



Project Acronym: **BIMERR**
 Project Full Title: **BIM-based holistic tools for Energy-driven Renovation of existing Residences**
 Grant Agreement: **820621**
 Project Duration: **45 months**

DELIVERABLE D4.7

BIMERR Information Collection & Enrichment Tool 2

Deliverable Status: **Final**
 File Name: **BIMERR_D4.7-v1.00**
 Due Date: **30/06/2021 (M30)**
 Submission Date: **30/06/2021 (M30)**
 Task Leader: **Suite5 (T4.4)**

Dissemination level	
Public	x
Confidential, only for members of the Consortium (including the Commission Services)	



This project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621

The BIMERR project consortium is composed of:

FIT	Fraunhofer Gesellschaft Zur Foerderung Der Angewandten Forschung E.V.	Germany
CERTH	Ethniko Kentro Erevnas Kai Technologikis Anaptyxis	Greece
UPM	Universidad Politecnica De Madrid	Spain
UBITECH	Ubitech Limited	Cyprus
SUITE5	Suite5 Data Intelligence Solutions Limited	Cyprus
HYPERTECH	Hypertech (Chaipertek) Anonymos Viomichaniki Emporiki Etaireia Pliroforikis Kai Neon Technologion	Greece
MERIT	Merit Consulting House Sprl	Belgium
XYLEM	Xylem Science And Technology Management Gmbh	Austria
CONKAT	Anonymos Etaireia Kataskevon Technikon Ergon, Emporikon Viomichanikonkai Nautiliakon Epicheiriseon Kon'kat	Greece
BOC	Boc Asset Management Gmbh	Austria
BX	Budimex Sa	Poland
UOP	University Of Peloponnese	Greece
UEDIN	University of Edinburgh	United Kingdom
NT	Novitech As	Slovakia
FER	Ferrovial Agroman S.A	Spain
UCL	University College London	United Kingdom

Disclaimer

BIMERR project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Commission (EC). EC is not liable for any use that may be made of the information contained therein.

AUTHORS LIST

Leading Author (Editor)				
	Surname	First Name	Beneficiary	Contact email
	Lampathaki	Fenareti	Suite5	fenareti@suite5.eu
Co-authors (in alphabetic order)				
#	Surname	First Name	Beneficiary	Contact email
1	Athanasiadou	Georgia	UOP	gathanas@uop.gr
2	Bountouni	Nefeli	Suite5	nefeli@suite5.eu
3	Chávez-Feria	Serge	UPM	schavez@delicias.dia.fi.upm.es
4	González-Gerpe	Salvador	UPM	salvador.gonzalez.gerpe@upm.es
5	Kousouris	Spiros	Suite5	spiros@suite5.eu
6	Michael	Ioanna	Suite5	ioanna@suite5.eu
7	Pertselakis	Minas	Suite5	minas@suite5.eu
8	Poveda-Villalón	María	UPM	mpoveda@fi.upm.es
9	Tsoulos	George	UOP	gtsoulos@uop.gr
10	Vafeiadis	Giorgos	UBITECH	gvafeiadis@ubitech.eu
11	Vergeti	Danai	UBITECH	vergetid@ubitech.eu

REVIEWERS LIST

List of Reviewers (in alphabetic order)				
#	Surname	First Name	Beneficiary	Contact email
1	Poveda-Villalón	María	UPM	mpoveda@fi.upm.es
2	Katsigarakis	Kyriakos	UCL	k.katsigarakis@ucl.ac.uk

REVISION CONTROL

Version	Author	Date	Status
0.10	Suite5	19/04/2021	Draft ToC
0.20	Suite5	28/05/2021	Draft Sections 1,2,3,4,5,6,8
0.30	UPM	09/06/2021	Draft Sections 7
0.40	Suite5	09/06/2021	Draft Sections 1, 2,3,4,5,6,7,8
0.50	Suite5	11/06/2021	Draft version to be circulated for Internal Quality Check

Version	Author	Date	Status
0.60	Suite5	29/06/2021	Updated version addressing comments received during the Internal Quality Check
1.00	Suite5	30/06/2021	Final version for submission to the EC

TABLE OF CONTENTS

<i>List of Figures.....</i>	9
<i>List of Tables.....</i>	12
EXECUTIVE SUMMARY	13
1. Introduction.....	15
1.1 Scope and Objectives of the Deliverable.....	15
1.2 Relation to other tasks/deliverables.....	16
1.3 Structure of the document.....	18
2. BIMERR INFORMATION COLLECTION & ENRICHMENT COMPONENT	19
2.1 Overview.....	19
2.2 Architecture	20
3. DATA INGESTER & FETCHER.....	22
3.1 Overview.....	22
3.2 Technology Stack and Implementation Tools.....	24
3.3 API Documentation	25
3.4 Changes introduced in final release	26
3.5 Assumptions and Restrictions	26
3.6 Installation Instructions	26
3.7 Licensing	27
4. DATA HANDLER.....	28

4.1	Overview.....	28
4.2	Technology Stack and Implementation Tools.....	29
4.3	API Documentation	31
4.4	Changes introduced in final release	31
4.5	Assumptions and Restrictions	32
4.6	Installation Instructions	32
4.7	Licensing	32
5.	<i>DATA STORAGE & INDEXING</i>	33
5.1	Overview.....	33
5.2	Technology Stack and Implementation Tools.....	34
5.3	API Documentation	35
5.4	Changes introduced in final release	35
5.5	Assumptions and Restrictions	35
5.6	Installation Instructions	35
5.7	Licensing	36
6.	<i>MASTER CONTROLLER.....</i>	37
6.1	Overview.....	37
6.2	Technology Stack and Implementation Tools.....	38
6.3	API Documentation	40
6.4	Changes introduced in final release	40
6.5	Assumptions and Restrictions	40

6.6	Installation Instructions	41
6.7	Licensing	41
7.	KNOWLEDGE GRAPH GENERATOR.....	42
7.1	Overview.....	42
7.2	Technology Stack and Implementation Tools.....	43
7.3	API Documentation	45
7.4	Changes introduced in final release	46
7.5	Assumptions and Restrictions	46
7.6	Installation Instructions	47
7.7	Licensing	47
8.	End-To-End Usage Walkthrough to the BIMERR Information Collection & Enrichment Component.....	48
8.1	Initiate a Data Collection Job.....	48
8.2	Configure a Data Collection Job for Files Upload	51
8.3	Configure a Data Collection Job through Data Providers' available API	54
8.4	Configure a Data Collection Job through the BIF APIs.....	62
8.5	Provision of an Asset's metadata	67
8.6	Update of the Data Collection Job	69
9.	CONCLUSIONS.....	72
	ANNEX I: BIBLIOGRAPHY.....	73

LIST OF FIGURES

Figure 2- 1: Architecture of the BIMERR Information Collection & Enrichment component	20
Figure 3-1: Architecture of the Data Ingester & Fetcher under the BIMERR Information Collection and Enrichment Component.....	24
Figure 4- 1: Architecture of the Data Handler under the BIMERR Information Collection and Enrichment Component	30
Figure 5-1: Architecture of the Data Storage & Indexing under the BIMERR Information Collection and Enrichment Component.....	34
Figure 6-1: Architecture of the Master Controller under the BIMERR Information Collection and Enrichment Component.....	39
Figure 7- 1: Architecture of the Knowledge Graph Generator	43
Figure 8-1: View of all user's data collection jobs	48
Figure 8-2: Creation of a new Data Collection Job.....	49
Figure 8-3: Data Collection configuration – Data Loading method selection	49
Figure 8-4 Data Collection configuration - Notification for final confirmation of modality selection.....	50
Figure 8-5: Configure data collection for Files	51
Figure 8-6: Configure data collection for Files – Review of uploaded data sample (if applicable)	52
Figure 8-7: Configure data collection for Files - Notification for successful saving of the data upload	53

