:football:1644267600000",
  "category": "football_match",
  "championship": "Primera División 2021/2022",
  "endDate": {
    "$date": "2022-02-07T22:45:00.000Z"
  }
},
```

²⁸ <https://github.com/smart-data-models/dataModel.TourismDestinations/tree/master/Event>

```

"guestTeam": "Espanyol",
"homeTeam": "Athletic Bilbao",
"location": {
  "coordinates": [
    43.264183,
    -2.949421
  ],
  "type": "Point"
},
"startDate": {
  "$date": "2022-02-07T21:00:00.000Z"
},
"type": "Event",
"passengerCount": 0,
"@context": [
  "https://smartdatamodels.org/context.jsonld",
  "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
],
"dateCreated": {
  "$date": "2022-05-23T12:09:52.540Z"
},
"dateModified": {
  "$date": "2022-09-12T00:00:04.391Z"
}
}

```

7.6 Transport Station

This model is based on FIWARE's Transport Station²⁹ data model, containing information about public transport stations, such as name, station type, location and zone identifier.

Example:

```

{
  "id": "urn:ngsi-ld:TransportStation:Amsterdam:1",
  "location": {
    "coordinates": [
      52.376071,
      4.893344
    ],
    "type": "Point"
  },
  "name": "Nieuwezijds Kolk",
  "stationType": [
    "tram"
  ],
  "type": "Transport Station",
  "zoneId": "2 | 12 | 13 | 17",
  "@context": [
    "https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/urbanite-context.json",
    "https://smartdatamodels.org/context.jsonld",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ],
  "dateCreated": "2022-09-15T13:32:22.723Z",
  "dateModified": "2022-09-15T13:32:22.723Z"
},
{
  "id": "urn:ngsi-ld:TransportStation:Bilbao:Bikes:3360",
  "location": {
    "coordinates": [43.2815, -2.96273],
    "type": "Point"
  },
  "name": "LEVANTE",
  "stationType": ["bike"],

```

²⁹ <https://github.com/smart-data-models/dataModel.Transportation/blob/master/TransportStation>

```

    "type": "TransportStation",
    "zoneId": "3360",
    "@context": ["https://smartdatamodels.org/context.jsonld",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"],
    "dateCreated": "2022-04-13T07:43:40.515Z",
    "dateModified": "2022-04-13T07:43:40.515Z"
  }

```

7.7 Point of Interest

This model is based on FIWARE's PointOfInterest³⁰ data model and stores information about points of interest, i.e., specific point locations that someone may find useful or interesting. For each POI, we store information such as its name, category, location and zone identifier.

Besides, the field “category” should be one of the terms defined in the taxonomy [Categories](#)³¹. For example: bicycles (129), bus stations (426).

Example:

```

[ {
  "id": "urn:ngsi-ld:PointOfInterest:3360",
  "category": "129",
  "location": {
    "coordinates": [43.2815, -2.96273],
    "type": "Point"
  },
  "name": "LEVANTE",
  "type": "PointOfInterest",
  "zoneId": "3360",
  "@context": ["https://smartdatamodels.org/context.jsonld",
  "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"],
  "dateCreated": "2022-04-05T16:16:51.917Z",
  "dateModified": "2022-04-05T16:16:51.917Z"
} ]

```

7.8 GtfsShape

This model is used to store geographic areas. The definition of the model is the one provided by FIWARE's GtfsShape³² data model. For each city, different categories of geographic areas can be stored. E.g. in Bilbao this model is used to store the geographic areas of districts and Wi-Fi zones. Districts are used for traffic counts whereas Wi-Fi zones are used for OD matrices.

To describe the geographic areas under the same category or group we use the “alternateName” field.

- "district" for districts
- "wifi_zone" for wifi zones

In Amsterdam, this model is used to store the geographic areas of:

- "amsterdam_nord_neighborhood" for the neighbourhoods in the North.

³⁰<https://github.com/smart-data-models/dataModel.PointOfInterest/blob/master/PointOfInterest/doc/spec.md>

³¹ https://github.com/Factual/places/blob/master/categories/factual_taxonomy.json

³² <https://github.com/smart-data-models/dataModel.UrbanMobility/blob/master/GtfsShape/doc/spec.md>

Besides, the field “name” contains the id of the geographic area.

Example:

```
[{
  "id": "urn:ngsi-ld:GtfsShape:bilbao:district:1",
  "type": "GtfsShape",
  "alternateName": "district",
  "name": "1",
  "description": "Deusto",
  "@context": ["https://smartdatamodels.org/context.jsonld",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"],
  "location": {
    "type": "Polygon",
    "coordinates": [
      [
        [-2.936220985248524, 43.26990470603399],
        ...
        [-2.940830644867825, 43.26943057334333],
        [-2.940626034390812, 43.26950059938804],
        [-2.939857790823712, 43.2696703515841],
        [-2.939316234715834, 43.26974118794736],
        [-2.938821033034044, 43.26979115997708],
        [-2.938637707562993, 43.26980969764579],
        [-2.938005839277664, 43.26985779218719],
        [-2.937536685039381, 43.26987167141188],
        [-2.936220985248524, 43.26990470603399]
      ]
    ]
  }
}]
```

7.9 Tourist Trip

This model is based on FIWARE’s Tourist Trip³³ data model. A tourist trip represents a created itinerary of visits to one or more places of interest. In URBANITE, this model is mainly used to store bike trajectories. E.g., in Bilbao this model is used to store the 2 waypoints in a bike rental service where a user picks up the bike and where the same user leaves the bike, as well as the time spent.

For example, to retrieve all the bike tourist trips done by users in Bilbao for the 28th of February 2021 we would use the query:

<https://bilbao.urbanite.esilab.org/data/getTDataRange/touristTrip/bilbao?startDate=2021-2-28T00%3A00%3B00.000Z&endDate=2021-2-28T23%3A59%3A00.000Z>

The identifiers used in the field named “position” need to match with those defined in Bilbao’s bikes transport stations³⁴.

Example:

```
[{
  "id": "urn:ngsi:touristTrip:bikes:112156240",
  "duration": 14,
  "endDate": "2021-02-22T12:55:25Z",
  "itinerary": [{
    "name": "3380",
    "position": 1
  }, {
```

³³ <https://github.com/smart-data-models/dataModel.TourismDestinations/tree/master/TouristTrip>

³⁴ <https://bilbao.urbanite.esilab.org/data/getTData/transportStation/bilbao?filters=%7BstationType%3Abike%7D>

```

        "name": "3360",
        "position": 2
    }],
    "owner": ["Bilbao Town Hall"],
    "startDate": "2021-02-22T12:55:25Z",
    "type": "TouristDestination",
    "idTrip": "112156240",
    "customer": "4093730",
    "@context": ["https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/urbanite-context.json",
    "https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"],
    "dateCreated": "2022-04-11T09:59:29.700Z",
    "dateModified": "2022-04-11T15:49:53.373Z"
  }
]

```

7.10 Origin Destination Matrix

This model is used to store an Origin Destination Matrix. As FIWARE does not provide a model for OD matrices, URBANITE has defined one based on NGSI-LD. The definition is available at:

<https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/ODMatrix-ngsi.jsonld>

OD matrices can be calculated for different aggregation periods and for different travel models. Depending on the travel mode, the geographic areas that represent the origin and the destination may vary. This is the case of Bilbao, where there are OD matrices based on bike data and OD matrices (for all travel modes) based on Wi-Fi data. To retrieve each of them, the field "zones" is used and its content should match the name provided in the "alternateName" field of the GtfsShape model.

- "district" for bike OD matrices as departure and arrival IDs correspond to districts id of [Bilbao Districts](https://bilbao.urbanite.es/itab.org/data/getTData/originDestinationMatrix/bilbao?filter=s=%7B%22zones%22%3A%20%22district%22%7D):
<https://bilbao.urbanite.es/itab.org/data/getTData/originDestinationMatrix/bilbao?filter=s=%7B%22zones%22%3A%20%22district%22%7D>
- "wifi_zone" for OD Matrix based on Wi-Fi data as the IDS correspond to Bilbo wifi Zones:
https://bilbao.urbanite.es/itab.org/data/getTData/originDestinationMatrix/bilbao?filter=s=%7Bzones%20%3A%20%22wifi_zone%22%7D
- "amsterdam_nord_neighborhood" for OD Matrix based on bikes data as the IDS correspond to [amsterdam nord neighborhood Zones](#)

Example:

```

[ {
  "id": "urn:ngsi:OriginDestinationMatrix:district_zones:bilbao",
  "aggregationType": "Aggregation period:01:00, Hour from:02:00, Hour to:03:00, Day type:Saturday, Sunday, Bank holiday",
  "type": "OriginDestinationMatrix",
  "category": "district",
  "endDate": "02-01-2021",
  "matrixData": [ {
    "arrivesTo": "1", -> District 1
    "departsFrom": "1", -> District 1
    "volume": 155
  }, {
    "arrivesTo": "2",
    "departsFrom": "1",
    "volume": 858
  }, {
    "arrivesTo": "3",
    "departsFrom": "1",
    "volume": 427
  }
]

```

```

    }, {
      "arrivesTo": "9",
      "departsFrom": "2",
      "volume": 366
    }, {
      "arrivesTo": "1",
      "departsFrom": "3",
      "volume": 308
    }, {
      "arrivesTo": "2",
      "departsFrom": "3",
      "volume": 70
    }, {
      "arrivesTo": "1",
      "departsFrom": "7",
      "volume": 100
    }
  ],
  "startDate": "02-01-2021",
  "travelMode": "Bikes",
  "zones": "Bilbao_districts",
  "@context": [
    "https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/urbanite-context.json",
    "https://smartdatamodels.org/context.jsonld",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ],
  "createdAt": "2022-06-22T09:10:52.172Z",
  "modifiedAt": "2022-06-22T09:10:52.172Z"
}]

```

7.11 Population and household models

7.11.1 CensusObserved

This model is a new NGSI-LD compliant model defined in URBANITE to store household related information. Most of the field names are taken from the Eurostat sample data on European Union Statistics on Income and Living Conditions (EU-SILC)³⁵. However, the labels have been modified with more readable names (e.g. householdID instead of db030)³⁶.

The context is defined in:

<https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/census-ngsi.jsonld>

Example:

