URBANITE

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

Deliverable D3.6

Data curation module Implementation v2

Editor(s):	TEC, ENG, FhG
Responsible Partner:	Fraunhofer FOKUS
Status-Version:	Final – v2.0
Date:	05.10.2022
Distribution level (CO, PU):	PU

Project Number:	GA 870338
Project Title:	URBANITE

Title of Deliverable:	Data curation 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, ENG, Fraunhofer FOKUS
Contributor(s):	TEC, ENG
Reviewer(s):	TECNALIA
Approved by:	All Partners
Recommended/mandatory readers:	WP5

Abstract:	This deliverable is the last version of two and presents the software implementation of the data curation module accompanied with the design specification and documentation. This deliverable is the result of Task 3.2.
Keyword List:	Curation, Preparation, Transformation, Data Management, Data Quality, 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
v0.1	16/07/2021	Draft ToC	FhG
v0.2	08/09/2021	First draft	FhG
v0.3	17/09/2021	Requirements and inclusion of subchapters in technical description	TECNALIA
v1.0	05/10/2021	Suggestions by reviewers	FhG
v1.1	26/08/2022	Content updated according to version 2	FhG
v1.2	31/08/2022	CaPe description	ENG
v1.3	09/09/2022	DataVaults description	FhG
v1.4	28/09/2022	Evaluation of CaPe integration	WAAG
V1.5	29/09/2022	Reordering and update of content	FhG
V1.6	30/09/2022	Finalisation of consent tool chapter	WAAG/FhG
V1.7	04/10/2022	Modifications asked for internal reviewer (Tecnalia)	FhG
V2.0	05/10/2022	Final version for submission	FhG/TECNALIA

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

API	Application Programming Interface
EC	European Commission
CC	Creative Commons
CPSV-AP	Common Public Service Vocabulary Application Profile
CSV	Comma Separated Values
DCAT	Data Catalogue Vocabulary
DCAT-AP	DCAT Application Profile
DPV	Data Privacy Vocabulary
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICT	Information and Communications Technology
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation Linked Data
MIF/MID	MapInfo Interchange Format
MQTT	MQ Telemetry Transport
NGSI	Next Generation Service Interface
NN	K-Nearest Neighbor
NGSI-LD	Next Generation Service Interface Linked Data
REST	Representational State Transfer
RDW	Specific Open Data Portal of Amsterdam
SOAP	Simple Object Access Protocol
SPDP	Standard for Publishing Dynamic Parking Data
UC	Use case
URL	Uniform Resource Locator
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

Executive Summary

This deliverable contains an overview over the software components that are related to the tasks of data manipulation between initial harvesting and storage. This includes, but is not limited to, the steps of data anonymization, preparation, transformation, and curation. Depending on the nature and quality of the harvested data none, some, or all of these steps could be necessary. The common goal regardless of the data's origin is the conversion into the applicable domain specific FIWARE Smart Data Model. These are described in deliverable D3.4 [1]. Only those components that were needed in achieving this for the data sources connected have been developed thus far.

This work is an update of D3.5. The update includes the curation modules and also more transformers, as well as more transformation scripts. For each existing component an overview along with a description is given. Where applicable, details on configuration and usage are provided. Additionally, a discussion of different tools for implementing data governance functions was added.

The components that are responsible for the aforementioned tasks integrate into the Piveau Pipeline Concept. Hence, this deliverable frequently references to deliverable D3.3 [2] when touching the theoretical background of this architectural concept. There, the Piveau Pipe is described thoroughly.

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 anonymization/preparation/transformation/curation, and data aggregation and storage. The deliverables D3.2, D3.5, and D3.7, together with their updated versions 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.

In this deliverable, a distinction is made among the terms “anonymization”, “preparation”, “transformation” and “curation”. Anonymization aims to address privacy protection by removing personally identifiable information from data sets, so that the people whom the data describe remain anonymous (this is compliant with GDPR regulations). Preparation refers to the process of ensuring a certain level of (meta-)data quality. [3] This includes detecting and removing false/implausible data, for example. Validating against a given specific schema could be one way of achieving this. Transformation is the conversion from one format into another, without altering the (meta-)data’s semantics. Data curation is considered the maintenance and enrichment of data after the previous steps have been completed. [4]

1.1 About this deliverable

Within the Data Management Platform, this deliverable focuses on the data anonymization, preparation, transformation, and curation. It presents the challenges involved in these steps, the proposed solution, and their implementation.

1.2 Document structure

Section 2.1 covers the functionalities provided by the anonymization, preparation, transformation, and curation components as well as how they fit into the general URBANITE architecture. This is followed by an overview of the available components and their technical functional descriptions in section 2.2. This includes the technical details that diverge from the ones discussed in deliverable D3.3. Next, section 3 contains instructions on how to build, configure, and run the application(s). The document wraps up with a conclusion and references.

1.3 Updates with respect to version 1

The main updates of this document in respect to version 1 consist of the added support of more data models the harvested data can be transformed into and new transformers to support additional data types that can be transformed into JSON. Additionally, a description into the exploration of two data governance tools was added.

2 Implementation

2.1 Functional description

The different kinds of data and metadata that have been harvested by the various importers or connectors (covered in D3.3) need to be sometimes anonymized (if they contain personal and/or sensitive information), and prepared/transformed/curated for further processing. After the data and metadata have been checked for quality/consistency, they are brought into a common format. The common format used in URBANITE is described in D3.4 [1]. Finally, they are stored in dedicated databases, which is covered in deliverable D3.8 [5].

The functional requirements for these components were listed in deliverable D5.8 [7]and a detailed design was provided in deliverable D5.5 [7]. We present here a short summary and the status of development. Most of the requirements applicable to the data models and datasets have been fulfilled or partially fulfilled. DC.04 was not fulfilled, as there was no data coming from outside of the platform that needed to be anonymized. DC.07 was not fulfilled, as the amount work that this would cost was out of scope. The following two tools are suggested as to provide the functionality for a manual check for suitable licenses: data.europa.eu Licensing Assistant[1] and the Joinup Licensing Assistant - Compatibility Checker[2]

Table 1: Status of the Requirements

Component	Requirements in D5.8	Current Status
Data Preparation	DC.07 Data license support. The module must check the data licenses and provide understandable information to the owners and the user of the data. For combined data sets with different licenses, it detects possible compatibility issues and informs users how to use and share the data	Not Fulfilled
	DC.05 Data validation and quality check. The data curation module must be able to validate the data provided by data harvesting module and its quality based on a defined format.	Covered. Quality checks are done on the data values and format for all data sources without distinguishing whether data is sensitive information or not.
Data transformation	DC.01 Data transformation after harvest. The harvested data may not be in a format and/or structure suitable for data storage. In this case, the data will need to be transformed in an automated way.	Fulfilled
	DC.03 Data Annotation. Data transformation module should add annotation in the form of metadata to	Fulfilled

[1] https://data.europa.eu/en/training/licensing-assistant

[2]https://joinup.ec.europa.eu/collection/eupl/solution/joinup-licensing-assistant/jla-compatibility-checker

	data to help the analysis. This metadata will be included in the data itself.	
	DC.06 Data Interoperability. Data transformation module should provide functionalities clean and annotate data to common semantics and data models, thus guaranteeing interoperability. It is important to note that there will not be one single common format that all data will be transformed into. Instead, established formats within the various domains will be targeted for transformation.	Fulfilled
	DC.08 Pipeline between data harvesting and curation modules. The data curation module must provide an API (REST service or MQTT endpoint) so that the data harvesting module can forward the data that has been retrieved.	Fulfilled
Data Curation	DC.02 Data Cleaning. Data curation module should be able to clean the data coming from the harvester eliminating duplicates or error.	Fulfilled
	DC.09 Data Cleaning. The data cleaning functionality must be capable of detecting and removing invalid or missing readings. The result should then be fit in terms of quality and type, for further processing	Fulfilled
	DC.09 Data Curation. This component is labelled as "triggered by user" in the architecture diagram. For this to be possible, it must feature an interface over which these functionalities are operated.	Fulfilled. All components offer an interface to configure and trigger them directly or through the scheduler.
Data anonymization	DC.04 Data anonymization. This module shall anonymize or pseudonymize data. Data anonymization could be done at the source or before storing it, depending on the use case. In any case, URBANITE platform will provide the anonymization functionality for users (UCs) to use it before the data is uploaded/used by the URBANITE platform.	Not covered, as the current prototype does not handle sensitive data. Data is provided anonymized.

2.1.1 Fitting into overall URBANITE Architecture

Like the harvesting modules, the components involved in data anonymization preparation, transformation, and curation are also part of the backend services of the URBANITE architecture. As such, the standalone modules follow a microservice approach making them a good fit with the Docker-based architecture designed in WP5. They also scale well, which is a key requirement when frequently processing potentially large amounts of data. The components that are described in this deliverable are highlighted in green in Figure 1, which comes from deliverable D5.8.