Figure 8-8: Configure data collection for Files - Notification for completed configuration of the Harvester	54
Figure 8-9: Data collection through external APIs - Harvester configuration	55
Figure 8-10: Data collection through external APIs - No authentication required	56
Figure 8-11: Data collection through external APIs - Bearer Authentication options	56
Figure 8-12: Data collection through external APIs – Custom Authentication options.....	57
Figure 8-13: Data collection through external APIs - Authentication, Method, Pagination and Query Parameters	58
Figure 8-14: Data collection through external APIs - Extra Headers & Retrieval Settings	59
Figure 8-15: Data collection through external APIs – Processing & Error Handling Strategy	60
Figure 8-16: Data collection through external APIs - Response handling, storage of additional parameters and API response selection	60
Figure 8-17: Data collection through external APIs – View and save sample of selected API response part.....	61
Figure 8-18: Data collection through external APIs - Notification for completing Harvester configuration	62
Figure 8-19: Data collection through BIF's API - Harvester configuration (text only)	63
Figure 8-20: Data collection through BIF's API – Instructions for data loading (text) and overview of data sample	64
Figure 8-21: Data collection through BIF's API - Harvester configuration (Text and Binary)	65

Figure 8-22: Data collection through BIF's API - Notification for successful saving of the data upload	66
Figure 8-23: Data collection through BIF's API - Notification for completing Harvester configuration	66
Figure 8-24: View of user's uploaded Asset's	67
Figure 8-25: Overview of data asset	67
Figure 8-26: Edit Asset Details.....	68
Figure 8-27: Updating the data collection configuration for external APIs – Part 1	69
Figure 8-28: Updating the data collection configuration for external APIs – Part 2.....	70
Figure 8-29: Updating the data collection configuration for external APIs – Part 3.....	71
Figure 8-30: Data collection through external APIs - Notification for completing Harvester configuration	71

LIST OF TABLES

Table 3-1: Technologies and libraries used in the Data Ingester & Fetcher, along with their licenses	25
Table 4- 1: Technologies and libraries used in the Data Handler, along with their licenses	30
Table 5-1: Technologies and libraries used in the Data Storage & Indexing, along with their licenses	35
Table 6- 1: Technologies and libraries used in the Master Controller, along with their licenses	39
Table 7- 1: Technologies and libraries used in the Knowledge Graph Generator, along with their licenses	44

EXECUTIVE SUMMARY

The BIMERR Deliverable D4.7 “BIMERR Information Collection & Enrichment Tool 2” aims at documenting the final release of the BIMERR Information Collection & Enrichment Component (BICE) and concluding the development activities performed in the context of T4.4 “Building Information Collection and Enrichment Tools Creation”. Overall, the BIMERR Information Collection & Enrichment Component lies at the core of the BIMERR Interoperability Framework (BIF) and is responsible for the timely and effective collection of building-related data from multiple sources, ranging from the BIMERR applications to legacy systems, while effectively addressing a number of semantic and syntactic interoperability challenges from a data management perspective.

In alignment with the BIMERR architecture, the BIMERR Information Collection & Enrichment Component has delivered five subcomponents, namely the Data Ingester & Fetcher (that is responsible for the retrieval of building-related data from various sources), the Data Handler (that is responsible for building-related data pre-processing, according to the configuration defined in the BIMERR Building Semantic Modelling tools in D4.5, and for metadata management), the Data Storage & Indexing (undertaking the data storage and indexing of the building data in the BIF), the Master Controller (orchestrating the execution of the building data collection jobs) and the Knowledge Graph Generator (generating the semantic representation of existing, open data). Such subcomponents build on over 20 state-of-the art technologies to deliver the planned functionalities for the intended users, i.e., BIMERR applications owners/developers that act as building data providers to the BIF.

The present documentation of the BIMERR Information Collection & Enrichment Component along with its subcomponents is oriented towards the functionalities they broadly deliver, the technology stacks they build upon, the APIs they expose, the installation instructions and end-to-end usage walkthroughs they offer to their users. In its final release, the BIMERR Information Collection & Enrichment Component has fully implemented all the envisaged functionalities, but depends on certain assumptions and imposes a set of restrictions.

The final iteration of the BIMERR Information Collection & Enrichment Component is built on the outcomes of D4.6 “BIMERR Information Collection & Enrichment Tool 1” and the first release of the component and introduces a new set of extensions and functionalities, based on the feedback received from the BIF components during the BIF integration activities undertaken in the context of WP4, and the BIMERR applications in collaboration with WP5 “As-is Building Information Extraction & Model Population Tools”, WP6 “Process Management Tools & End-User Apps for On-site Stakeholders”, WP7 “Renovation Decision Support System” and WP8 “ICT System Integration, Testing & Pre-Validation”.

1. INTRODUCTION

1.1 SCOPE AND OBJECTIVES OF THE DELIVERABLE

The present deliverable D4.7, entitled “BIMERR Information Collection & Enrichment Tool 2” reports the activities undertaken within the context of Task T4.4 – “Building Information Collection and Enrichment Tools Creation” of WP4 “BIMERR Interoperability Framework”, towards the delivery of the final version of the Building Information Collection & Enrichment Component (BICE) of the BIMERR Interoperability Framework (BIF). The BIMERR Building Information Collection & Enrichment Component plays an instrumental role from the perspective of the building-related data providers as it is practically responsible for: collecting building-related data in multiple modalities, processing them according to the provisions of the relevant BIMERR data models, and finally storing and indexing the transformed data along with their accompanying information so as to be eventually available to all authorized BIMERR applications.

The main objective of D4.7 is to provide a comprehensive overview and the documentation report of the final stable release of the BIMERR Information Collection & Enrichment Component. Since this deliverable is of type “Demonstrator”, it provides the updated documentation of the actual software that has been developed (for the final iteration) and delivered in accordance with the BIMERR requirements and architecture.

Each of the five subcomponents that constitute the BIMERR Information Collection & Enrichment component namely, the Data Ingestor & Fetcher, the Data Handler, the Data Storage & Indexing, the Master Controller and the Knowledge Graph Generator, is described in detail by:

- Elaborating on the functionalities of each subcomponent
- Defining the technology stack upon which each subcomponent is based.
- Documenting the Application Programming Interfaces (APIs), i.e., endpoints which will enable the required communications and information exchanges between the different subcomponents and / or within the BIF.
- Explaining any assumptions and restrictions considered in the final release of each subcomponent.
- Providing installation instructions, in order to deploy the subcomponents.

- Offering a usage walkthrough through a set of step-by-step screenshots to explain in detail each subcomponent's intended use.
- Identifying the accompanying licensing of each subcomponent.
- Presenting the changes introduced in the final release of the subcomponents.

In accordance with the BIMERR DoA [1], the final release of the Building Information Collection & Enrichment component delivered in M30 of the project's implementation, is built on the outcomes of deliverable D4.6 and contains all the planned functionalities, as well as refinements and enhancements based on the BIMERR partners' feedback during the BIF integration activities of WP4, as well as the preliminary integration activities with applications of WP5 "As-is Building Information Extraction & Model Population Tools", WP6 "Process Management Tools & End-User Apps for On-site Stakeholders" and WP7 "Renovation Decision Support System", in the context of the WP8 "ICT System Integration, Testing & Pre-Validation" activities.

Finally, considering that the technology and the architecture of the Building Information & Collection component and its associated subcomponents, have not undergone deviations from their initial release, specific parts of their documentation have essentially the same state, as presented in D4.6.

1.2 RELATION TO OTHER TASKS/DELIVERABLES

The BIMERR Deliverable D4.7 documents the activities performed in Task T4.4 "Building Information Collection and Enrichment Tools Creation" and its main scope is to deliver the final version of the Building Information Collection and Enrichment Component. Towards this direction, for the design and implementation of the subcomponents described in this deliverable, the current document receives input from the following BIMERR deliverables:

- D3.1 "Stakeholder requirements for the BIMERR system"[2], where the key BIMERR stakeholders and their requirements are documented, along with a thorough

description of the business scenarios, use cases and system requirements tailored to the project's goals, setting the foundations for the BIMERR framework.

- D3.6 “BIMERR system architecture 2nd version” [3], where the second version of the BIMERR architecture is provided, outlining the interaction of the BIF components, along with a detailed technical description of the Building Information Collection and Enrichment Component and its subcomponents.
- D4.3 “BIMERR Ontology & Data Model 2” [5], where the final BIMERR ontology and data model structure is developed, in order to address the various semantic interoperability challenges for BIM-related data in an efficient manner.
- D4.5 “Building Semantic Modelling Tool 2” [6] that is utilized for the definition of the semantic mapping of the data collected in the Building Information Collection and Enrichment component.
- D4.6 “BIMERR Information Collection & Enrichment Tool 1” [7], where the initial release of the BIMERR Information Collection & Enrichment Tool is documented and upon which the present deliverable (D4.7) is built on, while also considering the feedback and lessons learnt from the BIF integration activities with the BIMERR tools.
- D4.8 “Integrated BIMERR Interoperability Framework 1” [8], providing a comprehensive documentation of the integrated BIF, and presenting the key technical aspects of the Building Information Secure Provisioning component and Building Information Query Builder and their interaction with the other components of BIF.