```

{
  "id": "urn:ngsi-ld:CensusObserved:amsterdam:2016-11-30T07:00:00.00Z",
  "type": "CensusObserved",
  "dateObserved": "2021-11-11T07:00:00.00Z",
  "createdAt": "2021-12-01T13:39:10.00Z",
  "modifiedAt": "2021-12-01T13:39:10.00Z",
  "source": "https://ec.europa.eu/eurostat/",
  "location": {
    "type": "Point",
    "coordinates": [
      -4.754444444,
      41.640833333
    ]
  },
  "householdID": 1,
}

```

³⁵ <https://ec.europa.eu/eurostat/web/microdata/european-union-statistics-on-income-and-living-conditions>

³⁶ https://circabc.europa.eu/sd/a/b862932f-2209-450f-a76d-9cfe842936b4/DOCSILC065%20operation%202019_V9.pdf

```

"addressRegion": "Vienna",
"householdCSWeight": 0.0,
"personalCSWeight": 0.0,
"gender": "male",
"currentEconomicStatus": 1,
"nationality": "AT",
"employeeCashIncome": 0.0,
"selfEmploymentLosses": 0.0,
"unemploymentBenefits": 0.0,
"oldAgeBenefits": 0.0,
"survivorBenefits": 0.0,
"sicknessBenefits": 0.0,
"disabilityBenefits": 0.0,
"educationAllowances": 0.0,
"hsize": 2,
"age": 56,
"netIncome": 16999.29
"@context": [
  "https://smartdatamodels.org//context.jsonld",
  "https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/census-ngsi.jsonld"
]
}

```

7.11.2 PopulationObserved

This model is a new NGSI-LD compliant model defined in URBANITE to store population statistics by age, gender, districts, income and employment.

The context is defined in:

<https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/population-ngsi.jsonld>

Example:

```

{
  "id": "id",
  "type": "PopulationObserved",
  "name": "name",
  "population-summary": {
    "city-inhabitants": {
      "type": "number",
      "value": 243262
    },
    "population-density": {
      "type": "number",
      "value": 1128,
      "unit": "inhabitants/km^2"
    },
    "males": {
      "type": "number",
      "value": 116350
    },
    "females": {
      "type": "number",
      "value": 126912
    }
  },
  "population-ages": {
    "0": {
      "M": 895,
      "F": 803
    },
    "65-69": {
      "M": 7157,
      "F": 7846
    },
    "5-9": {
      "M": 5397,
      "F": 5217
    },
    "40-44": {
      "M": 8142,
      "F": 8581
    }
  }
}

```

```

    },
    "45-49": {
      "M": 8743,      "F": 9590
    },
    "20-24": {
      "M": 6596,      "F": 5946
    },
    "25-29": {
      "M": 7034,      "F": 7045
    },
    "90-94": {
      "M": 654,       "F": 1692
    },
    "70-74": {
      "M": 6340,      "F": 7412
    },
    "95-99": {
      "M": 155,       "F": 490
    },
    "75-79": {
      "M": 4574,      "F": 5949
    },
    ">100": {
      "M": 47,        "F": 90
    },
    "50-54": {
      "M": 9077,      "F": 9871
    },
    "55-59": {
      "M": 8724,      "F": 9626
    },
    "30-34": {
      "M": 7041,      "F": 7042
    },
    "35-39": {
      "M": 7227,      "F": 7325
    },
    "1-4": {
      "M": 3833,      "F": 3596
    },
    "10-14": {
      "M": 5684,      "F": 5505
    },
    "15-19": {
      "M": 6222,      "F": 5812
    },
    "80-84": {
      "M": 4574,      "F": 4000
    },
    "85-89": {
      "M": 1933,      "F": 3630
    },
    "60-64": {
      "M": 7789,      "F": 8943
    }
  },
  "districts-summary": {
    "district-I": {
      "residents": {
        "type": "number",
        "value": 22199
      },
      "families": {
        "type": "number",
        "value": 8819
      },
      "males": {
        "type": "number",
        "value": 10805
      },
      "females": {
        "type": "number",
        "value": 11394
      },
      "ages": {

```



```

    "0": {
      "M": 87,      "F": 78
    },
    "65-69": {
      "M": 626,     "F": 639
    },
    "5-9": {
      "M": 511,     "F": 450
    },
    "40-44": {
      "M": 762,     "F": 791
    },
    "45-49": {
      "M": 800,     "F": 922
    },
    "20-24": {
      "M": 681,     "F": 579
    },
    "25-29": {
      "M": 660,     "F": 716
    },
    "90-94": {
      "M": 62,      "F": 135
    },
    "70-74": {
      "M": 549,     "F": 614
    },
    "95-99": {
      "M": 13,      "F": 26
    },
    "75-79": {
      "M": 427,     "F": 510
    },
    ">100": {
      "M": 3,       "F": 9
    },
    "50-54": {
      "M": 875,     "F": 941
    },
    "55-59": {
      "M": 825,     "F": 765
    },
    "30-34": {
      "M": 608,     "F": 633
    },
    "35-39": {
      "M": 649,     "F": 662
    },
    "1-4": {
      "M": 368,     "F": 335
    },
    "10-14": {
      "M": 521,     "F": 506
    },
    "15-19": {
      "M": 579,     "F": 565
    },
    "80-84": {
      "M": 312,     "F": 438
    },
    "85-89": {
      "M": 205,     "F": 324
    },
    "60-64": {
      "M": 675,     "F": 761
    }
  },
  "district-III": {
    ...
  },
  "district-VI": {
    ...
  },
  "district-II": {

```

```

    ...
  },
  "district-IV": {
    ...
  },
  "district-V": {
    ...
  },
  "education-percentages": {
    "illiterates": {
      "type": "percentage",
      "value": 1.1,
      "unit": "%"
    },
    "literate-without-educational-qualifications": {
      "type": "percentage",
      "value": 5,
      "unit": "%"
    },
    "high-school-diploma": {
      "type": "percentage",
      "value": 31.5,
      "unit": "%"
    },
    "phd-or-equivalent-education": {
      "type": "percentage",
      "value": 0.3,
      "unit": "%"
    },
    "elementary-school-license": {
      "type": "percentage",
      "value": 17,
      "unit": "%"
    },
    "secondary-school-certificate": {
      "type": "percentage",
      "value": 33.5,
      "unit": "%"
    }
  },
  "working-people": {
    "level-of-employment": {
      "type": "percentage",
      "value": 39.694,
      "unit": "%"
    },
    "average-income": {
      "type": "number",
      "value": 15755,
      "unit": "€ per person per year"
    },
    "average-retirement-pension": {
      "type": "number",
      "value": 1031.25,
      "unit": "€ per month"
    }
  },
  "@context": [
    "https://smartdatamodels.org/context.jsonld",
    "https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/population-
ngsi.jsonld"
  ]
}

```

7.12 ElectroMagneticObserved

This model is based on FIWARE's ElectroMagneticObserved³⁷ data model. The Data Model is intended to measure excessive electric and magnetic fields (EMFs), or radiation in a work or public environment according to the level of exposure to electromagnetic fields on the air. The frequency of the hertzian waves is conventionally lower than 300 GHz, propagating in space without artificial guide.

In URBANITE, this model is used to store historical data from Messina.

Example:

```
[{
  "_id": "urn:ngsi-ld:electromagneticNoise:messina:tribunale:1546300861000",
  "dateObserved": {
    "$date": "2019-01-01T00:01:01.000Z"
  },
  "description": "The eMF value refers to electromagnetic fields measured in V/m (volt per meter).",
  "EMF": "1.4",
  "location": {
    "coordinates": [15.5523241, 38.188385],
    "type": "Point"
  },
  "source": "https://urbanite-nodel.comune.messina.it",
  "type": "ElectroMagneticObserved",
  "@context": [
    "https://smartdatamodels.org/context.jsonld",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ],
  "dateCreated": {
    "$date": "2022-09-08T10:04:27.059Z"
  },
  "dateModified": {
    "$date": "2022-09-08T14:22:50.286Z"
  }
}]
```

7.13 NoiseLevelObserved

This model is based on FIWARE's NoiseLevelObserved³⁸ data model. It is used to store an observation of those acoustic parameters that estimate noise pressure levels at a certain place and time.

In URBANITE, this model is used to store historical data from Messina.

Example:

```
[{
  "_id": "urn:ngsi-ld:noiseLevelObserved:messina:boccetta1577836895000",
  "alternateName": "Noise Pollution",
  "dateObservedFrom": {
    "$date": "2020-01-01T00:01:35.000Z"
  },
  "dateObservedTo": {
    "$date": "2020-01-01T00:01:35.000Z"
  },
  "description": "magnitude",
  "location": {
    "coordinates": [15.539456, 38.2022615],
    "type": "Point"
  }
}]
```

³⁷<https://github.com/smart-data-models/dataModel.Environment/tree/master/ElectroMagneticObserved>

³⁸<https://github.com/smart-data-models/dataModel.Environment/tree/master/NoiseLevelObserved>

```

    "source": "https://urbanite-nodel.comune.messina.it",
    "type": "NoiseLevelObserved",
    "frequencies": {},
    "@context": ["https://smartdatamodels.org/context.jsonld",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"],
    "dateCreated": {
        "$date": "2022-07-22T15:02:43.267Z"
    },
    "dateModified": {
        "$date": "2022-07-22T15:02:43.267Z"
    }
}
}]

```

7.14 MapLayer

This model is a new NGSI-LD compliant model defined in URBANITE to store map layers (GeoJSON files) of the use cases.

The context is defined in:

<https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/maplayer-ngsi.jsonld>

Example:

```

[
  {
    "id": "urn:ngsi-ld:MapLayer:messina:zone:Y",
    "alternateName": "test.geojson",
    "description": "test",
    "name": "Wifi zone Y",
    "type": "MapLayer",
    "@context": [
      "https://smartdatamodels.org/context.jsonld",
      "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld",
      "https://git.code.tecnalia.com/urbanite/public/-/raw/main/datamodels/maplayer-ngsi.jsonld"
    ],
    "map": {
      "type": "FeatureCollection",
      "name": "Helsinki_Espoo_Kaupunkipyöräasemat_2021",
      "description": "...",
      "crs": {
        "type": "name",
        "properties": {
          "name": "urn:ogc:def:crs:OGC:1.3:CRS84"
        }
      },
      "features": [
        {
          "type": "Feature",
          "properties": {
            "FID": 1,
            "ID": "501",
            "Nimi": "Hanasaari",
            "Nimn": "Hanaholmen",
            "Name": "Hanasaari",
            "Osoite": "Hanasaarenranta 1",
            "Adress": "Hanaholmsstranden 1",
            "Kaupunki": "Espoo",
            "Stad": "Esbo",
            "Operaattori": "CityBike Finland",
            "Kapasiteet": 10,
            "x": 24.840319,
            "y": 60.16582
          },
          "geometry": {
            "type": "Point",
            "coordinates": [
              24.840319,
              60.165819999000064
            ]
          }
        }
      ]
    }
  }
]

```