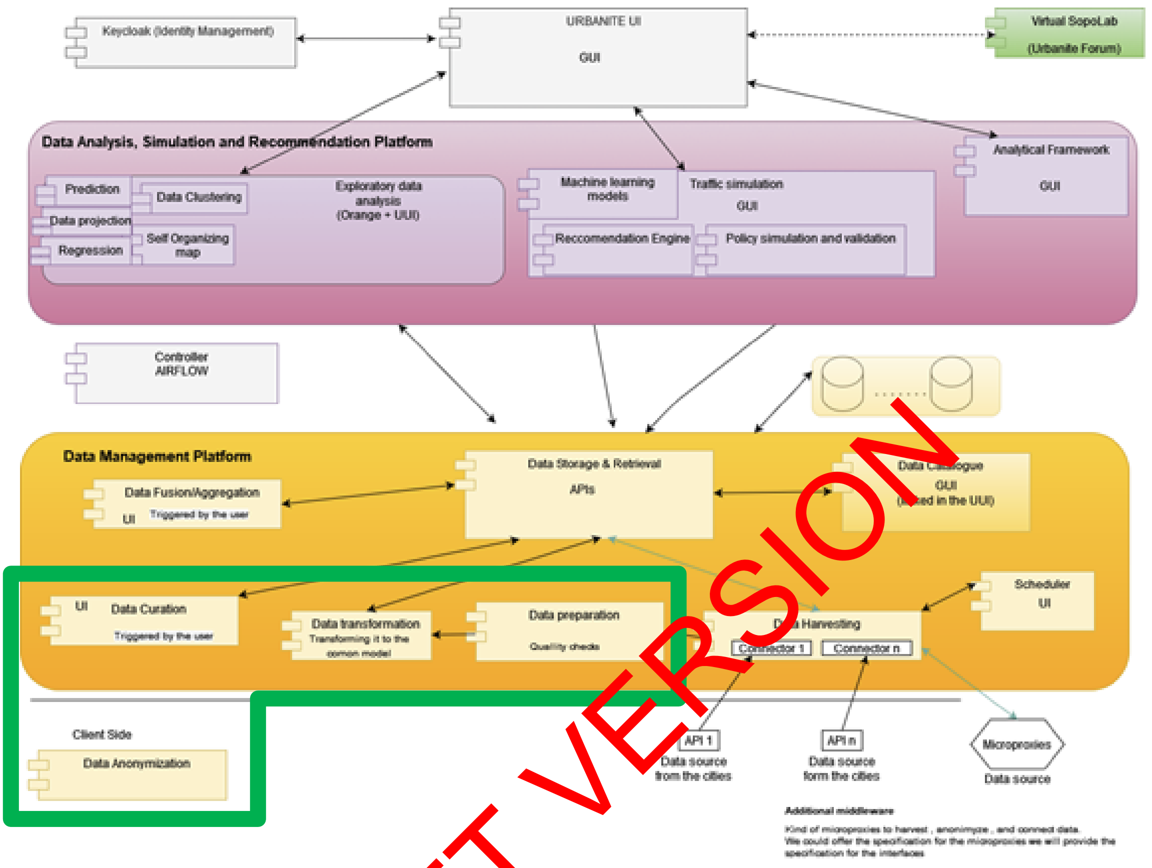


Figure 1: URBANITE architecture

2.2 Technical description

2.2.1 Data Preparation

Data Preparation refers to the process of ensuring a certain level of (meta-)data quality. According to ISO/TS 8000-1:2011[3], data quality is the "degree to which the characteristics of data satisfy stated and implied needs when used under specified conditions". Hence, data quality is not an independent concept. It can only be assessed meaningfully by considering the intended usage of the data and the context, in which the data is applied. For this reason, collaborative work between WP3, WP4 and WP6 has been carried out to identify the use case and information needs.

Before data is used by the algorithms and simulation models, we need to ensure that it meets certain quality criteria. Some of these quality checks are done in the data preparation or transformation steps, while others can be done over already stored data. The ISO Standard 25012:2008[4], defines fifteen quality aspects: Accuracy, Completeness, Consistency, Credibility,

[3] ISO (2011). ISO /TS 8000-1:2011 Data Quality – Part1: Overview.
[4] https://www.iso.org/standard/35736.html

Correctness, Accessibility, Compliance, Confidentiality, Efficiency, Precision, Traceability, Understandability, Availability, Portability, and Recoverability. Next, we describe the quality aspects analysed in URBANITE:

- **Accuracy**. It refers to error-free records that can be used as a reliable source of information. For example, we need to check whether the measurement is between its minimum and maximum possible values (e.g.: traffic intensity cannot be negative, an ambient temperature observation in a city is usually lower the 45 degrees, etc.), or pattern checks for strings or dates, as well as values being inside a set for days of the week, among others.

  As an example, in the data quality checks of the harvested data related to traffic flows, it was detected that occasionally some sensors sent negative intensities, which is not possible. Hence, all negative values are discarded and considered errors (Figure 2).

```
if (!entry.properties.Intensidad
   || entry.properties.Intensidad < 0) {
         continue;
}
```

Figure 2: Skipping empty/invalid readings in traffic flow

- **Completeness**. In this case, we need to check the number of available records in a specific range of time. For example, for traffic analysis, traffic flow data is aggregated in 5 and 15-minute periods. However, we cannot count on the fact that the sensors will always provide data every 5 minutes. So, we need to check the number of "holes" within the data to evaluate its quality. In addition, we also check the metadata completeness, i.e. that the main metadata attributes are available and if not, we complete them (if possible).
- **Consistency**. When aggregating data from multiple sources, we need to check if there is consistency in the measurement of variables throughout the datasets.
- **Precision.** Precision is the depth of knowledge encoded by the data, e.g. resolution of images, the degree of disaggregation of statistics or the number of decimals for numerical data.

2.2.2 Data Transformation

Data transformation is a key step in the Piveau Pipeline Concept. It cannot be expected that the municipalities provide their data in one of the common data models developed by FIWARE used in the URBANITE context. As such, the transformation of heterogeneous data sources into common models is vital for frictionless processing of the data henceforth. All architectural requirements and design constraints described in the respective section in D3.3 [2] also apply to the modules covered in this deliverable. Please refer to D3.3 for details on this topic. At time of writing a number of generic transformation related components have been developed. These are geared towards reusability and/or customizability. The aim here is that they can be employed for a wide variety of data formats, in order to keep the required effort of writing dedicated software for each new data source to a minimum. The following components have been developed:

- JSON to JSON

- CSV to JSON
- XLS(X) to JSON
- XML to JSON

These are covered in the next sections.

2.2.2.1 JSON to JSON

This transformer converts a given JSON structure into another JSON structure by means of JavaScript instructions.

In URBANITE, the harvested data is transformed into NGSI-LD format according to the data models defined in D3.4. For each data source, a different JavaScript file needs to be developed. It is also the responsibility of the transformer to adapt the data to the needs imposed by the NGSI-LD model, for example, date formats, value ranges, etc ...

The JavaScript file must cohere to some standards to ensure flawless evaluation. More precisely, it must feature a function named `transforming`, which takes a JSON object or array as its sole parameter. The function must return a JSON structure that is compliant with the URBANITE common data models based on FIWARE Smart Models. An example of what this can look like when transforming weather data is shown in Figure 3. The input would have to be a JSON object containing two fields, `data` and `metadata`. The former is transformed, the latter passed along as-is.

```
function transforming(input) {

  var output = {
    "@context": [
      "https://smartdatamodels.org/context.jsonld",
      "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
    ],
  };

  output.type = "WeatherObserved";
  output.source = "https://openweathermap.org/";
  output.temperature = input.data.main.temp;
  output.atmosphericPressure = input.data.main.pressure;
  output.windSpeed = input.data.wind.speed;
  output.windDirection = input.data.wind.deg;

  return {
    "metadata": input.metadata,
    "data": output
  };
}
```

Figure 3: Example of a JSON Transformation Script

In some cases, transformations need to be done not only to the format but also to the values. For example, percentage values from 0 to 100 may need to be adapted to ranges between 0 and 1; date formats may need to be transformed from string values to Epoch integers. In the case of Bilbao's air quality data, the date is harvested in the format `dd/mm/yyyy hh:mm` whereas the NGSI-LD format requires it as `yyyy-MM-dd'T'HH:mm:ss`. This means that a transformation, as shown in Figure 4 is required.

```
var datetime = lastMeasureDate.split(" ");
var datepart = datetime[0].split("/");
var timepart = datetime[1].split(":");
output.dateObserved = datepart[2] + "-" + datepart[1] + "-" +
datepart[0] + "T" + timepart[0] + ":" + timepart[1] + ":00";
```

Figure 4: Example of a datetime transformation

A complete example of a transformation script for air quality data from the Bilbao use case, including the JSON structure that is the output of the data harvester, the JavaScript file used for transformation, and the output of the transformer, which is a JSON in NGSI-LD format compliant to `airQualityObserved` FIWARE data model, is provided in the annex.

2.2.2.2 CSV to JSON

This transformer converts CSV into a JSON structure for further processing. Each row is mapped to a JSON array containing the field's values. An example of what this can look like is shown in Figure 5.

```
[
  ["ID", "Name", "Age"],
  ["1", "Jane", "25"],
  ["2", "John", "26"],
  ...
]
```

Figure 5: Example of CSV to JSON transformer

2.2.2.3 XSL(X) to JSON

This transformer converts XLS or XLSX files into a JSON structure for further processing. The data type must be configured in the applicable segment in the pipe descriptor. It can also be toggled whether to skip empty rows.


```
{
  "sheet_1": [
    [ "ID", "Name", "Age" ],
    [ "1", "Jane Doe", 29 ],
    [ "2", "John Doe", 35 ]
  ],
  "sheet_2": [
    ...
  ]
}
```

Figure 6: Example of XLS(X) to JSON transformer

2.2.2.4 XML to JSON

This transformer converts XML files into a JSON structure for further processing. The transformer utilises XSLT scripts to handle the XML files and to create the JSON structure.

2.2.3 Data Curation

Data curation is considered the maintenance and enrichment of data after the harvesting and transformation steps have been completed. Data curation has been focused on cleaning trajectory data based on GPS measurements.

2.2.3.1 Cleaning Trajectory Data

The GPS measurements obtained by affordable sensors can contain noise due to multiple reasons, among others:

- atmospheric and Ionospheric delays
- errors in the satellite and/or receptor clock
- multipath effect
- precision dilution
- selective Availability (S/A)
- anti-spoofing

Due to these effects, the obtained measurements do not match exactly with the real positions of the sensors. This effect is especially important close to intersections, auxiliary parallel roads, and road junctions. In the case of Bilbao, these sensors are tracking the bicycles from the renting city service. In general, the location obtained from the GPS allows finding the position of the bikes within the city; on many occasions, they are exact enough to locate the road where it is travelling. However, some errors can occur. Map-Matching processes do not only correct the measurement noise but also reconstruct the intermediate points, producing a complete set of locations between the origin and the final destination of the trajectory.