The outcome of the activities performed in D4.7 will be also used as input in the following tasks and work packages:

- T4.5 “Building Information Secure Provisioning Tool Creation”, as D4.7 is foreseen to provide input to the design and development of the final version of the Building Information Secure Provisioning component (documented in D4.9), for the efficient and secure export of data to the appropriate BIMERR applications and stakeholders in accordance with the applicable data access policies that their provider has defined.
- T4.6 “Building Information Query Builder Creation”, where D4.7, through its Data Storage & Indexing technology stack, will provide the basis upon which the final version of the Building Information Query Builder (documented in D4.9) will be built in order to search for building-related data (based on their metadata and/or actual data).

In addition, D4.7 will offer a better understanding of the data provisioning alternatives of building-related data to the BIF, to all BIMERR applications that are delivered in WP5 “As-is Building Information Extraction & Model Population Tools”, WP6 “Process Management Tools & End-User Apps for On-site Stakeholders” and WP7 “Renovation Decision Support System”. Finally, D4.7 and the Building Information Collection & Enrichment Component is naturally part of the system level software integration and pre-validation activities performed in WP8 “ICT System Integration, Testing & Pre-Validation” and thereafter in the validation and evaluation activities of WP9 “Validation & Evaluation Activities”.

1.3 STRUCTURE OF THE DOCUMENT

In order to address all the aspects relevant to the scope of D4.7, the present deliverable has been structured as follows:

- Section 1 introduces the work performed and the scope of this deliverable along with its relevance to other BIMERR tasks and the deliverable’s structure.
- Section 2 provides an overview of the final release of the BIMERR Building Information Collection & Enrichment Component, and its architecture.
- Sections 3–7 provide a comprehensive documentation of the different subcomponents forming the Building Information Collection and Enrichment component, i.e., the Data Ingester & Fetcher, the Data Handler, the Data Storage & Indexing, the Master Controller and the Knowledge Graph Generator, respectively.
- Section 8 offers an end-to-end usage walkthrough through step-by-step instructions accompanied by appropriate screenshots to explain in detail each subcomponent’s intended use.
- In Section 9, conclusions of this document are provided.

2. BIMERR INFORMATION COLLECTION & ENRICHMENT COMPONENT

2.1 OVERVIEW

In general, the BIMERR Information Collection & Enrichment component is responsible for preparing and executing the collection, manipulation and storage of building-related data in the BIMERR Interoperability Framework, in order to be eventually available to all authorized users and applications in BIMERR.

The BIMERR Information Collection & Enrichment component is composed of the following five main subcomponents (as depicted in Figure 2-1):

- The **Data Ingestor & Fetcher**, which provides all the required functionalities to enable the timely and successful retrieval of building-related data from various sources. These data can be directly retrieved from the users a) as uploaded file(s), b) via APIs exposed by the BIMERR applications for their application data, and c) via internal APIs exposed by BIF.
- The **Master Controller**, which aims to support the orchestration and scheduling of the data ingestion, manipulation and storage loading execution.
- The **Data Handler**, which oversees a set of mapping and transformation functions over the ingested data from their native data model and format to the target BIMERR data model based on the configuration defined in the Model Mapper (under the Building Semantic Modelling Component documented in D4.5).
- The **Data Storage & Indexing**, which is responsible for persisting the building-related data along with their accompanying information in the BIF, offering data and metadata storage and indexing capabilities.
- The **Knowledge Graph Generator**, which aims at generating knowledge graphs or the semantic representation of existing data derived either from external data sources or, if required, from the Data Storage & Indexing, through data translation allowing semantic querying over the translated data.

The final version of the BIMERR Information Collection & Enrichment component is deployed at: <https://bimerr.s5labs.eu/>

2.2 ARCHITECTURE

In alignment with the BIMERR deliverable D3.6 [3], Figure 2-1 illustrates the BIMERR Information Collection & Enrichment component's architecture including its main subcomponents, information flows and interactions among them.

As already mentioned, such a component is composed of five subcomponents, namely the Data Handler, the Data Storage & Indexing, the Data Ingester & Fetcher, the Master Controller and the Knowledge Graph Generator; each one designed with a distinct role, a distinct scope and a distinct set of core functionalities.

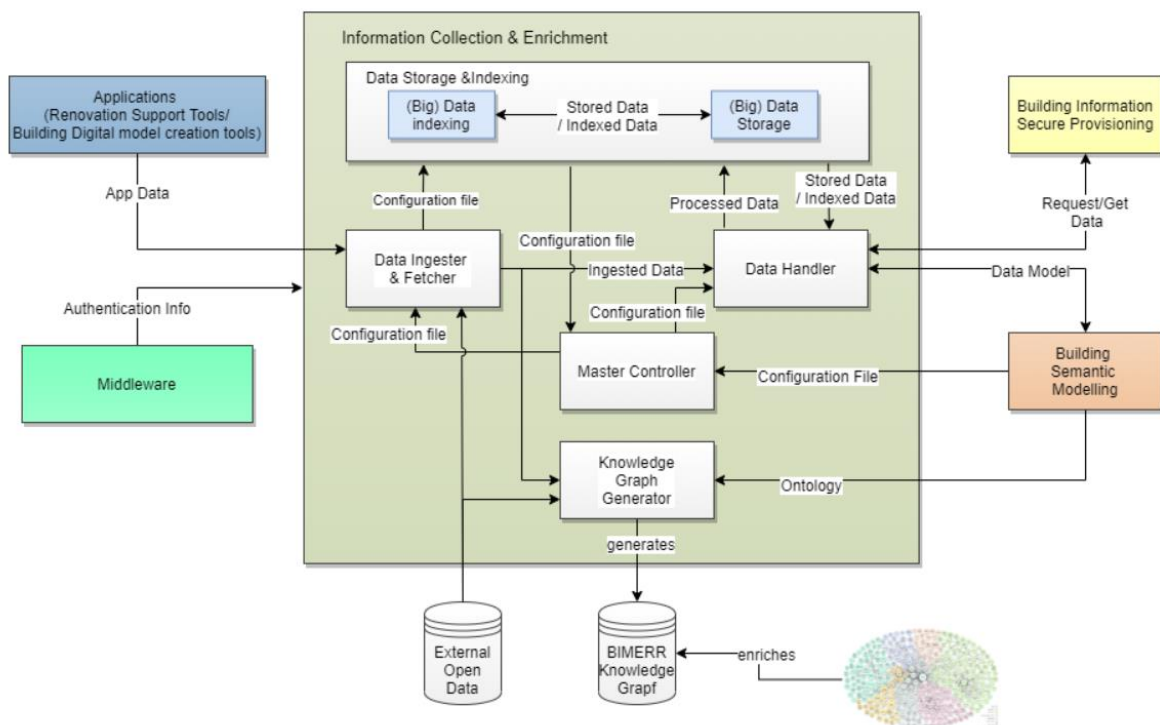


Figure 2- 1: Architecture of the BIMERR Information Collection & Enrichment component

The BIMERR Information Collection & Enrichment component allows for the definition of data collection jobs by the BIMERR application owners/developers. At preparation time, the configuration of the data collection job needs to be successfully created and stored. At execution time, the *Data Ingester & Fetcher* is triggered by the *Master Controller* that provides the configuration file of the data collection job that is to be executed in order to successfully retrieve the building data that are uploaded either directly by users or

indirectly through APIs. APIs may be provided by the BIMERR applications for application data or the BIF APIs can be utilised. Additionally, the Data Ingestor & Fetcher allows for import of open data that may come from various external open data sources by providing the necessary mechanisms to connect and retrieve the corresponding datasets based on the provided configuration.

The Master Controller then triggers the *Data Handler* passing through the corresponding configuration file in order to execute all the necessary mapping instructions and transformation rules (e.g., for datetime format transformations, timezone transformations and measurement unit transformations) and ensure that the underlying data are compliant with the respective BIMERR data model. The Data Handler is also responsible for preparing the requested data for export in collaboration with the BIF Building Information Secure Provisioning component.