```

    }
  },
  {
    "type": "Feature",
    "properties": {
      "FID": 2,
      "ID": "503",
      "Nimi": "Keilalahti",
      "Namn": "Kägelviken",
      "Name": "Keilalahti",
      "Osoite": "Keilalahdentie 2",
      "Adress": "Kägelviksvägen 2",
      "Kaupunki": "Espoo",
      "Stad": "Esbo",
      "Operaattor": "CityBike Finland",
      "Kapasiteet": 28,
      "x": 24.827467,
      "y": 60.171524
    },
    "geometry": {
      "type": "Point",
      "coordinates": [
        24.827467,
        60.171523999000044
      ]
    }
  },
  ...
  {
    "type": "Feature",
    "properties": {
      "FID": 457,
      "ID": "405",
      "Nimi": "Jollas",
      "Namn": "Jollas",
      "Name": "Jollas",
      "Osoite": "Jollaksentie 33",
      "Adress": "Jollasvägen 33",
      "Kaupunki": " ",
      "Stad": " ",
      "Operaattor": " ",
      "Kapasiteet": 16,
      "x": 25.061667866825,
      "y": 60.1644074899774
    },
    "geometry": {
      "type": "Point",
      "coordinates": [
        25.061667866825004,
        60.1644074899774
      ]
    }
  }
],
"@context": [
  "https://urldefense.com/v3/__https://smartdatamodels.org/context.jsonld__;!!LQkDIss!UMW6FTBZUI53fk6oRsTj8p_UHVmJftyryrRSPLxpMnHOnl2q72j0USTzSTmZ9paGYcMEbUxaPdbF9LVvrlhM_F86914-UlnUmwk$ "
],
  },
  "dateCreated": "2022-09-19T09:18:21.660Z",
  "dateModified": "2022-09-19T10:53:34.308Z"
}
]

```

7.15 Metadata

Metadata must be provided in a DCAT-compliant format.

Example:

```

{
  "@graph": [
    {
      "@id": "https://urbanite-project.eu/ontology/URBANITE_PROJECT",
      "@type": "foaf:Organization",
      "homepage": "https://urbanite-project.eu/",
      "name": "URBANITE"
    },
    {
      "@id": "https://urbanite-project.eu/ontology/dataset/Bilbao_Calendar",
      "@type": "dcat:Dataset",
      "description": {"@language": "en", "@value": "Calendar data Bilbao"},
      "issued": "2021-06-17T09:24:56",
      "modified": "2021-06-17T09:25:03",
      "publisher": "https://urbanite-project.eu/ontology/URBANITE_PROJECT",
      "title": {"@language": "en", "@value": "Calendar data Bilbao"},
      "distribution": [
        "https://urbanite-project.eu/ontology/distribution/52c20f95-66a2-412d-9ac0-efe673707615",
        "https://urbanite-project.eu/ontology/distribution/5b9e9ed4-769c-435a-af0f-e25b41adbf6f",
        "https://urbanite-project.eu/ontology/distribution/1be969b1-88bd-4209-808a-004ccaef7c30"
      ],
      "keyword": ["Calendar", "Bilbao"]
    },
    {
      "@id": "https://urbanite-project.eu/ontology/distribution/1be969b1-88bd-4209-808a-004ccaef7c30",
      "@type": "dcat:Distribution",
      "description": "Calendar data Bilbao year 2016 in NGSI-LD representation",
      "format": "http://publications.europa.eu/resource/authority/file-type/JSON_LD",
      "license": "http://publications.europa.eu/resource/authority/licence/CC_BY",
      "title": "Calendar data Bilbao 2016",
      "accessURL":
        "https://bilbao.urbanite.esilab.org/data/getTData/calendar/bilbao?filters=%7B%22year%22%3D2016%7D"
    },
    {
      "@id": "https://urbanite-project.eu/ontology/distribution/52c20f95-66a2-412d-9ac0-efe673707615",
      "@type": "dcat:Distribution",
      "description": "Calendar data Bilbao year 2018 in NGSI-LD representation",
      "format": "http://publications.europa.eu/resource/authority/file-type/JSON_LD",
      "license": "http://publications.europa.eu/resource/authority/licence/CC_BY",
      "title": "Calendar data Bilbao 2018",
      "accessURL":
        "https://bilbao.urbanite.esilab.org/data/getTData/calendar/bilbao?filters=%7B%22year%22%3D2018%7D"
    },
    {
      "@id": "https://urbanite-project.eu/ontology/distribution/5b9e9ed4-769c-435a-af0f-e25b41adbf6f",
      "@type": "dcat:Distribution",
      "description": "Calendar data Bilbao year 2015 in NGSI-LD representation",
      "format": "http://publications.europa.eu/resource/authority/file-type/JSON_LD",
      "license": "http://publications.europa.eu/resource/authority/licence/CC_BY",
      "title": "Calendar data Bilbao 2015",
      "accessURL":
        "https://bilbao.urbanite.esilab.org/data/getTData/calendar/bilbao?filters=%7B%22year%22%3D2015%7D"
    }
  ],
  "@context": {
    "name": {"@id": "http://xmlns.com/foaf/0.1/name"},
    "homepage": {"@id": "http://xmlns.com/foaf/0.1/homepage"},
    "accessURL": {"@id": "http://www.w3.org/ns/dcat#accessURL"},
    "title": {"@id": "http://purl.org/dc/terms/title"},
    "license": {"@id": "http://purl.org/dc/terms/license"},
    "format": {"@id": "http://purl.org/dc/terms/format"},
    "description": {"@id": "http://purl.org/dc/terms/description"},
    "distribution": {"@id": "http://www.w3.org/ns/dcat#distribution", "@type": "@id"},
    "modified": {"@id": "http://purl.org/dc/terms/modified", "@type":
      "http://www.w3.org/2001/XMLSchema#dateTime"},
    "keyword": {"@id": "http://www.w3.org/ns/dcat#keyword"},
  }
}

```