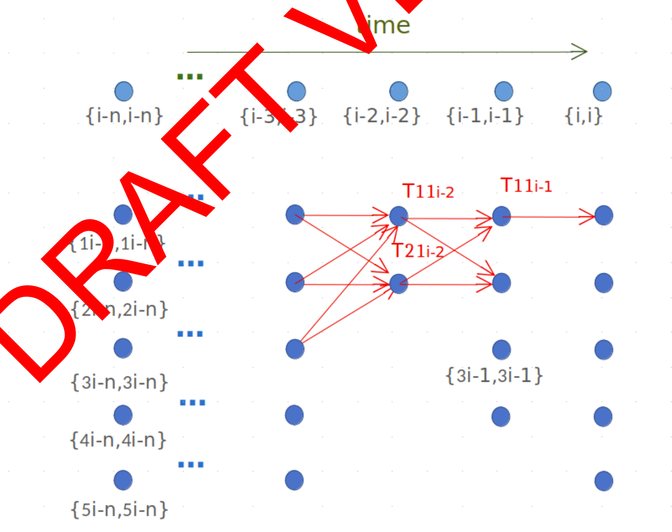


Figure 7: Trellis Diagram for the Hidden Markov Model used in the Map Matching Process.

The Map-Matching algorithm that we use in URBANITE consists in an independent implementation similar to the one introduced in [8]. This method works with GPS measurements and the timestamp at which each measurement is taken. From each of the measurements M_i a set of K possible candidates $\{C_i^k\}_{k=1}^{K}$ are obtained with the condition that these candidates correspond to points that define the roads. In general, the actual number of candidate values K changes from measurement to measurement depending on the density of roads close to M_i, i.e., if there is only a single road close to the measurement, then $K = 1$, and the candidate corresponds to the point of the road which is closest to the measurement.

To each of these candidate points, C_i^k, an emission probability is assigned, O^k_i. In URBANITE we have chosen this probability to follow a normal distribution of the straight-line distance between the measurement and the candidate:

Emission Probability: $O^k_i \sim exp\left\{-D_L(M_i, C_i^k)^2 \Big/ 2\,\sigma^2\right\}$

This probability captures the error within the measurement and basically means that a candidate is most probable if it is located closer to the obtained measurement. The easiest and simpler cleaning algorithms, like the known point-wise Map Matching, only use this probability to clean the trajectory measurements. In the case of URBANITE, it is also considered how probable it is to obtain a transition between candidates assigned to previous measurements. These transition probabilities, $T^{k\ l}_{i-1\ i}$, are assigned to candidate pairs, C_{i-1}^k, C_i^l, assigned to consecutive measurements *(i-1, i)* and they depend both on the straight-line distance $D_L\left(C_{i-1}^k, C_i^l\right)$, and the distance along the road network $D_R\left(C_{i-1}^k, C_i^l\right)$

Transmission Probability: $T^{k\,l}_{i-1\,i} \sim exp\left\{-\left[D_R\left(C_{i-1}^k, C_i^l\right) - D_L\left(C_{i-1}^k, C_i^l\right)\right]^2 \Big/ \beta \Delta T^2\right\}$

Using the emission and transmission probabilities, the overall probability of a sequence of candidate points can be computed in the Trellis Diagram (see Figure 7). The actual route corresponds to the trajectory that contains the candidate points that belong to the sequence with higher probability. This allows estimating not only the last point, but this algorithm has the ability of correcting also previous estimates using the new information, a posterior measurement.

As we mentioned before, within the transmission probability the distance along the road network is used. This implies that the algorithm estimates what is the most probable route between 2 candidates. This is achieved using the open-source routing service OSRM[5] which considers the shortest route.

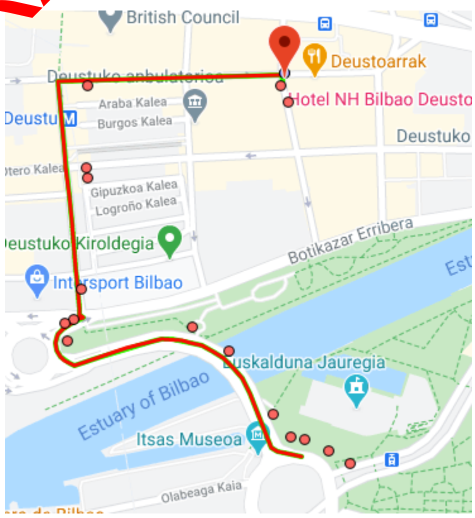


Figure 8: Result of the Map-Matching process applied to the bike GPS measurements.

[5] http://project-osrm.org/

2.2.3.2 Data imputation

In general, predictive models rely on data gathered by different types, which occasionally produce faulty readings due to several causes, such as malfunctioning hardware or transmission errors. Filling in those gaps is relevant for constructing accurate forecasting models, a task which is engaged by diverse strategies, from a simple null value imputation to complex spatio-temporal context imputation models.

Specifically, Orange[6], the tool extended in WP4, cannot handle unknown values in the input data, but provides specific widgets for data imputation: substitutes missing values with values either computed from the data or set by the user. By default, it applies the 1-NN method (hot desk imputation, where the value is taken from the most similar one).

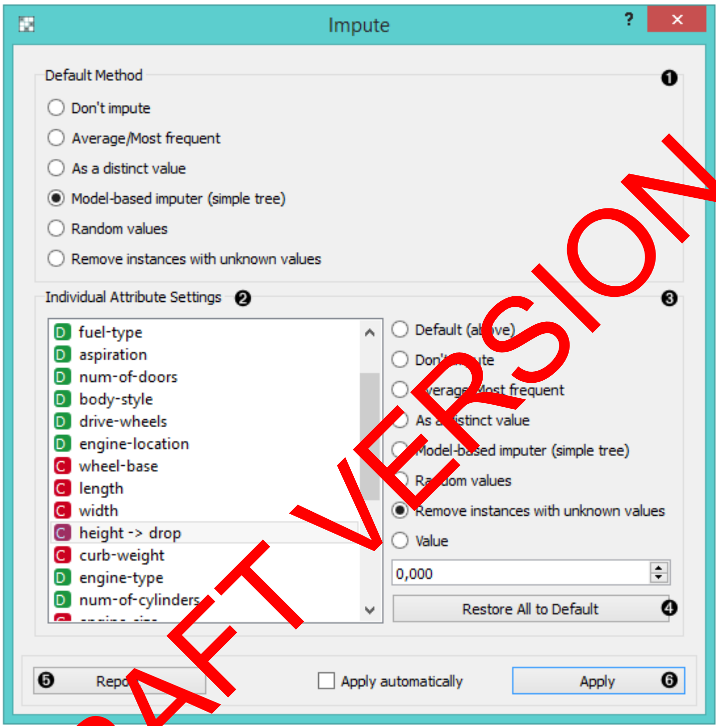


Figure 9: Widget of data imputation in Orange

In the top-most box, the Default method where the user can specify a general imputation technique for all attributes.

- Average/Most-frequent, uses the average value (for continuous attributes) or the most common value (for those discrete).
- As a distinct value, create new values to substitute the missing ones.
- Model-based imputer, construct a model for predicting the missing value, based on values of other attributes; a separate model is defined for each attribute of the dataset. The algorithm can be substituted by other options, but it must be considered the type of attributes, discrete or continuous.
- Random value, computes the distributions of values for each attribute and then imputes by picking random values from them
- Remove examples containing the missing values.

[6] https://orangedatamining.com/

It is possible to specify individual treatment for each attribute or specify a defined value manually, being applicable automatically in the data workflow.

2.2.4 Data Anonymization

The anonymization and pseud-anonymization of the data are key to the protection of privacy. A balance must be found between flexible solutions which are adaptable to each source and exploitation, but also easily manageable by the providers and users of this data, who not experts in ICT technologies. It is always a trade-off between anonymization and the loss of usefulness of the data in your application. Therefore, it is necessary for data providers and consumers to maintain an overview of the anonymization process and its subsequent implications.

ARX[7] is comprehensive open-source software for anonymizing sensitive personal data, implementing a simple three-step process. It provides support for all common privacy criteria, as well as arbitrary combinations. It uses a series of well-known, transparent and highly efficient anonymization algorithms. In addition, it implements a carefully selected set of techniques that can handle a wide spectrum of data anonymization tasks, while being intuitive and easy to understand. In addition, it presents a multiplatform user interface aimed at non-expert users, with high visualisation capacity, comparisons, etc. Finally, it provides a software library with an API, which facilitates integration with other components, as well as its use in isolation.

In order to cover a broad spectrum of privacy problems, it comes with implementations of commonly used privacy methods: k-Anonymity, that ensures that each register cannot be distinguished from at least k-1 other records regarding the quasi-identifiers defining groups of indistinguishable records (*equivalence class)*; k-Map, where the risks are calculated based on information about the underlying population, defined by the user or based on statistical frequency estimators; Average risk, that enforces a threshold on the average re-identification risk of the records; Population uniqueness, supported by statistical super-population models, is used to estimate characteristics of the overall population with probability distributions that are parameterized with sample characteristics (some methods can used as by Hoshino (Pitman), Zayatz and Chen and McNulty (SNB), other methods are: sample uniqueness, ℓ-Diversity, t-Closeness, δ-Disclosure privacy, β-Likeness, δ-Presence, Profitability or Differential privacy.

Additionally, the tool provides data quality analysis, estimating the utility of output data for the user scenarios, comparing the transformed dataset to the original input dataset, and statistics about the distribution of equivalence classes, suppressed records and other relevant metrics. Some attribute-level quality models implemented are: Precision, Granularity, Non-Uniform Entropy or Squared error.

Another tool analysed in the project is Amnesia[8], which provides functionalities similar to ARX in terms of the definition and application of replacing unique values or unique combinations of values, so that they are no longer identified, in a semi-automatic way.

Personal or sensitive data is processed in origin, and then stored on the platform. In order to carry out this prior anonymization, different tools such as those mentioned above have been provided to the data provider, and dedicated proxies have been provided on the platform.