Once the data handling processes are completed in the Data Handler, the *Data Storage & Indexing* subcomponent is initiated by the Master Controller to appropriately store the data payload along with its accompanying information in multiple big-data enabled storage modalities, while offering an effective indexing mechanism that facilitates the near real-time indexing and advanced querying capabilities over the indexed data.

Furthermore, open datasets available in the Linked Open Data cloud, as well as open data from the Data Handler (if they do not have any access policies restrictions), are accessed by the *Knowledge Graph Generator*, in order to generate complementary knowledge over the building-related data.

3. DATA INGESTER & FETCHER

3.1 OVERVIEW

The Data Ingestor & Fetcher subcomponent provides the necessary functionalities to enable the timely and successful retrieval of data from various sources in the BIMERR Interoperability Framework. The retrieval of data happens through the execution of a “data collection” process that is fully configurable by the user (i.e., BIMERR application developer and any other data provider to the BIF) through an intuitive user interface. Building-related data can be retrieved in multiple modalities: either through a manual operation to directly upload files, or through an automated operation to retrieve data from APIs that have been exposed a) by BIF or b) by the BIMERR applications. During the data collection job configuration, the user can define not only the ingestion parameters (such as the retrieval method, schedule etc.), but also which part of the ingested data should be stored in the BIF.

More specifically, the functionalities offered by the Data Ingestor & Fetcher are the following:

- **Step-by-step definition of the data collection configuration in an intuitive manner:** The Data Ingestor & Fetcher allows the user to define the parameters of the data collection process, according to their preferences and needs. Indicatively, the user is provided with several configuration options (e.g. regarding the retrieval modes, the collection schedule, the authentication parameters) in order to appropriately set up the data collection job.
- **Ingestion configuration creation and update under conditions on a case-by-case basis:** The data collection configuration that includes the user settings and preferences is automatically created and stored in a configuration file for the specific data collection job process. The user is able to persist the ingestion configuration progress made at any moment and revisit it in the future in order to finalize the configuration. For an already finalized configuration, updates by the user are available under certain conditions, e.g., the periodic data retrieval schedule can be revised if the data collection job is not running at the specific moment. In addition, only certain configuration

parameters are allowed to change, in order to ensure that no inconsistencies occur with data that have already been retrieved with this data collection job.

- **Retrieval of data as single and multiple files:** The Data Ingester & Fetcher supports the direct uploading of single and multiple files by the users, as well as of their samples, in multiple formats ranging from tabular formats (e.g., csv) to non-tabular formats (e.g. json, xml) and other files (e.g. IFC, IDF, images).
- **Retrieval of data through APIs:** The Data Ingester & Fetcher shall retrieve data that are exposed by the BIF's own APIs or by the BIMERR applications' APIs. In the latter case, it allows the user to provide their full definition, such as the method, path, query parameters and body of the API calls, if applicable. Depending on the configuration provided, the Data Ingester & Fetcher tests the API in order to retrieve sample data.
- **Handling advanced authentication aspects, while protecting sensitive data:** When an API-based data collection job is initiated (based on an application's APIs), the authentication aspects need to be defined for establishing a reliable API connection between the BIF and the endpoint defined by the user. As authentication is typically required in many different ways, the user shall define during the ingestion configuration in the Data Fetcher & Ingester, the type of authentication (ranging from bearer to custom authentication) and provide the necessary parameters (e.g. tokens, username and password for custom login). The Data Ingester & Fetcher performs a test login to verify the provided information and then secures and stores the sensitive parameters for the future API calls in a reliable manner.
- **User-dependent selection of data to be stored:** For data retrieved through applications' APIs, the user shall select, at configuration time, which part of the sample data that has been retrieved should be stored in BIF. The Data Ingester & Fetcher will only forward for processing the selected part, and the rest of the data that were not selected by the user are discarded from the final "dataset".
- **Generation of data collection status messages:** The Data Ingester & Fetcher shall communicate the status of its data collection jobs to the Master Controller through several feedback messages (e.g., if no API results were retrieved due to server or user errors along with their respective code).

3.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The Data Ingester & Fetcher builds on state-of-the-art technologies across 3 layers:

- The Presentation Layer, containing the Ingester & Fetcher User Interface that is developed in VueJS¹ and TailwindCSS²
- The Business Logic Layer, containing the different packages of the Data Ingester & Fetcher Backend that are based in the Flask micro web framework³
- The Data Access Layer that essentially refers to the BIF Storage and Indexing that has been set up and utilizes PostgreSQL⁴ (as the relational database for the data collection job storage), MinIO⁵ (as the data lake for the intermediate files management) and Vault⁶ (as the secure database for sensitive and secret parameters), for the Data Ingester & Fetcher needs.

Such layers along with the different technologies are depicted in Figure 3-1.

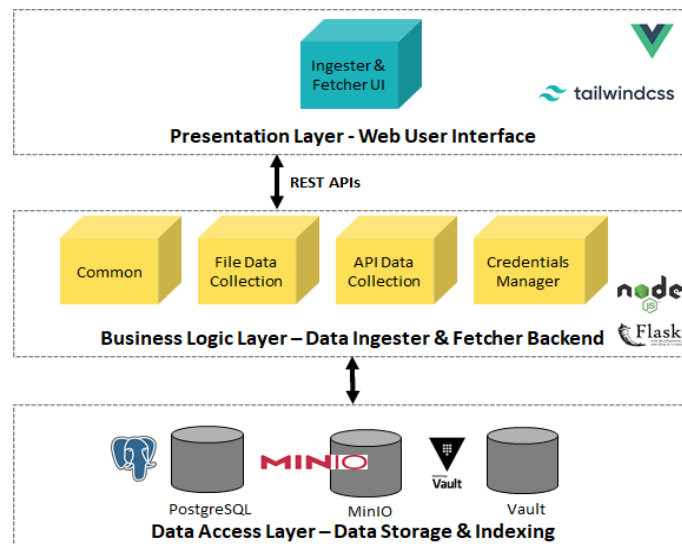


Figure 3-1: Architecture of the Data Ingester & Fetcher under the BIMERR Information Collection and Enrichment Component

¹ <https://vuejs.org/>

² <https://tailwindcss.com/>

³ <https://flask.palletsprojects.com/en/1.1.x/>

⁴ <https://www.postgresql.org/>

⁵ <https://min.io/>

⁶ <https://www.vaultproject.io/>

The Data Ingester & Fetcher is written in Python 3.8.2⁷ and utilizes the following open-source technologies defined in Table 3-1.

Table 3-1: Technologies and libraries used in the Data Ingester & Fetcher, along with their licenses

Name of the Library	Version	License
Flask	1.1.1	BSD 3-Clause
Flask RESTful extension	0.3.8	BSD 3-Clause
Flask CORS support	3.0.8	MIT
PostgreSQL	12.2	PostgreSQL License (similar to BSD/MIT)
MinIO	-	Apache License 2.0
Vault	-	Mozilla Public License 2.0
Vue.js	2.6.11	MIT
TailwindCSS	-	MIT
Pandas	1.0.3	BSD 3-Clause
Pika	1.1.0	BSD 3-Clause

3.3 API DOCUMENTATION

The services of the Data Ingester & Fetcher communicate with the other components and services under the BIMERR Information Collection and Enrichment Component through the Master Controller's messaging functionality and there are not any APIs needed or defined in the initial release. The communication between the front-end and back-end of the Data Ingester & Fetcher is typically through internal APIs, but as they only support inter-subcomponent integration, they are not documented at this point.

In order to allow external applications to push data, the BIF (through the Data Ingester & Fetcher) exposes auto-generated API endpoints that are customized per data collection job. This means that once a sample of data that an application intends to send and the type of data to be sent are provided by the user, the BIF generates an endpoint such as

⁷ <https://www.python.org/>

<https://bimerr.s5labs.eu/api/upload/968c4016-ba27-4047-9bbd6-e721cc272f44> and provides appropriate instructions: (a) for authentication, and (b) for multi-part requests in case a user intends to send text data and other data (files or binary data). In this context, since the BIF API endpoints are designed to receive any combination of data depending on the user preferences, it is not possible to document such APIs in the typical way (e.g. through Swagger).