```

    "issued": {"@id": "http://purl.org/dc/terms/issued", "@type":
"http://www.w3.org/2001/XMLSchema#dateTime"},
    "publisher": {"@id": "http://purl.org/dc/terms/publisher", "@type": "@id"},
    "dct": "http://purl.org/dc/terms/",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "dcat": "http://www.w3.org/ns/dcat#",
    "foaf": "http://xmlns.com/foaf/0.1/",
    "dc": "http://purl.org/dc/elements/1.1/"
  }
}

```

8 APPENDIX: Storage & Retrieval API

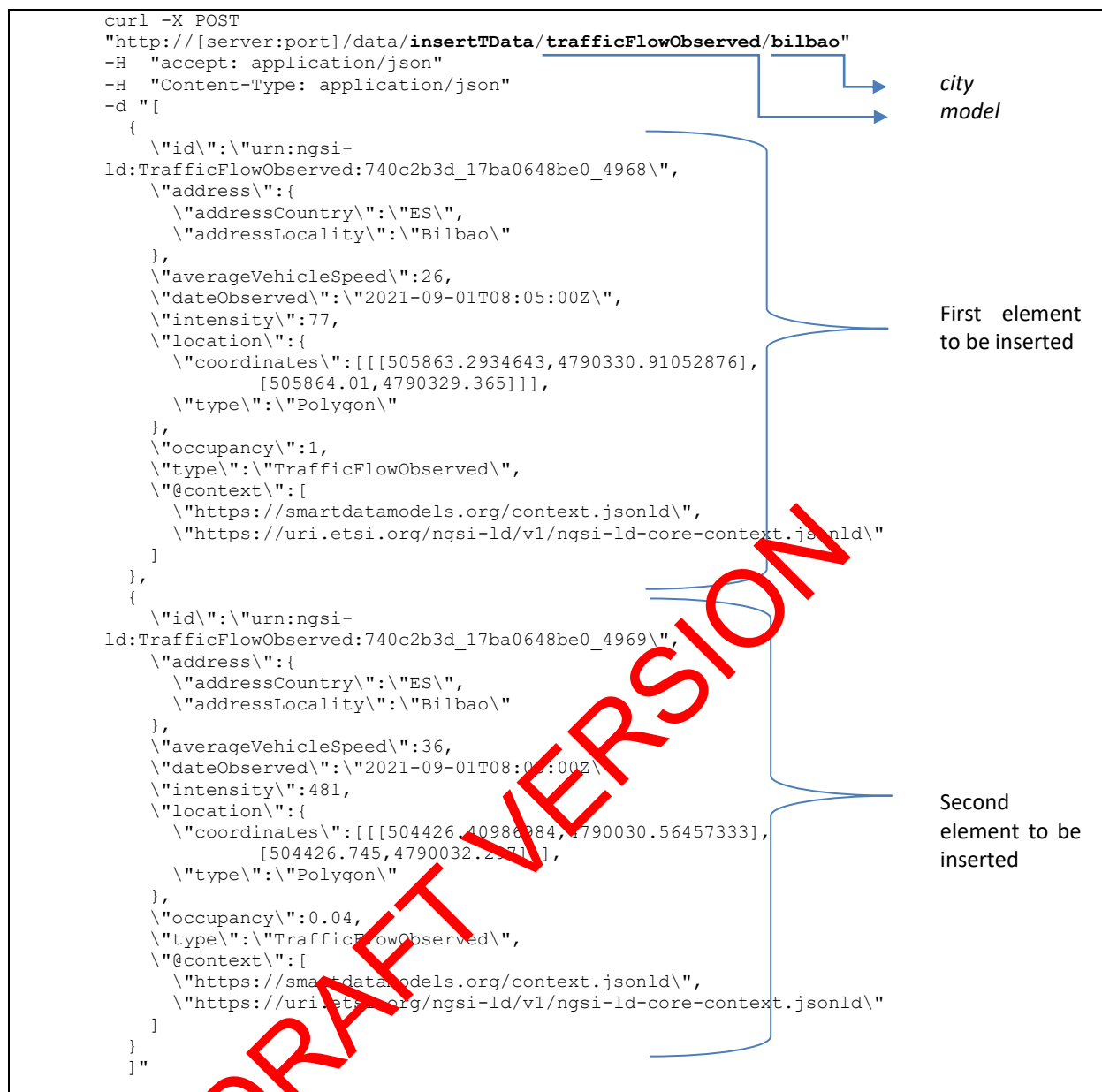
8.1 Storage

8.1.1 insertTData (POST)

Table 4: API for data insertion

/insertTData	Adds new data of a specific type to the database of a city.
Method	
POST	
Input Params (* means mandatory)	
model* (path param)	Text with the type of the data to be inserted. Must be one of the implemented data types, currently: <ul style="list-style-type: none"> • trafficFlowObserved • daySpecification • calendar • airQualityObserved • weatherObserved • event • censusObserved • pointOfInterest • transportStation • gtfsShape • touristTrip • originDestinationMatrix
city* (path param)	Text, with the city (use case) to which the data correspond. Must be one of: <ul style="list-style-type: none"> • bilbao • messina • helsinki • amsterdam
data* (request body)	Text, in a JSON array format, with the data to be inserted. These data must be in the format corresponding to the type of data indicated in the "model" parameter.
Success response	

200	<p>If all the input parameters are right, the method will return this code, and the JSON response will contain three fields with the details of the operations done:</p> <ul style="list-style-type: none"> • <i>inserted</i>: array with the IDs of the elements inserted successfully, • <i>updated</i>: array with the IDs of the elements updated. • <i>notInserted</i>: array with the IDs of the elements that couldn't be inserted. In this case, also will have a field <i>reason</i> with the description of the error. <p>Example:</p> <pre>{ "inserted": [{ "id": "urn:ngsi-ld:TrafficFlowObserved:5d993408_179e63999c7_-363a" }, { "id": "urn:ngsi-ld:TrafficFlowObserved:5d993408_179e63999c7_-3603" }], "notInserted": [{ "id": "urn:ngsi-ld:TrafficFlowObserved:5d993408_179e63999c7_1dec", "reason": "Wrong input data, missing some required field(s) or wrong values." }, { "id": "urn:ngsi-ld:TrafficFlowObserved:5d993408_179e63999c7_1df6", "reason": "Wrong input data, missing some required field(s) or wrong values." }], "updated": [{ "id": "urn:ngsi-ld:TrafficFlowObserved:740c2b3d_17ba0648be0_4968" }, { "id": "urn:ngsi-ld:TrafficFlowObserved:740c2b3d_17ba0648be0_4969" }] }</pre>
Error response	
400	<p>Bad Request</p> <p>The method checks if the data sent is in the required format (list of elements) and if each element to be inserted corresponds to the indicated model. It also checks if the <i>model</i> and <i>city</i> parameters are the expected values.</p> <p>If any of these checks fail, a Bad Request code will be returned. In this case, the method will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Example:</p> <pre>{ "Error": "Input data is not in required format (list of 'Traffic Flow Observation' objects)" }</pre>
Sample call	



8.1.2 updateTData (PUT)

Table 5: API for data update

/updateTData	Updates a specific record (identified by its ID field) of the database.
Method	
PUT	
Input Params (* means mandatory)	

model* (path param)	Text, with the data type of the element to be updated. Must be one of the implemented data types, currently: <ul style="list-style-type: none"> • trafficFlowObserved • daySpecification • calendar • airQualityObserved • weatherObserved • event • censusObserved • pointOfInterest • transportStation • gtfsShape • touristTrip • originDestinationMatrix
city* (path param)	Text, with the city (use case) to which the data correspond. Must be one of: <ul style="list-style-type: none"> • bilbao • messina • helsinki • amsterdam
id* (path param)	ID of the element to be updated.
data* (request body)	Text, in a JSON format, with the data to be updated. This data must be in the format corresponding to the type of data indicated in the "model" parameter.
Success response	

200	<p>If all the input parameters are right, the method will return this code, and the JSON response will contain one field <i>updatedData</i> with the new data of the element once updated.</p> <p>Example:</p> <pre> { "updatedData": { "address": { "addressCountry": "ES", "addressLocality": "Bilbao" }, "averageVehicleSpeed": 16, "dateObserved": "2021-08-17T00:45:00Z", "intensity": 36, "location": { "coordinates": [[[504417.9371092392, 4790030.564573327], [504453.27293873084, 4790030.564573327], [504453.27293873084, 4790133.21077546], [504417.9371092392, 4790133.21077546]]], "type": "Polygon" }, "name": "273", "type": "TrafficFlowObserved", "@context": ["https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"], "dateCreated": "2021-08-17T09:37:43.332Z", "dateModified": "2021-09-07T07:31:35.263Z", "id": "urn:ngsi-ld:TrafficFlowObserved:273:070920210045AAAA" } } </pre>
Error response	
400	<p>Bad Request</p> <p>The method checks if the data sent is in JSON format and if it corresponds to the indicated model. It also checks if the ID field of the data (mandatory field) is the same as the one passed as parameter, as well if the <i>model</i> and <i>city</i> parameters are the expected values.</p> <p>If any of these checks fail, a Bad Request code will be returned. In this case, the method will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Examples:</p> <pre> { "Error": "Input data is not in required format ('Traffic Flow Observation' object)" } { "Error": "Wrong input data, IDs are different." } </pre>
404	<p>Not Found</p> <p>If the element to be update doesn't exist, a 404 code will be returned, with a JSON response with the error:</p> <pre> { "Error": "Document 'urn:ngsi-ld:TrafficFlowObserved:273:07092023456' not found." } </pre>
Sample call	