[7] https://arx.deidentifier.org/

[8] https: //amnesia.openaire.eu/

2.2.5 Components description

Table 2 shows an overview over the preparation/curation/anonymization/transformation software that is available in URBANITE. All components except ARX have been developed by the consortium.

Table 2: Component Overview

	Name	Description
Data Quality (data preparation and curation)	Checks and processes	Quality checks and calculations that are embedded in the data harvesting component and as batch processes over stored data.
Transformer	CSV	Converts CSV into a JSON structure for further processing. Each row is mapped to a JSON array containing the field's values.
	XLS(X)	Converts XLS or XLSX files into a JSON structure for further processing. The data type must be configured in the applicable segment in the pipe descriptor. It can also be toggled whether to skip empty rows.
	JSON	Converts a given JSON structure into another JSON structure by means of JavaScript instructions. The script can be managed using Git (see transformation scripts below).
	XML	Converts a given XML file into a JSON structure for further processing by using a XSLT script. The script can be managed using Git (see transformation scripts below).
Misc.	Transformation Scripts	A simple GitLab repository that contains transformation scripts for use with the JSON transformer.
Data Curation	Algorithms	A map-matching method to reduce noise in GPS measurements.
Anonymization tool	ARX, Amnesia	It provides relevant data anonymization and pseudo-anonymization algorithms and supporting methods for the estimation of the data quality and usefulness of the outputs.

2.2.6 Data Governance

The most critical of all challenges in data sharing overarching all others is a lack of trust. Most people believe that data is a valuable commodity, but they are not ready to share own or use data from others if the data sharing is not organized in a meaningful way. People want to have more control over what, to whom, for which purpose and under which conditions they share.

The topics of trusted data sharing and managing the consents of individual contributors are out of scope of the URBANITE project, but they are relevant if in future individuals may become data suppliers in URBANITE. Therefore, we looked on two relevant initiatives focused on the topics and considered the ways of possible collaboration with them. A description of how a possible integration of the respective tool could work follows. A more in-depth explanation for each tool can be found in

APPENDIX: Data Governance.

2.2.6.1 Practical integration of the Cape consent tool with services

2.2.6.1.1 Introduction

The Amsterdam use-case in URBANITE researches ways to engage residents in the process of urban mobility transformation. One of those ways is the creation of cycling data commons. We at Waag see this as a more responsible approach to the use of citizens' data. Don't just use data from citizens, use it with them.

The need for a different approach is described well in *Reclaiming the Smart City: personal data, trust, and the new commons*[9]. In a data commons data becomes a shared resource that enables citizens to contribute, access, and use the data—for instance about air quality, mobility, or health—as a common good, without intellectual property rights restrictions. This follows Elinor Ostrom's[10] assertions that collective resources (such as citizens' mobility data) should be governed by nonprofit and voluntary actions, rather than by only governments and/or the private sector.

Currently, people have little to say over how their personal data gets collected and used. The successful cases of technical solutions that allow policymakers to acquire people's data in a more consent-driven way are non-existent. There are however several experiments going on. One of such experiments is the cyclist data commons within URBANITE.

We want to investigate whether we can arrange the consent between citizens and the partners of the Amsterdam use case. We want to see how we can exchange data between cyclists, a commercial initiative Ring-Ring, the NGO Fietsersbond, the gemeente A'dam and the platform Urbanite.

For this we explored the Cape Consent tool. To study the way it works and the consequences we don't want to do this with the actual 'real services' immediately. So, we pretend to be Ring-Ring and Urbanite by instantiating mock services. (These are two barebones web services that only implement the minimum set of API calls needed for Cape, with a fixed set of data)

CaPe doesn't allow transfer of data, nor does it provide a catalogue of what data is available or the ability to delegate the consent. In combination with DataVaults, solid pods and/or IDS connectors this might be possible, and this is subject to further research.

[9]https://decodeproject.eu/publications/reclaiming-smart-city-personal-data-trust-and-new-commons.html

[10] Ostrom, E. (1990). *Governing the commons: The evolution of institutions for collective action*. New York: Cambridge University Press.

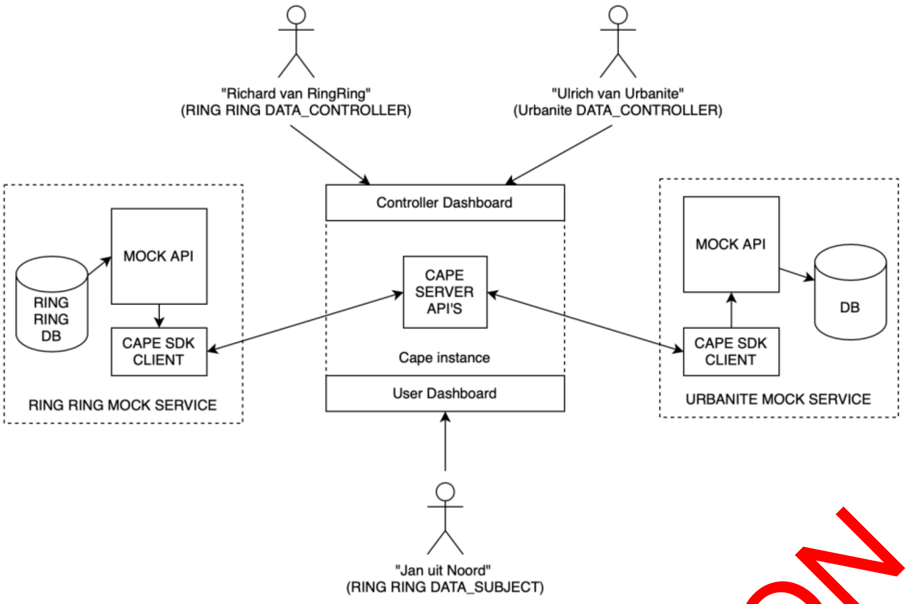


Figure 10: Overview. NB: the actual data flows are not indicated in this figure.

2.2.6.1.2 CaPe scenario

1. Richard (data controller of Ring-Ring) executes the Ring-Ring (individual) data service registration in cape. This includes to enter texts that describe well what the service does, and probably come back in the consent form. Also, Richard enters URLs and configuration to match Ring-Ring's mock service. Ulrich registers urbanite platform service in cape. This includes to enter texts that describe well what the service does, and probably come back in the consent form. Ulrich does the same things to match urbanite's mock service.

2. Jan links the Ring-Ring mock service on the user dashboard.

1. user centered vs delegated consent scenario

User 'Jan uit Noord' has cycling data at Ring-Ring. User 'Ulrich van Urbanite' wants access to this data. This access can be granted via cape after permission from Jan. (the data flows will be arranged differently, e.g. via DataVaults).

3. Ulrich makes a consent request for (Jan's) Ring-Ring data. Jan gives consent by means of the consent form.
4. Ulrich makes a data request via Cape (programmatically). Data (1 or more records from Jan) is read from the Ring-Ring mock service and stored in Urbanite mock service

2. delegated consent scenario

For the purpose of the Amsterdam use-case, we are also exploring a delegated consent Scenario, as we see that individuals are often not independently capable of managing the consent rules themselves but would rather delegate their consent to parties they trust. This partees are sometimes also called data trustees.

As CaPe doesn't offer the capabilities for this, we are researching how different tools could be combined.

2.2.6.2 Practical integration of the DataVaults platform with the Urbanite platform

An alternate tool for the mentioned use-case would be DataVaults, a platform that allows individual users, called Data Provider in the DataVaults context, to take ownership and control of their data and share them at will, through flexible data sharing and fair compensation schemes with other entities (companies or not). DataVaults uses the Piveau pipeline to ingest data, similar to Urbanite, but with different resulting formats.

2.2.6.2.1 Requirements for the integration

For an integration, a new importer would be needed that can import the desired data, e.g. from Ring-Ring, into the DataVaults platform. Furthermore, another importer would be needed to transfer the data that was shared by a user into the Urbanite platform.

Also, processes would be needed to guide a Ring-Ring user to the DataVaults platform where they can share their data and to automate the collection and transfer to the Data Seeker. It would be preferable, if an integration of the interactions into the corresponding platforms could be offered, as this would increase the number of possible users, especially Data Providers. If this is not the case, users would need to move to the DataVaults platform for sharing and purchasing.

2.2.6.2.2 Data Provider interactions

For this scenario, the user Jan wants to share his Ring-Ring cycling data to Urbanite. It is assumed that an importer does exist and is already configured, so the user Richard will not be used in this scenario.

1. Jan registers an account on the DataVaults Personal App.
2. He creates a new source collection with the Ring-Ring importer, for which he configures this importer to use Jan's Ring-Ring account to fetch his data. This collection process can be automated and scheduled based on the initial configuration.
3. Jan creates a sharing configuration for this source collection, where he defines the grade of the anonymization, visibility levels and the price for his data. This configuration can be executed once or on a schedule
4. The shared data will be transferred to the DataVaults Cloud Platform where Data Seekers can search for this data and purchase it
5. Jan then receives a notification, that someone wants to purchase his shared data and can either accept this or reject it.

2.2.6.2.3 Data Seeker interactions

The same interaction from the Data Seeker perspective would look like the following. For this the user Ulrich is going to purchase anonymized Ring-Ring data to be imported into DataVaults.

1. Ulrich searches for data that was harvested from Ring-Ring and finds the data set shared by Jan
2. He requests to purchase it, which Jan accepts
3. Ulrich then is able to download the data and import it into Urbanite. If a tighter integration between both services exist, the data can directly be imported without having to download it first

2.2.7 Technical specifications

Like the harvesting components described in deliverable D3.3 the data preparation and data transformation components covered in this document are also part of the Piveau Pipe concept. As such, all technical details described in the respective section in D3.3 also apply to these modules, i.e. they are written in Java and are based on the Vert.X[11] framework developed by the Eclipse foundation. The pipe functionality (parsing and manipulating the pipe descriptor) is provided by the Piveau Pipe Model library. The common endpoint each component exposes is implemented by the Piveau Pipe Connector library. Please, refer to D3.2 for more details on the Piveau Pipe concept.