3.4 CHANGES INTRODUCED IN FINAL RELEASE

In respect to the 1st release of the Data Ingestor & Fetcher component, a number of improvements and enhancements has been introduced in this final release. In particular, the following features and extensions have been developed upon discussions with the partners and the feedback received on the first release:

- Support for sending data via the BIF APIs. Provision for sending both binary and text data via the BIF's own APIs.
- Support for API pagination anticipating the different pagination strategies that are currently found in the latest web development practices and allowing the user to fully configure if/how pagination should be handled for their API.
- More advanced checks and restrictions on the selection of the API response part.
- Bug fixing and user experience improvements.

3.5 ASSUMPTIONS AND RESTRICTIONS

In the present, final release of the Data Ingestor & Fetcher, certain assumptions (that in certain cases, represent restrictions for the Data Ingestor & Fetcher) were taken:

- The data the users send through the APIs must be always the same (in terms of field names, field data types and structure) as in the sample provided.
- Only one binary file can be sent with each API call through the BIF APIs.

3.6 INSTALLATION INSTRUCTIONS

The Data Ingestor & Fetcher User Interface is served as a web application and does not require the installation of any component by the user. Detailed instructions for the Data

Ingestor & Fetcher deployment are provided in the related private code repository and all subcomponents are already packaged as Docker containers to speed up the process.

3.7 LICENSING

The BIMERR Data Ingestor & Fetcher is a closed source component.

4. DATA HANDLER

4.1 OVERVIEW

The Data Handler is responsible for performing the underlying background data processing operations to ensure the building-related data that are ingested in the BIF are appropriately harmonized, mapped, and aligned with the respective BIMERR data models as defined at configuration time in the Model Mapper of the BIMERR Building Semantic Modelling Component (described in D4.5). Since data processing typically entails data validation and transformation, the Data Handler acquires (from the Master Controller that is described in Section 6) the configuration file for each data collection job in order to proceed with the execution of the respective operation (ranging from data mapping and data type transformation to datetime format transformation and timezone transformation). The processed data are then forwarded to the Data Storage & Indexing subcomponent (described in Section 5) in order to be permanently stored. A user interface is also provided in order for the users (i.e., data providers) to provide metadata for the uploaded datasets, to ease their eventual identification and retrieval. The Data Handler is also responsible for preparing requested data for export in collaboration with the Building Information Secure Provisioning component under the BIF.

In particular, the Data Handler offers the following functionalities:

- **Mapping of the ingested data to a respective BIMERR data model:** According to the provisions of the configuration file that has been created in the Model Mapper based on the user-confirmed mapping (according to the documentation provided in D4.5), the alignment of the ingested data to the applicable BIMERR data model is performed.
- **Execution of data transformations rules based on user mapping configurations:** As described in D4.5 [6], during the mapping configuration in the Model Mapper, the user has specified the mapping details, including for example the measurement units that apply to their numerical data, the datetime format that is used and any other custom transformation. The Data Handler derives automatically from these settings the respective transformation rules (i.e., unit transformations, data type changes, datetime format changes, timezone conversions) that need to be applied on the data in order to be compliant with the BIMERR Interoperability Framework.

- **Definition of data asset metadata:** The Data Handler provides a user-friendly interface through which the user should provide metadata information (such as free-text tags, description, temporal and spatial coverage, licensing information, among others) for the data asset that will be created from their ingested and processed data (in the context of a data collection job). Such metadata will facilitate internal processes in the BIF, but also make the respective data assets more easily searchable (through the Query Builder interfaces in the BIF, described in D4.8 and D4.9).
- **Preparation of the processed data payload for storage and indexing in the Data Storage & Indexing component:** The Data Handler wraps up the processed data together with the metadata defined by the user and forwards this data asset to the Data Storage & Indexing subcomponent in order to be persisted in the BIF.
- **Preparation of export of data in collaboration with the Secure Provisioning component:** Whenever an authorised user or application requests data that are stored in the BIF, the Data Handler component is responsible for bringing the requested data “slice” in the appropriate format and forwarding the exportable payload to the Building Information Secure Provisioning component.
- **Generation of data handling status messages:** The Data Handler is designed to efficiently communicate the status of its data management processes to the Master Controller and any errors encountered through a number of feedback messages (e.g. the number of “rows” in the data that were dropped due to inconsistent data types).

4.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The Data Handler builds on state-of-the art technologies across three layers:

- The Presentation Layer, containing the Data Handler User Interface that is developed in VueJS and TailwindCSS
- The Business Logic Layer, containing the different packages of the Data Handler Backend that are based in the Flask micro web framework³

- The Data Access Layer that essentially refers to the BIF Storage and Indexing that has been set up and utilizes PostgreSQL⁴, ElasticSearch⁸, MinIO⁵ and PostgreSQL⁴, for the Data Handler needs.

Such layers along with the different technologies are depicted in Figure 4-1.

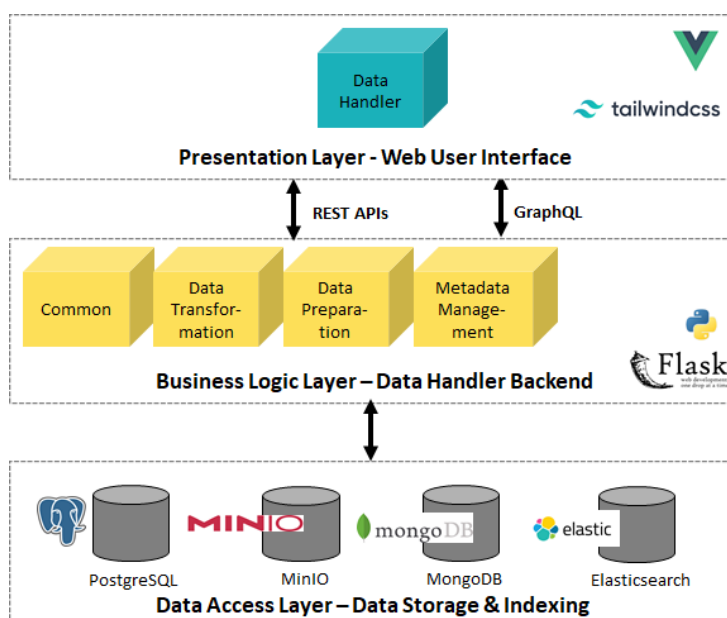


Figure 4- 1: Architecture of the Data Handler under the BIMERR Information Collection and Enrichment Component

The Data Handler is also written in Python 3.8.2⁷ and utilizes the open-source technologies defined in Table 4-1.

Table 4- 1: Technologies and libraries used in the Data Handler, along with their licenses

Name of the Library	Version	License
Flask	1.1.1	BSD 3-Clause
Flask RESTful extension	0.3.8	BSD 3-Clause
Flask CORS support	3.0.8	MIT
PostgreSQL	12.2	PostgreSQL License (similar to BSD/MIT)
MinIO	-	Apache License 2.0
PyMongo	3.10.1	Apache License 2.0

⁸ <https://www.elastic.co/>

Name of the Library	Version	License
Elasticsearch	7.6.0	Elastic License
Vue.js	2.6.11	MIT
TailwindCSS	-	MIT
Pandas	1.0.3	BSD 3-Clause
Pika	1.1.0	BSD 3-Clause
NumPy	1.18.1	BSD

4.3 API DOCUMENTATION

The services of the Data Handler communicate with the other components and services under the BIMERR Information Collection and Enrichment Component through the Master Controller's messaging functionality and there are not any APIs needed. The communication between the front-end and back-end of the Data Handler is typically through internal APIs, but they serve inter-subcomponent integration purposes and are thus not documented at this point.

4.4 CHANGES INTRODUCED IN FINAL RELEASE

In respect to the 1st release of the Data Handler component, a number of improvements and enhancements has been introduced in this final release. In particular, the following features and extensions have been developed upon discussions with the partners and the feedback received on the first release:

- Support for additional transformations depending on the needs and provisions of the BIMERR data models.
- Provision of additional metadata (e.g., status, category and version) for alignment with ISO 19650 -1:2018 "Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 1: Concepts and principles" [9]

4.5 ASSUMPTIONS AND RESTRICTIONS

In the final release of the Data Handler, handling all data under a unique format (with the exception of the other type of files that ranges from images to IFC files that are to be treated as objects which maintain their original format) is considered as instrumental in order to facilitate all data management processes in the BIF and address certain syntactic interoperability aspects. To this end, the data that are processed in the Data Handler are always transformed to the JSON format prior to any data handling and manipulation processes, irrespectively if they were collected in JSON, XML or tabular formats (like csv) by the Data Ingester & Fetcher. However, this decision imposes a relevant restriction on the access and provisioning of the building-related data that is also expected to happen in the JSON format.