```

curl -X PUT
"http://[server:port]/data/updateTData/trafficFlowObserved/bilbao/
urn%3Angsi-ld%3ATrafficFlowObserved%3A273%3A070920210045AAAA"
-H "accept: application/json"
-H "Content-Type: application/json"
-d "
{"id": "\urn:ngsi-ld:TrafficFlowObserved:273:070920210045AAAA",
"address": {
"addressCountry": "\ES",
"addressLocality": "\Bilbao"
},
"averageVehicleSpeed": 16,
"dateObserved": "\2021-08-17T00:45:00Z",
"intensity": 36,
"location": {
"coordinates": [[ [504417.9371092392, 4790030.564573327],
[504453.27293873084, 4790030.564573327],
[504453.27293873084, 4790133.21077546],
[504417.9371092392, 4790133.21077546] ]],
"type": "\Polygon"
},
"name": "\273",
"type": "\TrafficFlowObserved",
"@context": [
"https://smartdatamodels.org/context.jsonld",
"https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
]
}"

```

city
model
id

Data to
be
updated

8.1.3 deleteTDate (DELETE)

Table 6: API to delete data

/deleteTData	Delete a single record from the database
Method	
DELETE	
Input Params (* means mandatory)	
model* (path param)	<p>Text, with the data type of the element to be deleted. Must be one of the implemented data types currently:</p> <ul style="list-style-type: none"> • trafficFlowObserved • daySpecification • calendar • airQualityObserved • weatherObserved • event • censusObserved • pointOfInterest • transportStation • gtfsShape • touristTrip • originDestinationMatrix

city* (path param)	Text, with the city (use case) to which the data correspond. Must be one of: <ul style="list-style-type: none"> • bilbao • messina • helsinki • amsterdam
id* (path param)	ID of the element to be deleted.
Success response	
200	<p>If all the input parameters are right, the method will return this code, and the JSON response will contain one field with the id deleted.</p> <p>Example:</p> <pre>{ "deleted": "urn:ngsi:touristTrip:bikes:61825871" }</pre>
Error response	
400	<p>Bad Request</p> <p>The method checks if the data sent contains a valid id.</p> <p>If this check fails, a Bad Request code will be returned. In this case, the method will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Examples:</p> <pre>{ "Error": "Wrong input data." }</pre>
404	<p>Not Found</p> <p>If the element to be deleted doesn't exist, a 404 code will be returned, with a JSON response with the error:</p> <pre>{ "Error": "Document 'urn:ngsi:touristTrip:bikes:61825871' not found." }</pre>
Sample call	
<pre>curl -X DELETE "https:// [server:port] /data/deleteTData/touristTrip/bilbao/urn%3Angsi%3AtouristTrip%3Abikes%3A61 825871" -H "accept: application/json"</pre>	

8.2 Retrieval

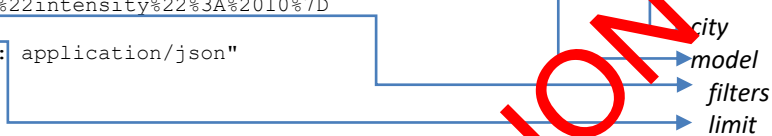
8.2.1 getTData (GET)

Table 7: API for data retrieval

/getTData	Gets data of the specified type from the database. Some filters can be applied to model fields.
------------------	---

Method	
GET	
Input Params (* means mandatory)	
model* (path param)	Text, with the data type of the element to be updated. Must be one of the implemented data types, currently: <ul style="list-style-type: none"> • trafficFlowObserved • daySpecification • calendar • airQualityObserved • weatherObserved • event • censusObserved • pointOfInterest • transportStation • gtfsShape • touristTrip • originDestinationMatrix
city* (path param)	Text, with the city (use case) to which the data corresponds. Must be one of: <ul style="list-style-type: none"> • bilbao • messina • helsinki • amsterdam
filters (query param)	Different filters to be applied to the search of the records to be returned. They must be in JSON format and match the names of the fields of the model. Otherwise, a BAD_REQUEST error will be returned. Example: <pre>{ "occupancy": 10, "intensity": 20 }</pre> both fields, <i>occupancy</i> and <i>intensity</i> are part of the <i>trafficFlowObserved</i> model.
limit (query param)	Number of records to be retrieved. If not set, the default number of records (1000) will be returned.
Success response	

200	<p>If all the input parameters are right, the method will return this code, and the response will be a JSON array with the list of the elements requested.</p> <p>Example:</p> <pre>[{ "id": "urn:ngsi-ld:TrafficFlowObserved:740c2b3d_17bbf3cb8f8_851", "address": { "addressCountry": "ES", "addressLocality": "Bilbao" }, "averageVehicleSpeed": 0, "dateObserved": "2021-09-07T07:45:00Z", "intensity": 10, "location": { "coordinates": [[[504169.87,4790169.594],[504215.57915011,4790128.87897186],[504268.78134 883,4790081.60210874]]], "type": "Polygon" }, "occupancy": 0, "type": "TrafficFlowObserved", "@context": ["https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"], "dateCreated": "2021-09-07T07:53:43.872Z", "dateModified": "2021-09-07T07:53:43.872Z" }, { "id": "urn:ngsi-ld:TrafficFlowObserved:59d1ce0d_17bbf0b8c3f_-5ba8", "address": { "addressCountry": "ES", "addressLocality": "Bilbao" }, "averageVehicleSpeed": 4, "dateObserved": "2021-09-07T06:50:00Z", "intensity": 10, "location": { "coordinates": [[[505482.00123305,478960.88571801],[505595.37956903,4789700.8144343],[50 5621.18245562,4789721.03079363],[505621.339,4789721.18]]], "type": "Polygon" }, "occupancy": 0, "type": "TrafficFlowObserved", "@context": ["https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"], "dateCreated": "2021-09-07T06:57:32.011Z", "dateModified": "2021-09-07T06:57:32.011Z" }]</pre>
Error response	

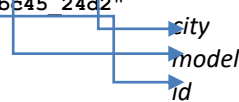
400	<p>Bad Request</p> <p>The method checks if the filters sent are in the required format (JSON) and if its content are fields of the specified data model. It also checks if the <i>model</i> and <i>city</i> parameters are the expected values, and if the <i>limit</i> parameter (if set) is greater than 0.</p> <p>If any of these checks fail, a Bad Request code will be returned. In this case, the method will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Example:</p> <pre>{ "Error": "Model {trafficFlowObserved} has not a field 'speed'" }</pre>
Sample call	
<pre>curl -X GET "http://[server:port]/data/getTData/trafficFlowObserved/bilbao? filters=%7B%22intensity%22%3A%2010%7D &limit=2" -H "accept: application/json"</pre> 	

8.2.2 getTData (single record) (GET)

Table 8: API for data retrieval (specific record)

/getTData (single record)	Gets a specific record from the database, identified by its ID.
Method	
GET	
Input Params (* means mandatory)	
model* (path param)	<p>Text, with the data type of the element to be updated. Must be one of the implemented data types, currently:</p> <ul style="list-style-type: none"> • trafficFlowObserved • daySpecification • calendar • airQualityObserved • weatherObserved • event • censusObserved • pointOfInterest • transportStation • gtfsShape • touristTrip • originDestinationMatrix

city* (path param)	Text, with the city (use case) to which the data correspond. Must be one of: <ul style="list-style-type: none"> • bilbao • messina • helsinki • amsterdam
id* (path param)	ID of the element to be updated.
Success response	
200	<p>If the element requested exists, the method will return this code, and the JSON response will contain the element data.</p> <p>Example:</p> <pre>{ "id": "urn:ngsi-ld:TrafficFlowObserved:740c2b3d_17b3e3ebc45_24d2", "address": { "addressCountry": "ES", "addressLocality": "Bilbao" }, "averageVehicleSpeed": 0, "dateObserved": "2021-08-31T22:00:00Z", "intensity": 9, "location": { "coordinates": [[[505863.2924647, 4790330.91052876], [505864.01, 4790329.365], [505864.637, 4790327.54]]], "type": "Polygon" }, "occupancy": 0, "type": "TrafficFlowObserved", "@context": ["https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"], "dateCreated": "2021-08-31T22:09:23.993Z", "dateModified": "2021-08-31T22:09:23.993Z" }</pre>
Error response	
400	<p>Bad Request</p> <p>The method checks if the <i>model</i> and <i>city</i> parameters are the expected values.</p> <p>If any of these checks fail, a Bad Request code will be returned. In this case, the method will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Example:</p> <pre>{ "Error": "Invalid value 'trafficFlow' for parameter 'model'" }</pre>

404	<p>Not Found</p> <p>If the element requested doesn't exist, a 404 code will be returned, with a JSON response with the error:</p> <pre>{ "Error": "Document 'urn:ngsi-ld:TrafficFlowObserved:273:07092023456' not found." }</pre>
Sample call	
<pre>curl -X GET "http://[server:port]/data/getTData/trafficFlowObserved/bilbao/ urn%3Angsi-ld%3ATrafficFlowObserved%3A740c2b3d_17b9e3eb-45_24d2" -H "accept: application/json"</pre> 	

8.2.3 getTDataRange (GET)

Table 9: API for data retrieval (time range)

/getTDataRange	Gets data of a specific model from the database within a specific time range.
Method	
GET	
Input Params (* means mandatory)	
model* (path param)	<p>Text, with the data type of the element to be updated. Must be one of the implemented data types, currently:</p> <ul style="list-style-type: none"> • trafficFlowObserved • daySpecification • calendar • airQualityObserved • weatherObserved • event • censusObserved • pointOfInterest • transportStation • gtfsShape • touristTrip • originDestinationMatrix
city* (path param)	<p>Text, with the city (use case) to which the data correspond. Must be one of:</p> <ul style="list-style-type: none"> • bilbao • messina • helsinki • amsterdam

startDate* (query param)	Date and time (ISO8601 ³⁹ UTC format) from which to get the data. Mandatory if parameter <i>endDate</i> is not present. Example: 2021-01-07T00:00:00.000Z
endDate* (query param)	Date and time (ISO8601 UTC format) until which to get the data. Mandatory if parameter <i>startDate</i> is not present. Example: 2021-01-08T22:45:00.000Z
limit (query param)	Number of records to be retrieved. If not set, the default number of records (1000) will be returned.
Success response	

DRAFT VERSION

³⁹ <https://www.iso.org/iso-8601-date-and-time-format.html>

200	<p>If all the input parameters are right, the method will return this code, and the response will be a JSON array with the list of the elements requested.</p> <p>Example:</p> <pre>[{ "id": "urn:ngsi-ld:TrafficFlowObserved:740c2b3d_17bb86bfdd2_28f0", "address": { "addressCountry": "ES", "addressLocality": "Bilbao" }, "averageVehicleSpeed": 0, "dateObserved": "2021-09-06T00:00:00Z", "intensity": 0, "location": { "coordinates": [[505863.2934643, 4790330.91052876], [505875.24029728, 4790333.17459119], [505863.2934643, 4790330.91052876]]], "type": "Polygon" }, "occupancy": 0, "type": "TrafficFlowObserved", "@context": ["https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"], "dateCreated": "2021-09-06T00:07:19.758Z", "dateModified": "2021-09-06T00:07:19.758Z" }, { "id": "urn:ngsi-ld:TrafficFlowObserved:740c2b3d_17bb86bfdd2_28f1", "address": { "addressCountry": "ES", "addressLocality": "Bilbao" }, "averageVehicleSpeed": 0, "dateObserved": "2021-09-06T00:00:00Z", "intensity": 18, "location": { "coordinates": [[504426.40986984, 4790030.56457333], [504425.67779232, 4790043.20283264], [504426.40986984, 4790030.56457333]]], "type": "Polygon" }, "occupancy": 0, "type": "TrafficFlowObserved", "@context": ["https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"], "dateCreated": "2021-09-06T00:07:19.760Z", "dateModified": "2021-09-06T00:07:19.760Z" }]</pre>
Error response	

400	<p>Bad Request</p> <p>The method makes several checks of the parameters. It checks if the <i>model</i> and <i>city</i> parameters are the expected values, and if the <i>limit</i> parameter (if set) is greater than 0. Also checks that at least one of the parameters <i>startDate</i> and <i>endDate</i> is set, and if so, checks that they are in ISO8601 format.</p> <p>If any of these checks fail, a Bad Request code will be returned, and will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Examples:</p> <pre>{ "Error": "No time range specified. 'startDate' and/or 'endDate' must be indicated." }</pre> <pre>{ "Error": "Invalid value '2021/08/01' for parameter 'startDate'" }</pre>
Sample call	
<pre>curl -X GET "http://[server:port]/data/getTDataRange/trafficFlowObserved/bilbao? startDate=2021-09-05T00%3A00%3A00.000Z &endDate=2021-09-06T00%3A00%3A00.000Z &limit=2" -H "accept: application/json"</pre> <p>city model startDate endDate limit</p> <pre>curl -X GET "http://[server:port]/data/getTDataRange/trafficFlowObserved/bilbao? startDate=2021-09-05T00%3A00%3A00.000Z &limit=2" -H "accept: application/json"</pre> <p>city model startDate limit</p>	

8.2.4 getSupportedDataModels (GET)

Table 10: API for the retrieval of the data models information

/getSupportedDataModels	Returns information about all the data models that are currently implemented.
Method	
GET	
Input Params (* means mandatory)	
None	
Success response	
200	Returns a JSON array with the information of the data models implemented. Each element (model) will have the following fields:

- **id:** identifier of the model, that is usually used as *model* parameter in the other services.
- **name:** full name of the model.
- **description:** brief description of the model.
- **reference:** link to the official reference of the model (i.e: FIWARE models). If the model has been developed specifically for the project, the reference will be empty.
- **example:** an example of the structure of the model.

Example (reduced to two models only):