One special case is the JSON transformer. It features a JavaScript engine for running the transformation scripts. In order to cover as many language features as possible, it relies on the GraalVM[12] for providing a more sophisticated JavaScript engine than provided in the default JVM. Work is on the way to mitigate this dependency.

Besides, the data cleaning methods with application to trajectories, used for data curation, are implemented in Java, with invocations to the open-source routing service OSRM[13].

Finally, the ARX library for data anonymization is implemented in Java providing a UI. To use the basic features of ARX, the following libraries must be included: Colt, HPPC, Commons math, JHPL Newton-Raphson library, Commons validator; for the utility estimation: exp4j, Apache Mahout and SMILE, to add support for some machine learning algorithms. As previously mentioned, ARX offers a public API, whose documentation provides a detailed description of the different components and interfaces for loading data, defining data transformations, altering and manipulating data and processing the results of the algorithm. On the other side, Amnesia, presents three ways of accessing services: an online version, not recommended when managing personal data, running the application or by cloning and compiling the source code; for all them, it is proving a set of ReST API endpoints.

The backend of Amnesia is implemented in Java using the Spring framework. Its components offer a ReST API that handles anonymization requests issued by the web interface. It uses a temporary local storage for the anonymization purposes and final results are returned by the ReST interface.

[11] https://vertx.io/
[12] https://www.graalvm.org/
[13] http://project-osrm.org/

3 Delivery and usage

3.1 Package information

See the respective section in deliverable D3.3 for information about packaging.

3.2 Installation instructions

In order to integrate well into the URBANITE platform, all components will be available as Docker images. However, before building the Docker images the corresponding JAR or WAR file needs to be created. The deployment of a service can be achieved using the three commands below. Note that curly brackets indicate that applicable values need to be substituted.

```
$> mvn clean package

$> docker build -t urbanite/{component-name} .

$> docker run -p {PORT}:8080 urbanite/{component-name}
```

Depending on the respective component a certain configuration may need to be applied, for example, an API key. This can be achieved using environment variables, which can be passed to Docker containers like so:

```
$> docker run -e {ENV_VAR}={value} urbanite/{component-name}
```


3.3 User Manual

See the respective section in deliverable D3.2 for general information about where to find user manuals and API specifications.

3.3.1 JSON to JSON Transformer

For the JSON to JSON or the XML to JSON transformer, transformation scripts can be made available in three ways. Which one is applicable for the respective pipelines has to be configured in the Piveau Pipe descriptor. An overview over the available options is shown in Table 3.

Table 3: JSON Transformer Configuration

Method	Description	Sample Configuration

embedded	The script is integrated into the pipe descriptor.	In Pipe descriptor:
		{
		"scriptType": "embedded",
		"script": "function transforming(input) { … }"
		}

localFile	The script is placed in the scripts folder of the application before compilation.	In Pipe descriptor:
		{
		"scriptType": "localFile",
		"path": "example.js"
		}

repository	The script resides in a Git repository. The component frequently polls the repository for any changes. Requires configuration at	In Pipe descriptor:
		{
		"scriptType": "repository",
		"path": "example.js"
		}
		
		Environment:
		GIT_URI: https://gitlab.com/scripts.git

	application level and in the Pipe descriptor.	`GIT_USER_NAME: urbanite_service_account`
		`GIT_TOKEN: myAccessToken`
		`GIT_BRANCH: master`

3.4 Licensing information

The software developed in this project is released under AGPLv3[14]. Piveau consus is available as licensed under Apache 2.0.

3.5 Download

The components that are included in the pipeline, i.e., data harvesting (described in D3.3), data preparation and data transformation are available in the GitLab maintained by Tecnalia[15]. ARX, as external tool, is available for download from the official site[16] and is maintained at GitHub[17]. Amnesia is also an external tool for data anonymization, available for downloading[18] and a public repository[19].

[14] https://www.gnu.org/licenses/agpl-3.0.en.html
[15] https://git.code.tecnalia.com/urbanite/private/wp3-data-management/harvester
[16] https://arx.deidentifier.org/downloads/
[17] https://github.com/arx-deidentifier/arx
[18] https://amnesia.openaire.eu/download.html
[19] https://github.com/dTsitsigkos/Amnesia

4 Conclusion

Overall, this document describes the technical details of the components involved in the preparation, transformation, curation and anonymization of data. Data curation, in this document considered to be the enrichment and maintenance of data, is covered in the v2 release of the components related to task 3.3. For anonymization the ARX software is presented but not integrated, as there is no current need for anonymization. All data is provided in anonymized form, where applicable various transformers have been developed: CSV to JSON, XLS(X) to JSON, JSON to JSON, and XML to JSON. While the former two follow a static rule for conversion, the JSON to JSON transformer is customizable by means of JavaScript instructions. And the XML to JSON transformations can be configured using XSLT scripts. This makes the latter two fit for tasks like the conversion into the common FIWARE models discussed in deliverable D3.4. Minor data preparation tasks are, if required by the individual data sources, directly handled in the importers/connectors. Due to the flexible design of the Piveau Pipe Concept outlined in D3.3, dedicated components handling the preparation of data could easily be integrated if required.

Two tools providing data governance functionality were investigated in this work, DataVaults and CaPe. Both tools seem to provide a promising way to coordinate the consent process for the usage of an individual's data. DataVaults also includes the ability to share the corresponding data, whereas the usage of CaPe would necessitate an additional service for that functionality. However, it seems that the integration of CaPe into the Urbanite platform would be more straightforward than the connection of Urbanite with DataVaults. But a good amount of work would have to be put into both solutions. Both paths seem to be promising but a more in-depth comparison before moving forward with any solution would be sensible.

In conclusion, this deliverable allows the reader to get an understanding of the technical solution(s) employed for the intermediated steps of data curation, preparation, transformation, anonymization, and data governance before storage, as well as how to build and deploy the components involved.

5 References

[1] T. E. FhG, „URBANITE data structure and semantic model specification," 2020.

[2] T. E. FhG, „Data harvesting module and connectors implementation-v2," 2022.

[3] D. Pyle, Data preparation for data mining, morgan kaufmann, 1999.

[4] M. H. Cragin, P. B. Heidorn, C. L. Palmer und L. C. Smith, „An Educational Program on Data Curation," *STS Conference Poster Session,* 25 June 2007.

[5] T. E. FhG, „Data aggregation and storage module implementation-v2," 2022.

[6] F. E. J. TEC, „URBANITE Ecosystem-v2," 2022.

[7] F. E. J. TEC, „URBANITE architecture v2,“ 2022.

[8] P. a. K. J. Newson, „Hidden Markov Map Matching Through Noise and Sparseness,“ in *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2009), November 4-6, Seattle, WA*, 2009.

[9] C. M. E. B. M. F. TEC, „URBANITE Mobility Data Sources Analysis,“ European Commission, 2020.

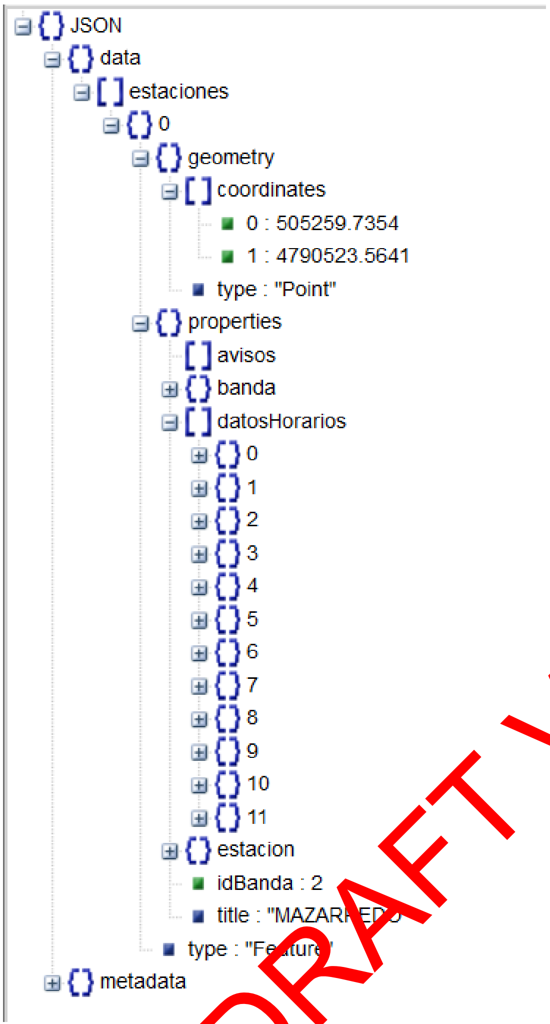
[10] F. E. J. TEC, „Detailed requirements specification,“ 2021.

[11] E. F. TEC, „Data curation module implementation v1,“ 2021.

6 APPENDIX: Data transformation example

This section provides an example of a transformation script for air quality data in Bilbao Use Case.