4.6 INSTALLATION INSTRUCTIONS

The Data Handler User Interface is served as a web application and does not require the installation of any component by the user. Detailed instructions for the Data Handler deployment are provided in the related private code repository and all subcomponents are already packaged as Docker containers to speed up the process.

4.7 LICENSING

The BIMERR Data Handler is a closed source component.

5. DATA STORAGE & INDEXING

5.1 OVERVIEW

The Data Storage & Indexing subcomponent is responsible for persisting a plethora of building-related data, along with their associated metadata and sensitive data, in a secure and reliable manner in the BIF. When all data handling processes are completed and the finalized data payload is available (validated, linked and mapped to the applicable BIMERR data model), the data are stored in a non-relational (NoSQL) database in the Data Storage & Indexing component. The NoSQL paradigm has been preferred over a relational database, as it is inherently scalable and allows the optimized management of big data. To facilitate better search performance, the specific subcomponent applies data indexing on the basis of different indexes (based on the respective data model and the metadata). Stored and indexed data are available from the database to authorized applications and users through the “Building Information Secure Provisioning” component with the help of the Data Handler.

In particular, the functionalities of the Data Storage & Indexing subcomponent are enlisted as follows:

- **Storage of data collection jobs and their associated configuration files:** The Data Storage & Indexing component acquires from the Data Ingestor & Fetcher and the Model Mapper subcomponents, the data collection job profile and the respective parts of the configuration file that are created at the data collection configuration time and stores them in the BIF relational database.
- **Storage of data:** Once the data are appropriately ingested and handled, the processed data along with the processed sample are forwarded from the Data Handler subcomponent to the Data Storage & Indexing, which in turn stores them in the BIF NoSQL database.
- **Storage of assets’ metadata:** Along with the processed data assets, the Data Storage & Indexing subcomponent receives the accompanying metadata for storage in the BIF, creating the appropriate links between the stored data and their metadata information.

- **Indexing of data:** After completing the storage of data in the BIF NoSQL database, the Data Storage & Indexing subcomponent associates appropriate indexes with each data document in a search engine, to enable faster queries in a highly scalable manner.
- **Indexing of assets' metadata:** After completing the storage of metadata in the BIF relational database, the Data Storage & Indexing subcomponent associates appropriate indexes with each metadata record, to prepare and eventually offer a quality search experience.
- **Temporary storage of intermediate files:** Between the data ingestion and handling steps, intermediate configuration and data files are created for temporary storage, enabling the pause of a process and its continuation at a later stage. Such files are available in a carefully selected data lake for performance purposes, but also serves for traceability purposes, in case a step is not successfully completed.
- **Secure, encrypted storage of sensitive data:** A dedicated database is deployed to secure, persist and tightly control access to the sensitive parameters for the API calls (such as tokens, API keys, usernames and passwords) as well as effectively manage such secret codes.

5.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The Data Storage & Indexing builds on state-of-the-art data storage and indexing technologies including: PostgreSQL⁴ as the BIF relational database, MongoDB⁹ as the BIF NoSQL database, Elasticsearch⁸ as the BIF search optimization and indexing engine, MinIO⁵ as the BIF data lake and Vault⁶ for the BIF secrets and sensitive data storage. Such layers along with the different technologies are depicted in Figure 5-1.

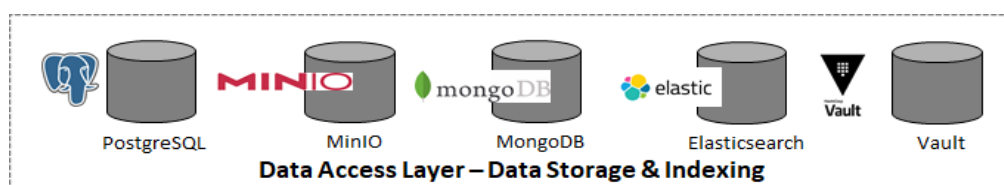


Figure 5-1: Architecture of the Data Storage & Indexing under the BIMERR Information Collection and Enrichment Component

⁹ <https://www.mongodb.com/>

The Data Storage & Indexing utilizes the open-source technologies defined in Table 5-1.

Table 5-1: Technologies and libraries used in the Data Storage & Indexing, along with their licenses

Name of the Library	Version	License
PostgreSQL	12.2	PostgreSQL License (similar to BSD/MIT)
MinIO	-	Apache License 2.0
MongoDB	4.4	Apache License 2.0
Elasticsearch	7.6.0	Elastic License
Vault	-	Mozilla Public License 2.0

5.3 API DOCUMENTATION

There are no external APIs that are exposed by the Data Storage & Indexing to accompany its final version.

5.4 CHANGES INTRODUCED IN FINAL RELEASE

Data Storage & Indexing has not undergone significant changes in its final release as only bug fixing, performance and scalability improvements have been essentially performed.

5.5 ASSUMPTIONS AND RESTRICTIONS

In order to ensure that all the BIF related data are appropriately stored, a polyglot data persistence layer bringing the best of breed of different data storage and indexing solutions was designed and developed. Even though such a decision increases the complexity of the BIF data access layer, it was considered as essential in order to increase reliability, trust and effectiveness of the end-to-end data collection processes.

5.6 INSTALLATION INSTRUCTIONS

Detailed instructions for the Data Storage & Indexing deployment are provided in the related private code repository and all loaders to the different databases are already packaged as Docker containers to speed up the process.

5.7 LICENSING

The BIMERR Data Storage & Indexing is a closed source component.

6. MASTER CONTROLLER

6.1 OVERVIEW

The Master Controller subcomponent is responsible for the orchestration and execution (once and according to a schedule) of all services for building-related data ingestion, handling and storage loading. To this end, the Master Controller effectively communicates with the appropriate subcomponents in the BIF (namely the Data Ingester & Fetcher, the Data Handler and the Model Mapper that is described in D4.5) getting or providing them with the configuration file of each data collection job which contains all necessary information to perform their tasks. In parallel, the Master Controller is tasked to ensure that the data collection jobs are allocated with the appropriate resources for their execution while keeping track of their execution status.

In more detail, the functionalities that are supported by the Master Controller enlist the following aspects:

- **Scheduling of data ingestion, data handling and data storage based on the user-defined configuration files:** At the ingestion configuration time for the API modality, the user should specify a retrieval period and schedule (in case it is not a one-off retrieval process). The data handling process may also occur at scheduled times defined by the user. Such settings are typically stored in the configuration file of the specific data collection job and are leveraged by the Master Controller when scheduling the different data ingestion and processing tasks.
- **Monitoring of the status of the data ingestion, data handling and data storage execution processes:** The Master Controller provisions the necessary computation resources for the execution of the different steps (i.e., ingestion, handling and storage) of a data collection job and anticipates status messages from the respective BIF subcomponents, in order to continue with the next scheduled activity, make any needed rescheduling or proceed with error handling.
- **Management and scaling of containerised data ingestion, data handling and data storage services:** The Master Controller is responsible for the orchestration and monitoring of the containerised services for data ingestion, handling and storage when a data collection job is executed. In this context, the Master Controller

automatically manages load balancing, scaling and failure management (e.g. restarting a container that failed without encountering any data loss).

- **Management of the lifecycle of the configuration file, persistence and export:** The Master Controller acquires and stores the configuration file in collaboration with the different BIF components that utilize or update it. The Master Controller also exports and grants access to the most up-to-date version of the configuration file to the subcomponents that need it for their operations (e.g., to the Data Ingestor & Fetcher upon the data retrieval initiation).
- **Applying an error handling strategy:** The Master Controller is responsible for managing errors that occur during the execution of a data collection job at any stage, from ingesting the data to processing and storing them. If an error happens at ingestion time, the Master Controller will apply the error handling strategy defined by the user at the configuration time (for example, providing the number of additional requests in the API data collection modality), by sending corresponding action triggers to the respective subcomponent.

6.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The Master Controller builds on state-of-the art technologies across three layers:

- The Orchestration Engine, based on Kubernetes¹⁰ for automating deployment, scaling, and management of containerized services for data ingestion (from the Data Ingestor & Indexer), data handling (from the Data Handler) and data storage (from the Storage & Loader).
- The Business Logic Layer, containing the different packages of the Master Controller Backend that are based in the Nest (NodeJS¹¹) web framework and implements the scheduling functionality that is actually facilitated through a RabbitMQ¹² message broker system, that sends the appropriate messages/triggers to the respective subcomponents.