```
[
  {
    "id": "trafficFlowObserved",
    "name": "Traffic Flow Observed",
    "description": "An observation of traffic flow conditions at a certain place and time.",
    "reference": "https://github.com/smart-data-models/dataModel.Transportation/tree/master/TrafficFlowObserved",
    "example": {
      "@context": [
        "https://smartdatamodels.org/context.jsonld",
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
      ],
      "address": {
        "addressCountry": "ES",
        "addressLocality": "Valladolid",
        "streetAddress": "Avenida de Salamanca",
        "type": "PostalAddress"
      },
      "averageHeadwayTime": 10.5,
      "averageVehicleLength": 4.87,
      "averageVehicleSpeed": 52.7,
      "dateObserved": "2015-12-07T11:10:00Z",
      "id": "urn:ngsi-ld:TrafficFlowObserved:TrafficFlowObserved-Valladolid-osm-60821110",
      "intensity": 197,
      "laneDirection": "forward",
      "laneId": 1,
      "location": {
        "coordinates": [[-4.73735395519672, 41.6538181849672], [-4.7341485165193, 41.6600594193478], [-4.73447575302641, 41.659585195093]],
        "type": "LineString"
      },
      "occupancy": 0.76,
      "reversedLane": false,
      "type": "TrafficFlowObserved"
    },
    {
      "id": "calendar",
      "name": "Calendar",
      "description": "Information about calendars: year, city, days...",
      "reference": "",
      "example": {
        "example": [
          {
            "id": "urn:ngsi-ld:Calendar:Bilbao:2015",
            "type": "Calendar",
            "city": "Bilbao",
            "location": {
              "coordinates": [-2.93609619140625, 43.26345626603949],
              "type": "Point"
            },
            "year": 2015,
            "days": [
              "urn:ngsi-ld:DaySpecification:Bilbao:2015_01_01",
              "urn:ngsi-ld:DaySpecification:Bilbao:2015_01_02"
            ],
            "createdAt": "2021-05-20T09:32:08.809Z",

```

	<pre> "modifiedAt": "2021-05-20T09:52:04.255Z", "@context": ["https://git.code.tecnalia.com/urbanite/public/- /raw/master/datamodels/calendar-ngsi.jsonld", "https://git.code.tecnalia.com/urbanite/public/- /raw/master/datamodels/calendar-ngsi.jsonld2"] }, { "id": "urn:ngsi-ld:DaySpecification:Bilbao:2015_01_01", "type": "DaySpecification", "date": "2015-01-01", "description": "AÃfÃto nuevo", "workingDay": 0, "schoolDay": 0, "publicHoliday": 3, "weekDay": 4, "createdAt": "2021-05-24T13:26:41.828Z", "modifiedAt": "2021-05-25T08:49:11.841Z", "@context": ["https://git.code.tecnalia.com/urbanite/public/- /raw/master/datamodels/calendar-ngsi.jsonld"] }, { "id": "urn:ngsi-ld:DaySpecification:Bilbao:2015_01_02", "type": "DaySpecification", "date": "2015-01-02", "description": "", "workingDay": 1, "schoolDay": 1, "publicHoliday": 0, "weekDay": 5, "createdAt": "2021-05-25T08:26:22.811Z", "modifiedAt": "2021-05-25T08:49:11.946Z", "@context": ["https://git.code.tecnalia.com/urbanite/public/- /raw/master/datamodels/calendar-ngsi.jsonld"] }] } }] </pre>
Sample call	
<pre> curl -X GET "http://localhost:4242/getSupportedDataModels" -H "accept: application/json" </pre>	

8.2.5 getDistinct (GET)

Table 11: API for the retrieval of distinct values.

/getTDistinct	Returns the different values for a specific field.
Method	
GET	
Input Params (* means mandatory)	

model* (path param)	Text, with the data type of the element to be updated. Must be one of the implemented data types, currently: <ul style="list-style-type: none"> • trafficFlowObserved • daySpecification • calendar • airQualityObserved • weatherObserved • event • censusObserved • pointOfInterest • transportStation • gtfsShape • touristTrip • originDestinationMatrix
city* (path param)	Text, with the city (use case) to which the data correspond. Must be one of: <ul style="list-style-type: none"> • bilbao • messina • helsinki • amsterdam
field (query param)	Data model field to return its different values.
Success response	

200	<p>If all the input parameters are right, the method will return this code, and the response will be a JSON array with the list of the elements requested.</p> <p>Example:</p> <pre> { "values": [{ "name": "8", "description": "Basurto-Zorroza" }, { "name": "6", "description": "Abando" }, { "name": "5", "description": "Ibaiondo" }, { "name": "2", "description": "Uribarri" }, { "name": "7", "description": "Recalde" }, { "name": "1", "description": "Deusto" }, { "name": "3", "description": "Otxarkoaga-Murdinaga" }, { "name": "4", "description": "Lezama" }] }</pre>
Error response	
400	<p>Bad Request</p> <p>The method makes several checks of the parameters. It checks if the <i>model</i> and <i>city</i> parameters are the expected values, also checks that the fields correspond to the model.</p> <p>If any of these checks fail, a Bad Request code will be returned, and will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Example:</p> <pre> { "Error": "Wrong fields , please check 'GtfsShape' data model's fields." }</pre>
Sample call	
<pre> curl -X GET "http://[server:port]/data/getDistinct/gtfsShape/bilbao? field=description%2Cname &limit=2" -H "accept: application/json"</pre>	

8.3 Metadata

8.3.1 dataset (PUT)

Table 12: API for the insert and update of metadata

/dataset	Adds new metadata of a dataset into the database, or updates the metadata if already exists for that id.
Method	
PUT	
Input Params (* means mandatory)	
id* (query param)	Unique identifier of the metadata.
data* (request body)	Text, in JSON format, with the metadata to be inserted.
Success response	
200	<p>If all the input parameters are right, the method will return this code, and the JSON response will contain three fields with the details of the operations done:</p> <ul style="list-style-type: none"> <i>inserted</i>: array with the ID of the metadata if it has been inserted successfully. <i>updated</i>: array with the ID of the metadata if it has been updated. <i>notInserted</i>: array with the ID of the metadata if it couldn't be inserted. In this case, also will have a field <i>reason</i> with the description of the error. <p>Example:</p> <pre>{ "inserted": [{ "id": "6bb9c361_177a635e86a23333333" }], "notInserted": [], "updated": [] }</pre>

Error response	
400	<p>Bad Request</p> <p>The method checks that the ID and the metadata has been included, and that this one is in JSON format.</p> <p>If any of these checks fail, a Bad Request code will be returned. In this case, the method will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Example:</p> <pre>{ "Error": "Input metadata is not in JSON format" }</pre>
Sample call	

DRAFT VERSION

```

curl -X PUT
"http://[server:port]/data/dataset?id=6bb9c361_177a635e86a2"
-H "accept: application/json"
-H "Content-Type: application/json"
-d "{
  \"_id\": \"6bb9c361_177a635e86a2\",
  \"@graph\": [
    {
      \"@id\": \"https://urbanite-project.eu/ontology/URBANITE_PROJECT\",
      \"@type\": \"foaf:Organization\",
      \"homepage\": \"https://urbanite-project.eu/\",
      \"name\": \"URBANITE\"
    },
    {
      \"@id\": \"https://urbanite-project.eu/ontology/dataset/Bilbao_Calendar\",
      \"@type\": \"dcat:Dataset\",
      \"description\": {
        \"@language\": \"en\",
        \"@value\": \"Calendar data Bilbao\"
      },
      \"issued\": \"2021-05-12T10:36:46\",
      \"modified\": \"2021-05-12T15:36:46\",
      \"publisher\": \"https://urbanite-project.eu/ontology/URBANITE_PROJE\",
      \"title\": {
        \"@language\": \"en\",
        \"@value\": \"Calendar data Bilbao\"
      },
      \"distribution\": [
        \"https://urbanite-project.eu/ontology/distribution/a732f6c6-fcd8-4962-8aa9-db7d913a20ae\"
      ],
      \"keyword\": [\"Calendar\", \"Bilbao\"]
    },
    {
      \"@id\": \"https://urbanite-project.eu/ontology/distribution/a732f6c6-fcd8-4962-8aa9-db7d913a20ae\",
      \"@type\": \"dcat:Distribution\",
      \"description\": \"Calendar data Bilbao year 2015 in NGSI-LD representation\",
      \"format\": \"http://publications.europa.eu/resource/authority/file-type/JSON_LD\",
      \"license\": \"http://publications.europa.eu/resource/authority/licence/CC_BY\",
      \"title\": \"Calendar data Bilbao 2015\",
      \"accessURL\": \"http://storageAPI-to-bedefined/2015\"
    }
  ],
  \"@context\": {
    \"name\": {\"@id\": \"http://xmlns.com/foaf/0.1/name\"},
    \"homepage\": {\"@id\": \"http://xmlns.com/foaf/0.1/homepage\"},
    \"accessURL\": {\"@id\": \"http://www.w3.org/ns/dcat#accessURL\"},
    \"title\": {\"@id\": \"http://purl.org/dc/terms/title\"},
    \"license\": {\"@id\": \"http://purl.org/dc/terms/license\"},
    \"format\": {\"@id\": \"http://purl.org/dc/terms/format\"},
    \"description\": {\"@id\": \"http://purl.org/dc/terms/description\"},
    \"distribution\": {\"@id\": \"http://www.w3.org/ns/dcat#distribution\"},
    \"type\": {\"@id\": \"\"},
    \"keyword\": {\"@id\": \"http://www.w3.org/ns/dcat#keyword\"},
    \"issued\": {\"@id\": \"http://purl.org/dc/terms/issued\"},
    \"type\": {\"@id\": \"http://www.w3.org/2001/XMLSchema#dateTime\"},
    \"publisher\": {\"@id\": \"http://purl.org/dc/terms/publisher\"},
    \"type\": {\"@id\": \"\"},
    \"modified\": {\"@id\": \"http://purl.org/dc/terms/modified\"},
    \"type\": {\"@id\": \"http://www.w3.org/2001/XMLSchema#dateTime\"},
    \"dct\": \"http://purl.org/dc/terms/\",
    \"rdf\": \"http://www.w3.org/1999/02/22-rdf-syntax-ns#\",
    \"xsd\": \"http://www.w3.org/2001/XMLSchema#\",
    \"rdfs\": \"http://www.w3.org/2000/01/rdf-schema#\",
    \"dcat\": \"http://www.w3.org/ns/dcat#\",
    \"foaf\": \"http://xmlns.com/foaf/0.1/\",
    \"dc\": \"http://purl.org/dc/elements/1.1/\"
  }
}