The structure of the json that is the output of the data harvester is:



The JavaScript file used for the transformation is:

```
function transforming(input) {
  var result = [];
  var data = input.data;
  var stations= data.estaciones;
  for (var i in stations) {
    var station = stations[i];
    var output = {
      "@context": [
        "https://smartdatamodels.org/context.jsonld",
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
      ],
    };
    var lastMeasureDate = station.properties.datosHorarios[0].fechaHoraUltimaMedidaHoraria;

    var procesedDate = lastMeasureDate.replaceAll(" ","");
```

```
    procesedDate = procesedDate.replaceAll("/","");
    procesedDate = procesedDate.replaceAll(":","");
    output.id = "urn:ngsi-ld:AirQualityObserved:" +
station.properties.estacion.codigoEstacion+":"+procesedDate;//confecha

    output.type = "AirQualityObserved";

    var loc = new Object();
    var coordinates = [];
    coordinates.push(station.properties.estacion.coordenadaLat);
    coordinates.push(station.properties.estacion.coordenadaLon);
    loc.coordinates = coordinates;
    loc.type = "Point";
    output.location = loc;
    //yyyy-MM-dd'T'HH:mm:ss
    var  datetime = lastMeasureDate.split(" ");
    var datepart = datetime[0].split("/");
    var timepart = datetime[1].split(":");
    output.dateObserved =datepart[2]+"-"+datepart[1]+"-
"+datepart[0]+"T"+timepart[0]+":"+timepart[1]+":00";//lastMeasureDate;
    var contaminantes = station.properties.datosHorarios;
    for (var icont in contaminantes) {
      var contaminante = contaminantes[icont];
      var nombreContaminante = contaminante.contaminante.nombreContaminante;
      switch (nombreContaminante) {
        case "NO":
          output.no = contaminante.ultimoValorHorario;
          break;
        case "CO":
          output.no = contaminante.ultimoValorHorario;
          break;
        case "NO2":
          output.no2 = contaminante.ultimoValorHorario;
          break;
        case "NOX":
          output.nox = contaminante.ultimoValorHorario;
          break;
        case "PM10":
          output.pm10 = contaminante.ultimoValorHorario;
          break;
        case "SO2":
          output.so2 = contaminante.ultimoValorHorario;
          break;
        default:
          break;
      }
    }

    result.push(output);
  }

  return {
    "metadata": input.metadata,
    "data": result
  };
}
```

And the output of the transformer, that is a JSON in NGSI-LD format compliant to `airQualityObserved` FIWARE data model is:


```
{
 "metadata" : {
  "@graph" : [ {
   "@id" : "_:b0",
   "@type" : "foaf:Organization",
   "homepage" : "https://urbanite-project.eu/",
   "name" : "URBANITE"
  }, {
   "@id" : "_:b1",
   "@type" : "accessRights",
   "label" : "public"
  }, {
   "@id" : "http://urbanite-project.eu/ontology/dataset/bilbao_airquality_JULY_2021",
   "@type" : "dcat:Dataset",
   "accessRights" : "_:b1",
   "description" : {
    "@language" : "en",
    "@value" : "Air Quality information for Bilbao, downloaded at 2021-07-21."
   },
   "issued" : "2021-07-21T07:07:58.553410200Z",
   "modified" : "2021-07-21T07:07:58.553410200Z",
   "publisher" : "_:b0",
   "title" : {
    "@language" : "en",
    "@value" : "Bilbao Air Quality for 2021 7"
   },
   "distribution" : "http://urbanite-project.eu/ontology/distribution/airquality/bilbao_2021721",
   "keyword" : [ "Bilbao", "Air Quality", "2021", "JULY" ],
   "theme" : [ "http://publications.europa.eu/resource/authority/data-theme/REGI",
"http://publications.europa.eu/resource/authority/data-theme/TRAN" ]
  }, {
   "@id" : "http://urbanite-project.eu/ontology/distribution/airquality/bilbao_2021721",
   "@type" : "dcat:Distribution",
   "description" : {
    "@language" : "en",
    "@value" : "NGSI-LD representation of FIWARE Air Quality data model bilbao_210720210500"
   },
   "format" : "http://publications.europa.eu/resource/authority/file-type/JSON",
   "license" : "http://publications.europa.eu/resource/authority/licence/CC_BY",
   "title" : {
    "@language" : "en",
    "@value" : "JSON-LD"
   },
   "accessURL" :
"https://urbanite.esilab.org:8443/data/getTDataRange/airQualityObserved/bilbao?startDate=2021-
7-21T00%3A00%3A00.000Z&endDate=2021-7-21T23%3A59%3A00.000Z"
  } ],
  "@context" : {
   "publisher" : {
    "@id" : "http://purl.org/dc/terms/publisher",
    "@type" : "@id"
   },
   "accessRights" : {
```

```
    "@id" : "http://purl.org/dc/terms/accessRights",
    "@type" : "@id"
  },
  "keyword" : {
    "@id" : "http://www.w3.org/ns/dcat#keyword"
  },
  "theme" : {
    "@id" : "http://www.w3.org/ns/dcat#theme",
    "@type" : "@id"
  },
  "modified" : {
    "@id" : "http://purl.org/dc/terms/modified",
    "@type" : "http://www.w3.org/2001/XMLSchema#dateTime"
  },
  "issued" : {
    "@id" : "http://purl.org/dc/terms/issued",
    "@type" : "http://www.w3.org/2001/XMLSchema#dateTime"
  },
  "description" : {
    "@id" : "http://purl.org/dc/terms/description"
  },
  "title" : {
    "@id" : "http://purl.org/dc/terms/title"
  },
  "distribution" : {
    "@id" : "http://www.w3.org/ns/dcat#distribution",
    "@type" : "@id"
  },
  "name" : {
    "@id" : "http://xmlns.com/foaf/0.1/name"
  },
  "homepage" : {
    "@id" : "http://xmlns.com/foaf/0.1/homepage"
  },
  "label" : {
    "@id" : "http://www.w3.org/2000/01/rdf-schema#label"
  },
  "license" : {
    "@id" : "http://purl.org/dc/terms/license",
    "@type" : "@id"
  },
  "format" : {
    "@id" : "http://purl.org/dc/terms/format",
    "@type" : "@id"
  },
  "accessURL" : {
    "@id" : "http://www.w3.org/ns/dcat#accessURL",
    "@type" : "@id"
  },
  "schema" : "http://schema.org/",
  "dcatap" : "http://data.europa.eu/r5r/",
  "adms" : "http://www.w3.org/ns/adms#",
  "spdx" : "https://spdx.org/rdf/terms/#",
  "gsp" : "http://www.opengis.net/ont/geosparql#",
  "owl" : "http://www.w3.org/2002/07/owl#",
  "org" : "http://www.w3.org/ns/org#",
```

```
    "xsd" : "http://www.w3.org/2001/XMLSchema#",
    "skos" : "http://www.w3.org/2004/02/skos/core#",
    "rdfs" : "http://www.w3.org/2000/01/rdf-schema#",
    "hydra" : "http://www.w3.org/ns/hydra/core#",
    "dct" : "http://purl.org/dc/terms/",
    "v" : "http://www.w3.org/2006/vcard/ns#",
    "time" : "http://www.w3.org/2006/time#",
    "dcat" : "http://www.w3.org/ns/dcat#",
    "odrl" : "https://www.w3.org/TR/odrl-vocab/#",
    "locn" : "http://www.w3.org/ns/locn#",
    "prov" : "http://www.w3.org/ns/prov#",
    "foaf" : "http://xmlns.com/foaf/0.1/",
    "gmd" : "http://www.isotc211.org/2005/gmd#"
  }
 },
 "data" : [ {
   "@context" : [ "https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld" ],
   "id" : "urn:ngsi-ld:AirQualityObserved:62:210720210500",
   "type" : "AirQualityObserved",
   "location" : {
    "coordinates" : [ 43.26750551179745, -2.935188110338201 ],
    "type" : "Point"
   },
   "dateObserved" : "2021-07-21T05:00:00",
   "no" : 0,
   "no2" : 8,
   "nox" : 8,
   "pm10" : 33.08,
   "so2" : 0
 }, {
   "@context" : [ "https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld" ],
   "id" : "urn:ngsi-ld:AirQualityObserved:3:210720210500",
   "type" : "AirQualityObserved",
   "location" : {
    "coordinates" : [ 43.25491056506432, -2.902376115931137 ],
    "type" : "Point"
   },
   "dateObserved" : "2021-07-21T05:00:00",
   "no" : 1,
   "no2" : 9,
   "nox" : 11,
   "pm10" : 31.71,
   "so2" : 3
 }, {
   "@context" : [ "https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld" ],
   "id" : "urn:ngsi-ld:AirQualityObserved:12:210720210500",
   "type" : "AirQualityObserved",
   "location" : {
    "coordinates" : [ 43.245553729819484, -2.960475856567045 ],
    "type" : "Point"
   },
   "dateObserved" : "2021-07-21T05:00:00",
   "no" : 3,
```

```
    "no2" : 10,
    "nox" : 14,
    "pm10" : 24.75,
    "so2" : 4
  }, {
    "@context" : [ "https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-
ld-core-context.jsonld" ],
    "id" : "urn:ngsi-ld:AirQualityObserved:8:210720210500",
    "type" : "AirQualityObserved",
    "location" : {
      "coordinates" : [ 43.26522199716327, -2.9475439999557134 ],
      "type" : "Point"
    },
    "dateObserved" : "2021-07-21T05:00:00"
  }, {
    "@context" : [ "https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-
ld-core-context.jsonld" ],
    "id" : "urn:ngsi-ld:AirQualityObserved:7:210720210500",
    "type" : "AirQualityObserved",
    "location" : {
      "coordinates" : [ 43.28099012364825, -2.953278416670214 ],
      "type" : "Point"
    },
    "dateObserved" : "2021-07-21T05:00:00"
  }, {
    "@context" : [ "https://smartdatamodels.org/context.jsonld", "https://uri.etsi.org/ngsi-ld/v1/ngsi-
ld-core-context.jsonld" ],
    "id" : "urn:ngsi-ld:AirQualityObserved:6:210720210500",
    "type" : "AirQualityObserved",
    "location" : {
      "coordinates" : [ 43.25880286639215, -2.945656664175787 ],
      "type" : "Point"
    },
    "dateObserved" : "2021-07-21T05:00:00",
    "no" : 5,
    "no2" : 14,
    "nox" : 21,
    "pm10" : 31.71,
    "so2" : 1
  } ]
}
```

7 APPENDIX: Data Governance

7.1 DataVaults

DataVaults[20] is a Horizon2020 project, which aims to deliver a framework and a platform that has personal data, coming from diverse sources in its center and that defines secure, trusted and privacy preserving mechanisms allowing individuals to take ownership and control of their data and share them at will, through flexible data sharing and fair compensation schemes with other entities (companies or not). The overall approach rejuvenates the personal data value chain, which could from now on be seen as a multi-sided and multi-tier ecosystem governed and regulated by smart contracts which safeguard personal data ownership, privacy and usage and attributes value to the ones who produce it.