¹⁰ <https://kubernetes.io/>

¹¹ <https://nestjs.com/>

¹² <https://www.rabbitmq.com/>

- The Data Access Layer that essentially refers to the BIF Storage and Indexing that mostly utilizes MinIO5, Vault6 and PostgreSQL4, for the different Master Controller operations.

These layers along with the different technologies are depicted in Figure 6-1.

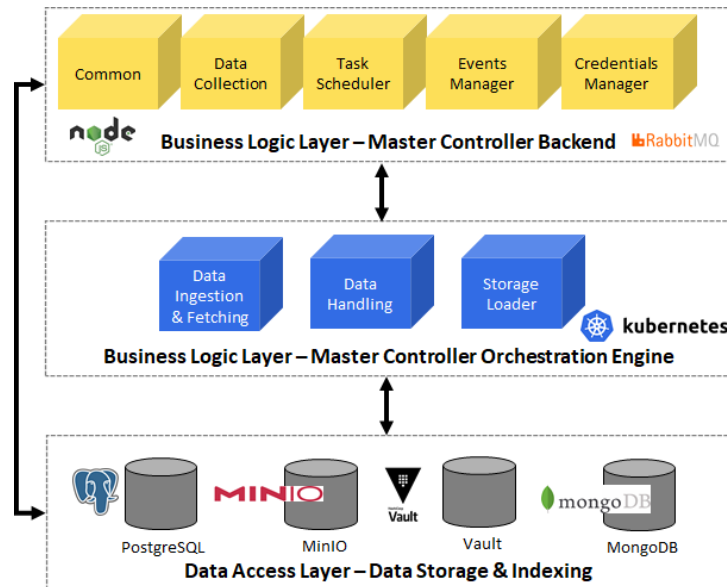


Figure 6-1: Architecture of the Master Controller under the BIMERR Information Collection and Enrichment Component

In addition, the Master Controller is written in Typescript¹³ and utilizes the open-source technologies as defined in Table 6-1.

Table 6- 1: Technologies and libraries used in the Master Controller, along with their licenses

Name of the Library	Version	License
Nest NodeJS Web Framework	7	MIT
TypeORM	-	MIT
Kubernetes	1.18	Apache License 2.0
RabbitMQ	3.8.2	Mozilla Public License
PostgreSQL	12.2	PostgreSQL License (similar to BSD/MIT)
MinIO	-	Apache License 2.0

¹³ <https://www.typescriptlang.org/>

Name of the Library	Version	License
MongoDB	4.4	Apache License 2.0
Vault	-	Mozilla Public License 2.0

6.3 API DOCUMENTATION

There are no external APIs that are exposed by the Master Controller to accompany its final version. The Master Controller typically communicates with the other components and services under the BIMERR Information Collection and Enrichment Component and the Building Semantic Modelling Component through its messaging functionality, and there are not any APIs needed.

6.4 CHANGES INTRODUCED IN FINAL RELEASE

In respect to the 1st release of the Master Controller component, the following improvements and enhancements has been introduced in this final release upon the partners' feedback received on the initial release:

- Scheduling & Orchestration Refactoring to address certain limitations related to the recurring jobs.
- Bug fixing.

6.5 ASSUMPTIONS AND RESTRICTIONS

In the present, final release of the Master Controller, certain assumptions (that in certain cases, represent restrictions for the Master Controller) were taken:

- The Master Controller spawns different virtual machines to execute the different data services containers according to the scheduling triggers contained in the configuration of each data collection job. Once a certain step (e.g., data ingestion or data handling) is completed, the respective virtual machine is destroyed. Such a decision was taken considering the optimization of the allocated resources.
- The Master Controller currently treats all stages of the data collection jobs and all data collection jobs with the same priority without any preferential allocation of resources.

- The Master Controller currently applies a basic error handling strategy in case a data collection job fails (e.g., in the case of APIs, allowing a minimum configuration to the user to define the number of retries). However, there are obviously many errors that cannot be anticipated in advance as it would require an even more complex configuration provided by the user in a data collection job.

6.6 INSTALLATION INSTRUCTIONS

Detailed instructions for the Master Controller deployment are provided in the related private code repository.

6.7 LICENSING

The BIMERR Master Controller is a closed source component.

7. KNOWLEDGE GRAPH GENERATOR

7.1 OVERVIEW

The Knowledge Graph Generator (KGG) subcomponent is responsible for generating knowledge graphs or the semantic representation of existing data. The RDF¹⁴ data is generated using mappings, which are sets of rules that establish a relation between the ontology schema (classes and properties) and the data source schema. The RDF graph generated is finally stored in a Triple Store that includes a SPARQL¹⁵ endpoint to allow the request of data using semantic queries, which is the main interface provided by the KGG. Such query interface is open to offer non personal or close data to systems external to BIMERR in a linked open data fashion following BIMERR ontologies, and therefore data models, increasing the interoperability. The KGG also includes the connectors and processing functions for some external sources that might not be connected to the Data Ingester & Fetcher if the target data is not going to be shared between different BIMERR applications. Another use case of the KGG is to provide an external access point to non-confidential data that can be linked with other knowledge graphs in a Linked Open Data way.

The KGG has configured to generate knowledge graph for three different types of data, more precisely meteorological, building occupancy behavior and material data types. For each of them, different sources and file formats have been used, for example, for meteorology, the "EnergyPlus", "Climate.OneBuilding Weather" and "OpenWeatherMap" repositories have been used, with "EPW" (EnergyPlus Weather) and "Json" file formats. For buildings, the files corresponding to "ObXML" have been used, with an "xml" file format. Finally, the "EnergyPlus" repository has been used for the materials data, with the "IDF" file format.

¹⁴ <https://www.w3.org/TR/rdf-primer/>

¹⁵ <http://www.w3.org/TR/sparql11-query/>

7.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The KGG implements a sequence of software sub-modules, each specialized in handling a specific task within the transformation pipeline. The internal details of the implementation are shown in Figure 7-1. At this point of development, the data to be used and transformed by this BIF component comes from the following domains: Weather, Building Components and Building Materials. The data are taken from “external” resources and are not provided by internal BIMERR components or applications, as the data generated by these modules might be confidential and access policies might be applied.

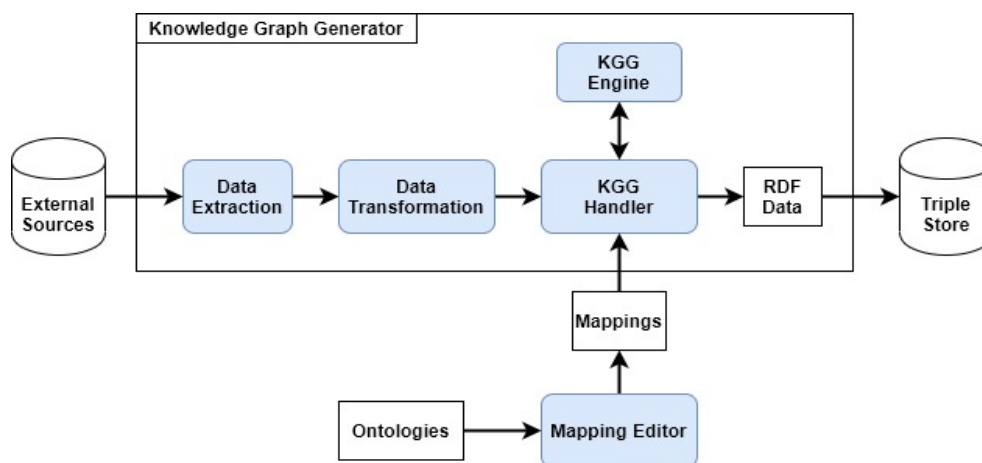


Figure 7- 1: Architecture of the Knowledge Graph Generator

The first step in the conversion process involves the acquisition of information. The Data Extraction module fetches data from external sources that incorporate APIs to easily get access to their databases.

The Data Transformation stage performs the conversion from the original format in which data was collected to a format supported by the KGG Engine. The engine can handle the following formats: CSV, XML, JSON and idf. The transformer is implemented as a series of sub-modules, where each of them is responsible for transforming a particular data format. In particular transformation from xml and idf files to json have been carried out in order to prepare the data for the next step.