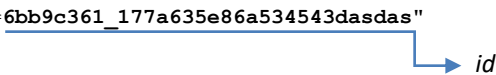
```

Id

M e t a d a t a

8.3.2 dataset (DELETE)

Table 13: API for the deletion of metadata

/dataset	Delete the metadata of a dataset from the database.
Method	
DELETE	
Input Params (* means mandatory)	
id* (query param)	Unique identifier of the metadata.
Success response	
200	<p>If the metadata with the <i>id</i> passed is deleted, the method will return this code, and the JSON response will contain one field <i>deleted</i> with the <i>id</i> of the metadata deleted.</p> <p>Example:</p> <pre>{ "deleted": "6bb9c361_177a635e86a534543" }</pre>
Error response	
400	<p>Bad Request</p> <p>The method checks that the <i>id</i> parameter has been included, otherwise Bad Request code will be returned.</p> <p>In this case, the method will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Example:</p> <pre>{ "Error": "Parameter 'id' is required." }</pre>
404	<p>Not found</p> <p>If the metadata with the <i>id</i> passed doesn't exist, this error will be returned with a JSON response with the error.</p> <p>Example:</p> <pre>{ "Error": "Dataset '6bb9c361_177a635e86a234' not found." }</pre>
Sample call	
<pre>curl -X DELETE "http://[server:port]/data/dataset?id=6bb9c361_177a635e86a534543dasdas" -H "accept: application/json"</pre> 	

8.3.3 getDataset (GET)

/getDataset	Gets the metadata of a specific dataset from the database.
Method	
GET	
Input Params (* means mandatory)	
id* (query param)	Unique identifier of the metadata.
Success response	
200	<p>If the metadata with the <i>id</i> passed exists, the method will return this code, and the JSON response will contain the metadata stored in the database in JSON format.</p> <p>Example:</p> <pre> { "@graph": [{ "@id": "https://urbanite-project.eu/ontology/URBANITE_PROJECT", "@type": "foaf:Organization", "homepage": "https://urbanite-project.eu/", "name": "URBANITE" }, { "@id": "https://urbanite-project.eu/ontology/dataset/Bilbao_Calendar", "@type": "dcat:Dataset", "description": {"@language": "en", "@value": "Calendar data Bilbao"}, "issued": "2021-05-12T10:36:46", "modified": "2021-05-12T15:36:46", "publisher": "https://urbanite-project.eu/ontology/URBANITE_PROJECT", "title": {"@language": "en", "@value": "Calendar data Bilbao"}, "distribution": ["https://urbanite-project.eu/ontology/distribution/a732f6c6-fcd8-4962-8aa9-db7d913a20ae", "https://urbanite-project.eu/ontology/distribution/059fd3cc-92b3-4f3d-97de-d050ae022eb5"], "keyword": ["Calendar", "Bilbao"] }, { "@id": "https://urbanite-project.eu/ontology/distribution/059fd3cc-92b3-4f3d-97de-d050ae022eb5", "@type": "dcat:Distribution", "description": "Calendar data Bilbao year 2016 in NGSI-LD representation", "format": "http://publications.europa.eu/resource/authority/file-type/JSON_LD", "license": "http://publications.europa.eu/resource/authority/licence/CC_BY", "title": "Calendar data Bilbao 2016", "accessURL": "http://storageAPI-to-be-defined/2016" }, { "@id": "https://urbanite-project.eu/ontology/distribution/a732f6c6-fcd8-4962-8aa9-db7d913a20ae", "@type": "dcat:Distribution", "description": "Calendar data Bilbao year 2015 in NGSI-LD representation", "format": "http://publications.europa.eu/resource/authority/file-type/JSON_LD", "license": </pre>

	<pre> "http://publications.europa.eu/resource/authority/licence/CC_BY", "title": "Calendar data Bilbao 2015", "accessURL": "http://storageAPI-to-bedefined/2015" }], "@context": { "name": {"@id": "http://xmlns.com/foaf/0.1/name"}, "homepage": {"@id": "http://xmlns.com/foaf/0.1/homepage"}, "accessURL": {"@id": "http://www.w3.org/ns/dcat#accessURL"}, "title": {"@id": "http://purl.org/dc/terms/title"}, "license": {"@id": "http://purl.org/dc/terms/license"}, "format": {"@id": "http://purl.org/dc/terms/format"}, "description": {"@id": "http://purl.org/dc/terms/description"}, "distribution": {"@id": "http://www.w3.org/ns/dcat#distribution"}, "@type": "@id"}, "keyword": {"@id": "http://www.w3.org/ns/dcat#keyword"}, "issued": {"@id": "http://purl.org/dc/terms/issued", "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}, "publisher": {"@id": "http://purl.org/dc/terms/publisher", "@type": "@id"}, "modified": {"@id": "http://purl.org/dc/terms/modified", "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}, "dct": "http://purl.org/dc/terms/", "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#", "xsd": "http://www.w3.org/2001/XMLSchema#", "rdfs": "http://www.w3.org/2000/01/rdf-schema#", "dcat": "http://www.w3.org/ns/dcat#", "foaf": "http://xmlns.com/foaf/0.1/", "dc": "http://purl.org/dc/elements/1.1/" } } </pre>	
Error response		
400	<p>Bad Request</p> <p>The method checks that the <i>id</i> parameter has been included, otherwise Bad Request code will be returned.</p> <p>In this case, the method will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Example:</p> <pre> { "Error": "Parameter 'id' is required." } </pre>	
404	<p>Not found</p> <p>If the metadata with the <i>id</i> passed doesn't exist, this error will be returned with a JSON response with the error.</p> <p>Example:</p> <pre> { "Error": "Dataset '6bb9c361_177a635e86a234' not found." } </pre>	
Sample call		
<pre> curl -X GET "http://[server:port]/data/getDataset?id=6bb9c361_177a635e86a" -H "accept: application/json" </pre> <p style="text-align: right;">  </p>		

8.3.4 getCatalogueDatasets (GET)

Table 14: API for the retrieval of dataset metadata

/getCatalogueDatasets	Gets the metadata of all the datasets stored in the database.
Method	
GET	
Input Params (* means mandatory)	
None	
Success response	
200	<p>Returns a JSON array with all the datasets stored.</p> <p>Example:</p> <pre>[{ "id": "6bb9c361_177a635e86a", "metadata": { "@graph": [{ "@id": "https://urbanite-project.eu/ontology/URBANITE_PROJECT", "@type": "foaf:Organization", "homepage": "https://urbanite-project.eu/", "name": "URBANITE" }, { "@id": "https://urbanite-project.eu/ontology/dataset/Bilbao_Calendar", "@type": "dcat:Dataset", "description": {"@language": "en", "@value": "Calendar data Bilbao"}, "issued": "2021-05-12T10:36:46", "modified": "2021-05-12T15:36:46", "publisher": "https://urbanite-project.eu/ontology/URBANITE_PROJECT", "title": {"@language": "en", "@value": "Calendar data Bilbao"}, "distribution": ["https://urbanite-project.eu/ontology/distribution/a732f6c6-fcd8-4962-8aa9-db7d913a20ae", "https://urbanite-project.eu/ontology/distribution/059fd3cc-92b3-4f3d-97de-d050ae022eb5"], "keyword": ["Calendar", "Bilbao"] }, { "@id": "https://urbanite-project.eu/ontology/distribution/059fd3cc-92b3-4f3d-97de-d050ae022eb5", "@type": "dcat:Distribution", "description": "Calendar data Bilbao year 2016 in NGSi-LD representation", "format": "http://publications.europa.eu/resource/authority/file-type/JSON_LD", "license": "http://publications.europa.eu/resource/authority/licence/CC_BY", "title": "Calendar data Bilbao 2016", "accessURL": "http://storageAPI-to-be-defined/2016" }] }, "@type": "dcat:Distribution", </pre>

	<pre> "description": "Calendar data Bilbao year 2015 in NGSI-LD representation", "format": "http://publications.europa.eu/resource/authority/file- type/JSON_LD", "license": "http://publications.europa.eu/resource/authority/licence/CC_BY", "title": "Calendar data Bilbao 2015", "accessURL": "http://storageAPI-to-bedefined/2015" }], "@context": { "name": {"@id": "http://xmlns.com/foaf/0.1/name"}, "homepage": {"@id": "http://xmlns.com/foaf/0.1/homepage"}, "accessURL": {"@id": "http://www.w3.org/ns/dcat#accessURL"}, "title": {"@id": "http://purl.org/dc/terms/title"}, "license": {"@id": "http://purl.org/dc/terms/license"}, "format": {"@id": "http://purl.org/dc/terms/format"}, "description": {"@id": "http://purl.org/dc/terms/description"}, "distribution": {"@id": "http://www.w3.org/ns/dcat#distribution", "@type": "@id"}, "keyword": {"@id": "http://www.w3.org/ns/dcat#keyword"}, "issued": {"@id": "http://purl.org/dc/terms/issued", "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}, "publisher": {"@id": "http://purl.org/dc/terms/publisher", "@type": "@id"}, "modified": {"@id": "http://purl.org/dc/terms/modified", "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}, "dct": "http://purl.org/dc/terms/", "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#", "xsd": "http://www.w3.org/2001/XMLSchema#", "rdfs": "http://www.w3.org/2000/01/rdf-schema#", "dcat": "http://www.w3.org/ns/dcat#", "foaf": "http://xmlns.com/foaf/0.1/", "dc": "http://purl.org/dc/elements/1.1/" } }, { "id": "6bb9c361_177a635sfd124a", "metadata": { "id": "6bb9c361_177a635sfd124a", "metadata": { "@graph": [{ "@id": "https://urbanite- project.eu/ontology/URBANITE_PROJECT", "@type": "rdf:Organization", "homepage": "https://urbanite-project.eu/", "name": "URBANITE" }, { "@id": "https://urbanite- project.eu/ontology/dataset/Bilbao_Calendar", "@type": "dcat:Dataset", "description": {"@language": "en", "@value": "Calendar data Messina"}, "issued": "2021-05-12T10:36:46", "modified": "2021-05-12T15:36:46", "publisher": "https://urbanite- project.eu/ontology/URBANITE_PROJECT", "title": {"@language": "en", "@value": "Calendar data Messina"}, "distribution": ["https://urbanite- project.eu/ontology/distribution/059fd3cc-63ws-4f3d-97de-fdgs4fdh2eg", "https://urbanite- project.eu/ontology/distribution/059fd3cc-43s3-4gds-w436-fhd45sdaf32"], "keyword": ["Calendar","Messina"] }, { "@id": "https://urbanite- project.eu/ontology/distribution/059fd3cc-63ws-4f3d-97de-fdgs4fdh2eg", "@type": "dcat:Distribution", "description": "Calendar data Messina year 2016 in NGSI-LD </pre>
--	--

	<pre> representation", "format": "http://publications.europa.eu/resource/authority/file- type/JSON_LD", "license": "http://publications.europa.eu/resource/authority/licence/CC_BY", "title": "Calendar data Messina 2016", "accessURL": "http://storageAPI-to-bedefined/2016" }, { "@id": "https://urbanite- project.eu/ontology/distribution/059fd3cc-43s3-4gds-w436-fhd45sdaf32", "@type": "dcat:Distribution", "description": "Calendar data Messina year 2016 in NGSI-LD representation", "format": "http://publications.europa.eu/resource/authority/file- type/JSON_LD", "license": "http://publications.europa.eu/resource/authority/licence/CC_BY", "title": "Calendar data Messina 2016", "accessURL": "http://storageAPI-to-bedefined/2016" }], "@context": { "name": {"@id": "http://xmlns.com/foaf/0.1/name"}, "homepage": {"@id": "http://xmlns.com/foaf/0.1/homepage"}, "accessURL": {"@id": "http://www.w3.org/ns/dcat#accessURL"}, "title": {"@id": "http://purl.org/dc/terms/title"}, "license": {"@id": "http://purl.org/dc/terms/license"}, "format": {"@id": "http://purl.org/dc/terms/format"}, "description": {"@id": "http://purl.org/dc/terms/description"}, "distribution": {"@id": "http://www.w3.org/ns/dcat#distribution", "@type": "@id"}, "keyword": {"@id": "http://www.w3.org/ns/dcat#keyword"}, "issued": {"@id": "http://purl.org/dc/terms/issued", "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}, "publisher": {"@id": "http://purl.org/dc/terms/publisher", "@type": "@id"}, "modified": {"@id": "http://purl.org/dc/terms/modified", "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}, "dct": "http://purl.org/dc/terms/", "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#", "xsd": "http://www.w3.org/2001/XMLSchema#", "rdfs": "http://www.w3.org/2000/01/rdf-schema#", "dcat": "http://www.w3.org/ns/dcat#", "foaf": "http://xmlns.com/foaf/0.1/", "dc": "http://purl.org/dc/elements/1.1/" } } </pre>	