DataVaults aims to establish a way to improve the data management, tackling the collection, formalization, storage, sharing and access control of these data in order to obtain a value. This value (or even profit) may be economic in some cases but also can improve other different aspects of society, for example, using analytics algorithms to understand the way of how the people move around the city.

DataVaults covers the personal data value chain. It includes the collection of data from different sources in a secure personal DataVault, anonymization and other processing of these data for sharing, storing, sharing and analyzing the prepared data on the *Data Seeker's* request enabling her/him to get insights and finally receiving apart of the value back from the *Data Seeker.*

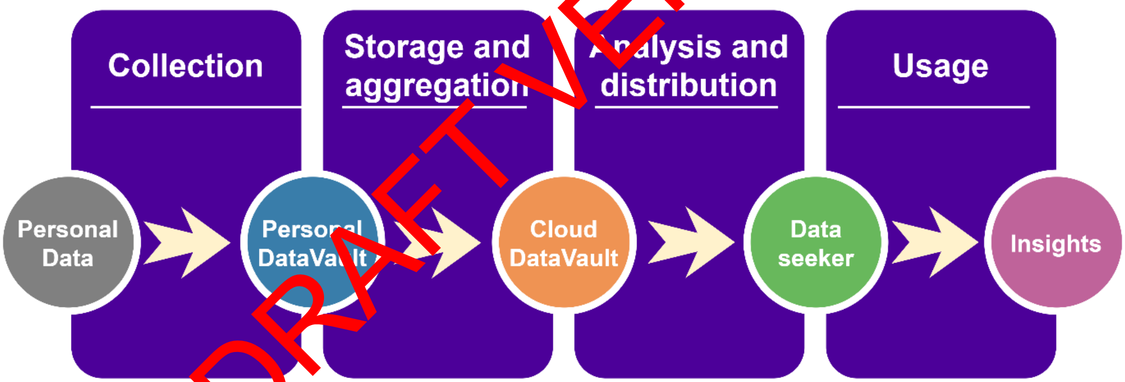


Figure 11 - DataVaults data value chain

Whenever an *Individual* decides to make data available to *Data Seekers* through the platform, s-/he is prompted to define various details regarding data sharing. These span from selecting the anonymisation level (non-anonymised, anonymised as Digital Twin, etc.) to setting the price and choosing the licensing terms.

In particular, the sharing configuration consists, amongst others, of the following selection parameters:

a) Anonymisation Level Selection: In particular, the *Individual* can select whether they want to share their data eponymously or anonymously. In the second case, they are provided with two options: The first option is to share an anonymized version of their data, where all personally identifiable information has been altered to

[20] https://www.datavaults.eu/

avoid identification; in other words, create a Digital Twin. The second option is to share their data only as part of anonymised user groups that are constructed by the DataVaults Data Scientist by aggregating multiple users' data, i.e. a Persona.

b) Visibility Level Selection: A second aspect regarding data sharing is related to the discoverability of shared data in search queries. This is closely related to the access policies selection, that will be described in the next point, as the Individual can define not only the attributes of users eligible for purchasing the full data asset, but also the users eligible to find the data assets in their search results and view the selected previews.

c) Access Policies Level Selection: The construction of the appropriate access policies will enable the *Individual* to define who should be able to access each of her/his data assets based on various attributes, such as the type of organisation the *Data Seeker* is related to, the type of personal data (e.g. health, social) and more. An indicative usage example could be that of a user who generally wants to share eponymously data but is reluctant with a specific type of organisations having access to her/his eponymous personal data. At the same time, this user is eager to provide them access to her/his Digital Twin. Such cases are facilitated through attribute-based access control mechanisms that enable the construction of fine-grained policies that will be bound to the data assets and will be resolved whenever an access request for the specific data asset is made. It should be noted that a user could upload data assets to the DataVaults platform without allowing sharing with any *Data Seekers* in any form, by appropriately defining the access policies.

d) Pricing Selection: Afterwards, the Individuals are prompted to create the pricing scheme for each of the data assets they share. As the monetisation of data assets is closely related to the anonymisation level, the availability and demand for this kind of data, and other aspects, DataVaults will provide the Individuals with pricing suggestions that could help them in setting a viable price while maximizing their possible gains. The Individual sets the price tag for the shared data assets.

e) Licensing Selection: The *Individual* can select an applicable licencing scheme that will be applied whenever a *Data Seeker* purchases the specific data asset under the specific pricing. The Individual can define the license parameters, such as the expiry period, permitted purpose of use, sharing terms and more.

The user is offered with the option to automate the data sharing configuration procedure for data that are collected and updated in a recurring manner, through the option to define a sharing schedule for the specific data source.

Furthermore, the various parameters of a data asset sharing configuration can be modified by the *Individual*. This involves changes in the access policies, the sharing schedule and more.

The data requests are presented to the *Individual*, alongside with a message from the Data Seeker as shown in the next figures. The relevant information is displayed in a clear way, to allow users of the target audience to identify the requested asset and understand the implications of the sharing activity. They have the option to accept this request or reject it.

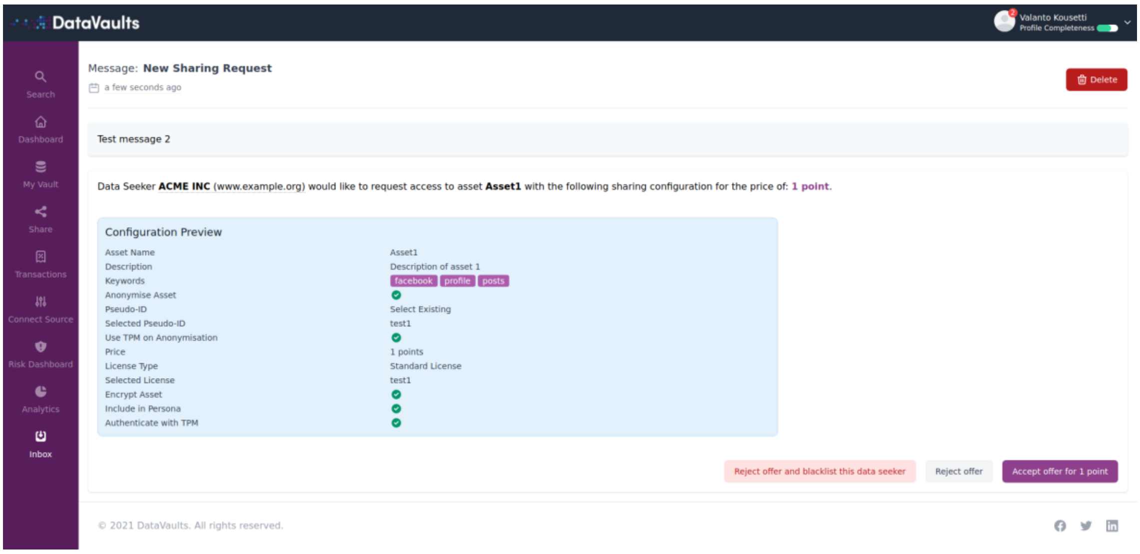


Figure 12 - View an active data sharing request

To mitigate the occurrence of unwanted data requests, DataVaults can be configured to automatically reject all requests from certain *Data Seekers*.

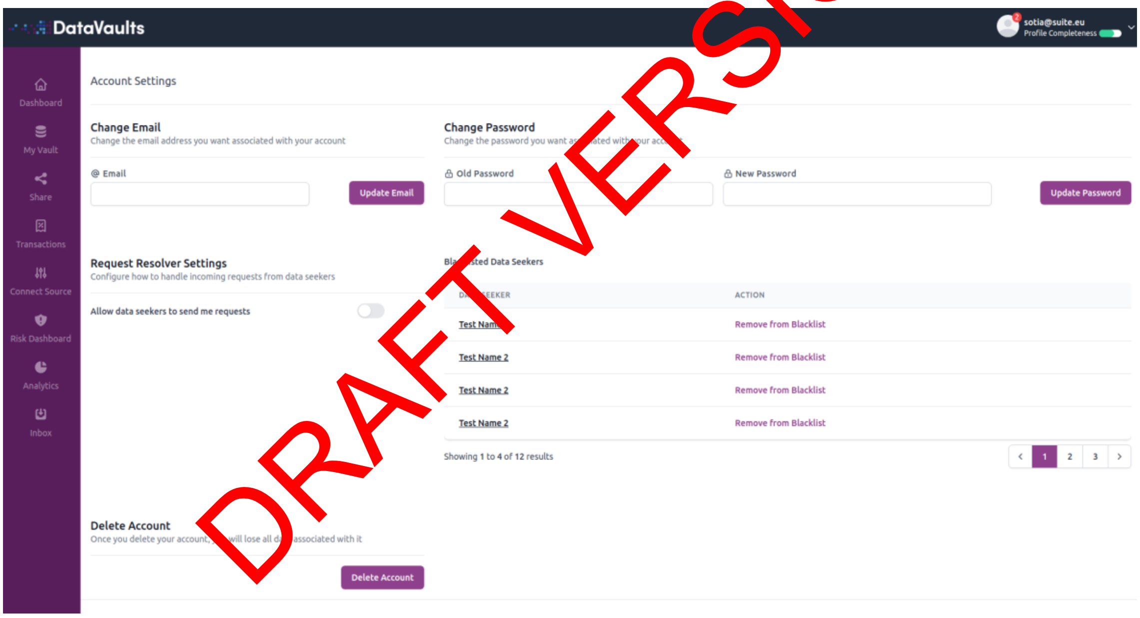


Figure 13 - Blacklisting Data Seekers and Service Toggling

7.2 CaPe

In the scenario of data sharing and related governance we have generally a consumer party and a provider party exchanging data according to an agreement (for example a contract). This agreement authorizes provider party to data provision to consumer party and authorizes the latter to process that data. Also, this agreement can refer to a Data Usage Policy describing processing restriction obligations and duties. In the case that the data spectrum includes personal data, the scenario is more complex, we have not only data provider and consumer, but also the data subject, and usage rules are user centric, and consent based, so they can vary dynamically according to the privacy preferences modification performed by the data subject.

Therefore, the presence of personal data suggests following a user-centric approach of data sharing, which also involves aspects of compliancy, from a legislative point of view, and other aspects about technological and semantic interoperability and the use of standards in order to introduce a use centric approach on data including also the personal spectrum.

CaPe is a technological solution to automate the collection and use of privacy preferences expressed by the individual. It can be integrated into existing processes for the collection of the acceptance of privacy disclaimers and in the more complex case of punctual management of the collection of consents.

CaPe solution[21] is a technological tool that goes beyond the individual aspects of compliance with the GDPR specifications. In particular, the semantic and machine-readable formalization allows on the one hand to "attach" an electronic reference to the consent just given and at the same time support a simpler, more immediate and user-friendly representation of the information, for example through iconography, to accompany the more verbose textual information. At the same time, a machine-readable format allows processing by those involved in the processing within their own information systems together with those already in place for the fulfilment of contracts.

In the specific personal data governance scenario CaPe acts (Figure 14) as an intermediary and as a tool of communication between data subjects and controllers/processors, supporting the generation and management of dynamic consents.

Figure 14 - CaPe solution as intermediary between Data Controller/Processor and Data Subject

CaPe supports fine granularity in consent management and provides open and machine-readable consent format to be processed in usage enforcement. It provides self-service transparency tools for individuals by means they have the possibility at any time to manage their consent, receive notification and exercise data subject rights: objection, right to be forgotten, rectification and copy of data.

And in order to enable the applicability of user centric personal data and consent management in distributed ecosystem of services the availability of Open API, ensuring interoperability (technological and semantic) and compliancy are fundamental. CaPe provides an API ecosystem

[21] https://github.com/OPSILab/Cape

to have consents status so that any changes are automatically available to be used with other systems or with partner organization. CaPe follows MyData Architecture Framework and principles[22] that aim to provide a standard for implementations that satisfies the legal requirements for processing of personal data and provides transparency to individuals about how their data is being used. CaPe also is compliant and supports MIM 4 capabilities Personal Data Management [23] In order to provide a solution for personal data management among public services it is important assure interoperability for an easier information processing. CaPe service description is based on ISA2 Common Public Service Vocabulary Application Profile (CPSV-AP[24]) and Data Privacy Vocabulary (DPV) from W3C[25].

CaPe suite is a web platform based on the microservices paradigm, in which several modules expose a set of APIs through an API Gateway, to be consumed by front-end applications and other external services. The following picture (Figure 15) illustrates the architecture of the CaPe Suite.

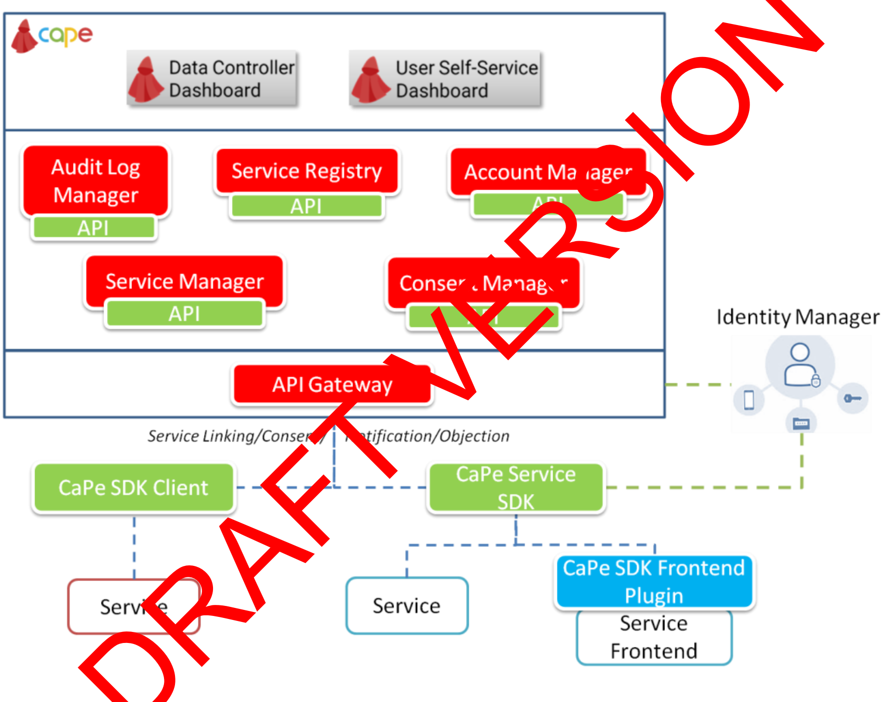


Figure 15- CaPe architecture

Each component of that modular architecture is involved (Figure 16) to support the end-to-end process of consent management, from the formal definition of privacy rules disclaimer to the collection and management of data subject consents and related privacy enforcement.

[22] https://mydata.org/
[23] https://mims.oascities.org/mims/oasc-mim4-trust
[24] https://ec.europa.eu/isa2/solutions/core-public-service-vocabulary-application-profile-cpsv-ap_en
[25] https://w3c.github.io/dpv/dpv/

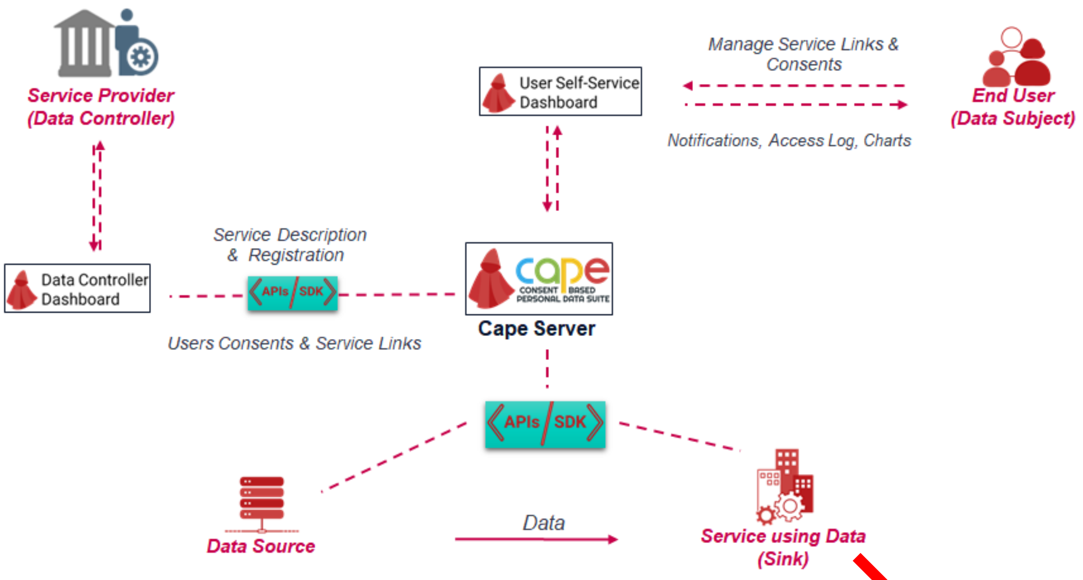


Figure 16- CaPe core components in action

In order to use all the functionalities of CaPe components, a workflow has been defined (Figure 17) and it consists of the following steps:

1. Service description and registration
2. Service Linking
3. Consent Request (for processing within a service or sharing among services)
4. Data Request, Notification and Activity Logs
5. Consent Management & User Data Usage Control

Figure 17- CaPe workflow for a user-centric end-to-end consent management

With CaPe (Figure 18) through the Data Controller Dashboard an organization as service provider can model (Step1) the legal basis for the processing of personal data describing in a standard format (taxonomies, service models…) the relevant information (i.e., purpose, processing, type of data and so on) in line with the related privacy policy. According to the derived model, CaPe automatically generates the consent form that can be shown to the data subject.

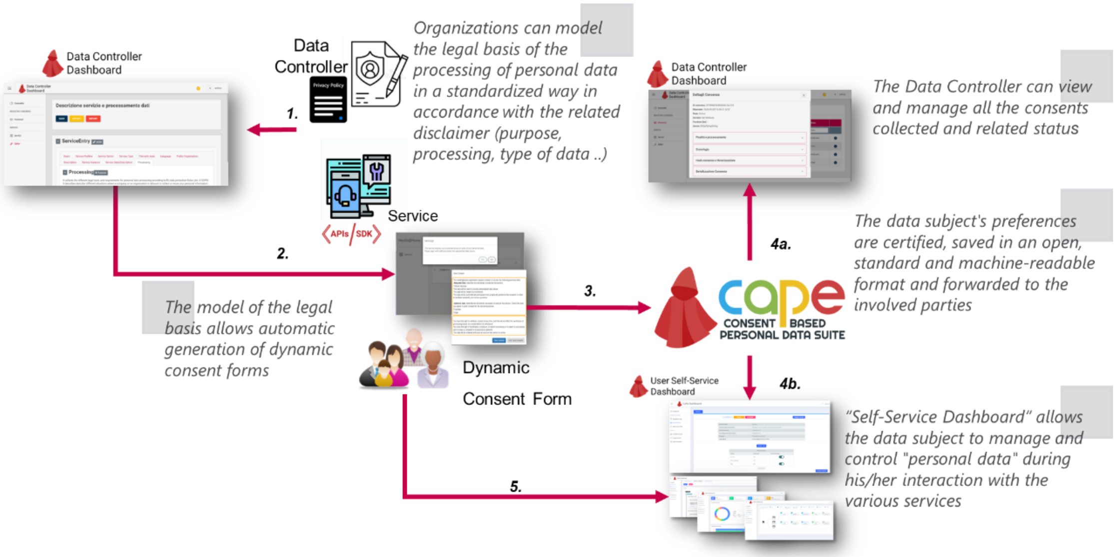


Figure 18 - End to End process of user centric personal data management

The two separated dashboards can let, on one side, the Data Controller to view and manage all the consents collected, on the other, the Data Subject, through the User Self-Service Dashboard, to check which data is used, how and for what purpose and to manage the related consents.