Sample call		
<pre> curl -X GET "http://[server:port]/data/getCatalogueDatasets" -H "accept: application/json" </pre>		

8.3.5 searchDatasets (GET)

Table 15: API for the search and retrieval of dataset metadata

/searchDatasets	<p>Searches among the metadata of the existing dataset.</p> <p>It makes a search in typical metadata fields: <i>title</i>, <i>description</i> and <i>keyword</i>.</p> <p>All the <i>tags</i> search must be present in at least one of these fields.</p>
Method	

GET	
Input Params (* means mandatory)	
search	Tags to search for, separated by a space.
(query param)	Optional. If not set, all the metadata will be returned. Example: <i>Calendar Messina</i>
Success response	
200	<p>The method will return this code, and the JSON response will contain a JSON array with that metadata stored in the database that contain all the tags passed in the parameter <i>search</i>.</p> <p>Example:</p> <pre>[{ "id": "6bb9c361_177a635e86a", "metadata": { "@graph": [{ "@id": "https://urbanite-project.eu/ontology/URBANITE_PROJECT", "@type": "foaf:Organization", "homepage": "https://urbanite-project.eu/", "name": "URBANITE" }, { "@id": "https://urbanite-project.eu/ontology/data/Bilbao_Calendar", "@type": "dcat:Dataset", "description": {"@language": "en", "@value": "Calendar data Bilbao"}, "issued": "2021-05-12T10:36:46", "modified": "2021-05-12T15:36:46", "published": "https://urbanite-project.eu/ontology/URBANITE_PROJECT", "title": {"@language": "en", "@value": "Calendar data Bilbao"}, "distribution": ["https://urbanite-project.eu/ontology/distribution/a732f6c6-fcd8-4962-8aa9-db7d913a20ae", "https://urbanite-project.eu/ontology/distribution/059fd3cc-92b3-4f3d-97de-d050ae022eb5"], "keyword": ["Calendar", "Bilbao"] }, { "@id": "https://urbanite-project.eu/ontology/distribution/059fd3cc-92b3-4f3d-97de-d050ae022eb5", "@type": "dcat:Distribution", "description": "Calendar data Bilbao year 2016 in NGSI-LD representation", "format": "http://publications.europa.eu/resource/authority/file-type/JSON_LD", "license": "http://publications.europa.eu/resource/authority/licence/CC_BY", "title": "Calendar data Bilbao 2016", "accessURL": "http://storageAPI-to-be-defined/2016" }, { "@id": "https://urbanite-project.eu/ontology/distribution/a732f6c6-fcd8-4962-8aa9-db7d913a20ae", "@type": "dcat:Distribution", "description": "Calendar data Bilbao year 2015 in NGSI-LD representation", "format": "http://publications.europa.eu/resource/authority/file-type/JSON_LD", </pre>

	<pre> "license": "http://publications.europa.eu/resource/authority/licence/CC_BY", "title": "Calendar data Bilbao 2015", "accessURL": "http://storageAPI-to-beDEFINED/2015" }], "@context": { "name": {"@id": "http://xmlns.com/foaf/0.1/name"}, "homepage": {"@id": "http://xmlns.com/foaf/0.1/homepage"}, "accessURL": {"@id": "http://www.w3.org/ns/dcat#accessURL"}, "title": {"@id": "http://purl.org/dc/terms/title"}, "license": {"@id": "http://purl.org/dc/terms/license"}, "format": {"@id": "http://purl.org/dc/terms/format"}, "description": {"@id": "http://purl.org/dc/terms/description"}, "distribution": {"@id": "http://www.w3.org/ns/dcat#distribution", "@type": "@id"}, "keyword": {"@id": "http://www.w3.org/ns/dcat#keyword"}, "issued": {"@id": "http://purl.org/dc/terms/issued", "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}, "publisher": {"@id": "http://purl.org/dc/terms/publisher", "@type": "@id"}, "modified": {"@id": "http://purl.org/dc/terms/modified", "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}, "dct": "http://purl.org/dc/terms/", "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#", "xsd": "http://www.w3.org/2001/XMLSchema#", "rdfs": "http://www.w3.org/2000/01/rdf-schema#", "dcat": "http://www.w3.org/ns/dcat#", "foaf": "http://xmlns.com/foaf/0.1/", "dc": "http://purl.org/dc/elements/1.1/" } }] </pre>	
Sample call		
<pre> curl -X GET "http://[server:port]/data/search-targets?search=Bilbao%20Calendar" -H "accept: application/json" </pre>		

9 APPENDIX: Data Aggregation API

9.1 Types of Aggregators

Table 16: Types of available aggregators

Aggregator	Description	Interpolation
avg	Averages the data points	Linear Interpolation
count	The number of raw data points in the set	Zero if missing
dev	Calculates the standard deviation	Linear Interpolation
ep50r3	Calculates the estimated 50th percentile with the R-3 method *	Linear Interpolation
ep50r7	Calculates the estimated 50th percentile with the R-7 method *	Linear Interpolation
ep75r3	Calculates the estimated 75th percentile with the R-3 method *	Linear Interpolation
ep75r7	Calculates the estimated 75th percentile with the R-7 method *	Linear Interpolation

ep90r3	Calculates the estimated 90th percentile with the R-3 method *	Linear Interpolation
ep90r7	Calculates the estimated 90th percentile with the R-7 method *	Linear Interpolation
ep95r3	Calculates the estimated 95th percentile with the R-3 method *	Linear Interpolation
ep95r7	Calculates the estimated 95th percentile with the R-7 method *	Linear Interpolation
ep99r3	Calculates the estimated 99th percentile with the R-3 method *	Linear Interpolation
ep99r7	Calculates the estimated 99th percentile with the R-7 method *	Linear Interpolation
ep999r3	Calculates the estimated 999th percentile with the R-3 method *	Linear Interpolation
ep999r7	Calculates the estimated 999th percentile with the R-7 method *	Linear Interpolation
first	Returns the first data point in the set. Only useful for downsampling, not aggregation.	Indeterminate
last	Returns the last data point in the set. Only useful for downsampling, not aggregation.	Indeterminate
mimmin	Selects the smallest data point	Maximum if missing
mimax	Selects the largest data point	Minimum if missing
min	Selects the smallest data point	Linear Interpolation
max	Selects the largest data point	Linear Interpolation
none	Skips group by aggregation of all-time series.	Zero if missing
p50	Calculates the 50th percentile	Linear Interpolation
p75	Calculates the 75th percentile	Linear Interpolation
p90	Calculates the 90th percentile	Linear Interpolation
p95	Calculates the 95th percentile	Linear Interpolation
p99	Calculates the 99th percentile	Linear Interpolation
p999	Calculates the 999th percentile	Linear Interpolation
sum	Adds the data points together	Linear Interpolation
zimsum	Adds the data points together	Zero if missing

9.2 aggregate (GET)

/aggregate	Aggregate data from the database within a specific time range.
Method	
GET	

Input Params (* means mandatory)	
model* (path param)	Text, with the data type of the element to be updated. Must be one of the implemented data types, currently: <ul style="list-style-type: none"> • trafficFlowObserved
city* (path param)	Text, with the city (use case) to which the data correspond. Must be one of: <ul style="list-style-type: none"> • bilbao • messina • helsinki • amsterdam
Metric* (path param)	Text, with the metric. Must be one of the implemented currently: <ul style="list-style-type: none"> • intensity
startDate* (query param)	Date and time (ISO8601 ⁴⁰ UTC format) from which to get the data. Mandatory if parameter <i>endDate</i> is not present. Example: 2021-01-07T00:00:00.000Z
endDate* (query param)	Date and time (ISO8601 UTC format) until which to get the data. Mandatory if parameter <i>startDate</i> is not present. Example: 2021-01-08T22:45:00.000Z
Aggregator* (query param)	Text, with the type of aggregation function. Must be one of the aggregators described in 9.1 Types of Aggregators.
downsample (query param)	Text, with the downsample. The format must be a time interval followed by a function of aggregation. Example: 1h sum
tags (query param)	Text, tags in JSON format. They are the tags related to the OpenTSDB structure persisted. For the intensity metric, one tag is the 'id_spiral'. Example: { "id_spiral": "123" } // Filter by spiral called 123. Example: { "id_spiral": "*" } // Filter by all spirals.
Success response	

⁴⁰ <https://www.iso.org/iso-8601-date-and-time-format.html>

200	<p>If all the input parameters are correct, the method will return this code, and the response will be a JSON array with the list of the elements requested.</p> <p>Example:</p> <pre>[{ "metric": "trafficflowobserved.intensity", "tags": { "id_spiral": "123", "city": "bilbao" }, "aggregateTags": [], "dps": { "1613347200": 24, "1613350800": 24, "1613354400": 24, "1613358000": 60, "1613361600": 60, "1613365200": 108, "1613368800": 72, "1613372400": 540, "1613376000": 1018, "1613379600": 854, "1613383200": 785, "1613397600": 517, "1613401200": 998, "1613404800": 1074, "1613408400": 1301, "1613412000": 1152, "1613415600": 1105, "1613419200": 906, "1613422800": 426, "1613426400": 135, "1613430000": 57, "1613433600": 0 } }]</pre>
Error response	
400	<p>Bad Request.</p> <p>The method makes several checks of the parameters. It checks if the <i>model</i>, <i>city</i>, <i>metric</i>, <i>startDate</i>, <i>endDate</i> and <i>aggregator</i> parameters are the expected values. It also checks that at least one of the parameters <i>startDate</i> and <i>endDate</i> is set, and if so, checks that they are in ISO 8601 format.</p> <p>If any of these checks fail, a Bad Request code will be returned, and will return a JSON response, with just one field <i>Error</i>, that will contain a description of the error.</p> <p>Examples:</p> <pre>{ "Error": "No time range specified. 'startDate' and/or 'endDate' must be indicated." } { "Error": "Invalid value '2021/08/01' for parameter 'startDate'" }</pre>
Sample call	
<pre>curl -X GET "http://[server:port]/data/aggregate/trafficFlowObserved/bilbao/intensity? startDate=2021-02-15T00%3A00%3A00.000Z &endDate=2021-02-16T00%3A00%3A00.000Z &aggregator=SUM &downsample=1h-sum &tags=%7B%20%22id_spiral%22%3A%20%22123%22%7D"-H "accept: application/json"</pre>	

DRAFT VERSION