Having the data preprocessed, the next step consists of the mappings codification. The mappings are expressed in a file containing a set of rules relating the ontology and the data source schemas that should be provided by the user. While there are a number of mappings already defined, a user could construct new mappings using a simple text editor and taking as input the ontologies provided by the Building Semantic Modelling component (described in D4.5) and the preprocessed data obtained from the previous step. The codification of the rules follows the RML¹⁶ mapping language and Helio¹⁷ mappings. RML mappings have been used for the transformation of data belonging to epw files, due to their large size and long processing time. On the other hand, the Helio mapping language has been used for use with files smaller than epw files, because they are easier to write and for this type of small files the transformation process is faster.

The KGG Handler has to be provided with the data source of the processed file and its corresponding mappings in order to be able to perform the transformation process. Then the user can run the engine to initiate the conversion process. This is when the KGG engine comes into play; parsing the source data file and generating the corresponding <subject-predicate-object> triples. The materialized data is finally stored manually in a specialized RDF-graph triple store, (i.e. Virtuoso) and made accessible by means of a SPARQL endpoint.

The Knowledge Graph Generator has been developed using Python3 as the main programming language, and also Java 8. Table 7-1 indicates all the libraries and software packages used in the KGG component.

Table 7- 1: Technologies and libraries used in the Knowledge Graph Generator, along their licenses

Name of the Library	Version	License
Helio-materializer	0.3.11	Apache License 2.0
RMLMapper-java	4.10	MIT
Virtuoso	6.1	GPL Version 2
Django	3.0.7	Creative Commons 3.0
Flask	1.1.1	BSD License

¹⁶ <https://rml.io/specs/rml/#example-mapping-a-csv-data-source>

¹⁷ <https://github.com/oeg-upm/helio/wiki/Helio-Materialiser-for-Users#helio-mappings>

Name of the Library	Version	License
pyidf	0.2.1	Apache License 2.0
pyepw	0.1	Apache License 2.0

7.3 API DOCUMENTATION

The main interface provided by the KGG is the SPARQL endpoint from where any user can retrieve data using the SPARQL language. The endpoint is available at <https://data.bimerr.iot.linkeddata.es/sparql>

In addition, HTML documentation is provided, for example:

- For meteorological data:
http://data.bimerr.iot.linkeddata.es/resource/weather/CeilingHeightMeasurement/ESP_Bilbao-AP_2017-10-16-2
- For occupancy behaviour data:
https://data.bimerr.iot.linkeddata.es/resource/16Y9ahzJ5DjhmNdLJbEHS9_1wyeFgulzFhhJwSLyhKQk2_31gUPijz4Qvpt6sZCOGVg
- For materials data:
https://data.bimerr.iot.linkeddata.es/resource/Material/G01_16mm_gypsum_board

The documentation of the code used during the process to generate the knowledge graph or each domain is available in GitHub:

- For meteorological data: <https://github.com/oeg-upm/bimerr-epw>
- For occupancy behaviour data: <https://github.com/oeg-upm/bimerr-obxml>
- For materials data: <https://github.com/oeg-upm/bimerr-materials>

In addition, the following converters are available for transforming original files to json:

- For meteorological data:
https://weather.bimerr.iot.linkeddata.es/apidocs/#/default/post_epw
- For occupancy behaviour data:
https://weather.bimerr.iot.linkeddata.es/apidocs/#/default/post_obxml
- For materials data:
https://weather.bimerr.iot.linkeddata.es/apidocs/#/default/post_idf

7.4 CHANGES INTRODUCED IN FINAL RELEASE

In the final release the technology stack used has been modified to incorporate both Helio and RML mappings. Also, the publication of HTML version of the RDF is based on Helio features rather than in Virtuoso ones, as Helio offers more configuration features, (e.g., template generation).

In addition, as already mentioned, converter from original files to json has been necessary, being such converters offered as services independently of the knowledge graph.

7.5 ASSUMPTIONS AND RESTRICTIONS

The data for which transformation mechanisms are provided, is data that can be published and linked with other data resources and general obXML files.

Transformations to json from original file formats have been developed in order to enable mapping correspondence that were not possible to establish due to current mapping language limitations.

For .epw files some fields have been discarded for the transformation, as these are not involved in energy-based renovation process calculations. These fields are included in the header and contains metadata about the provenance of the data. The fields kept for transformation are the location (line 1), the extreme periods observed (line 3) and measures of the soil in different depths (line 4).

Another limitation involves the manual triggering of the KGG every time the external data is updated. Before the transformation process, the maintainer of the system has to verify the source schema, confirming that the mapping rules defined initially still apply. In case the data schema suffers modifications, the maintainer needs to change the corresponding mapping files, allocated within the KGG, following the RML specification.

7.6 INSTALLATION INSTRUCTIONS

The KGG is provided as a web service, and does not require the installation of any component by the user. In order to be able to visualize and use the data belonging to the knowledge graph, a Virtuoso SPARQL endpoint¹⁸ has been deployed, which can be consulted. This endpoint also contains the Helio system¹⁹, which is used to publish the data included in the knowledge graph.

7.7 LICENSING

The Knowledge Graph Generator is licensed under the Apache 2.0 license.

The different licenses for each of the files used are provided below.

The License on which this source is based, “Energy Plus, Copyright (c) 1996-2019”, is open-source, accepting contributions to the EnergyPlus source, test file utilities, documentation and other materials distributed with the program.

The license offered by OpenWeatherMap, distributes the data provided under the terms of the Creative Commons Attribution-ShareAlike 4.0 International license (CCBY-SA 4.0), under which the data can be freely used via the API for non-commercial or commercial purposes. Therefore, the name OpenWeatherMap should be mentioned as a source of weather information in a visible part of the application to be used/created.

The obXML license is: LAWENCE BERKELEY NATIONAL LABORATORY RESEARCH & DEVELOPMENT, NON-COMMERCIAL USE ONLY, LICENSE

The License on which the idf files for materials is based, “Energy Plus, Copyright (c) 1996-2019”, is open-source, accepting contributions to the EnergyPlus source, test file utilities, documentation and other materials distributed with the program.

¹⁸ <https://data.bimerr.iot.linkeddata.es/sparql>

¹⁹ <https://oeg-upm.github.io/helio/>

8. END-TO-END USAGE WALKTHROUGH TO THE BIMERR INFORMATION COLLECTION & ENRICHMENT COMPONENT

The following section present an end-to end usage walkthrough to the final version of the BIMERR Information Collection & Enrichment Component by presenting screenshot.

8.1 INITIATE A DATA COLLECTION JOB

When users (e.g., a BIMERR Application owner/developer) log-in to the BIMERR Information Collection & Enrichment Component (Figure 8-1), they first view a list of all the existing Data Collection Jobs they created in the past and their details (such as the stages that comprised each job and their execution status). If required, by clicking on any of the existing jobs, they can edit or delete it.

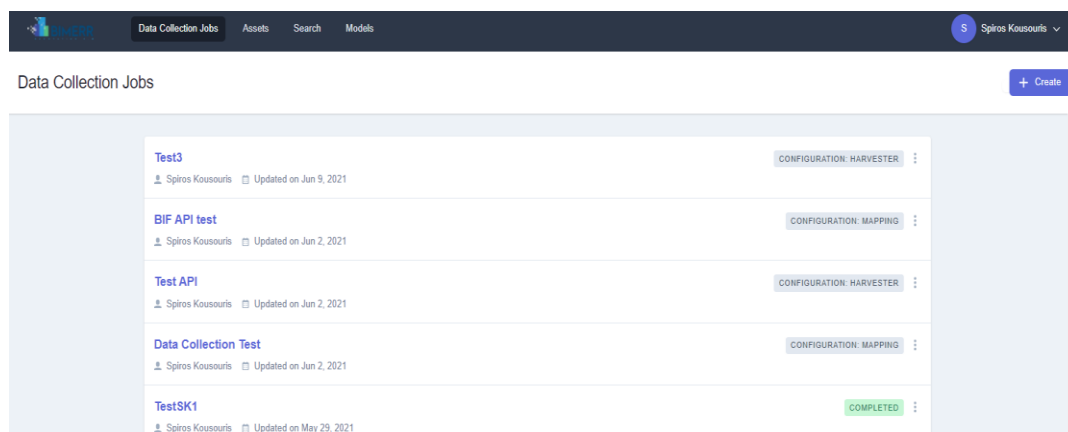


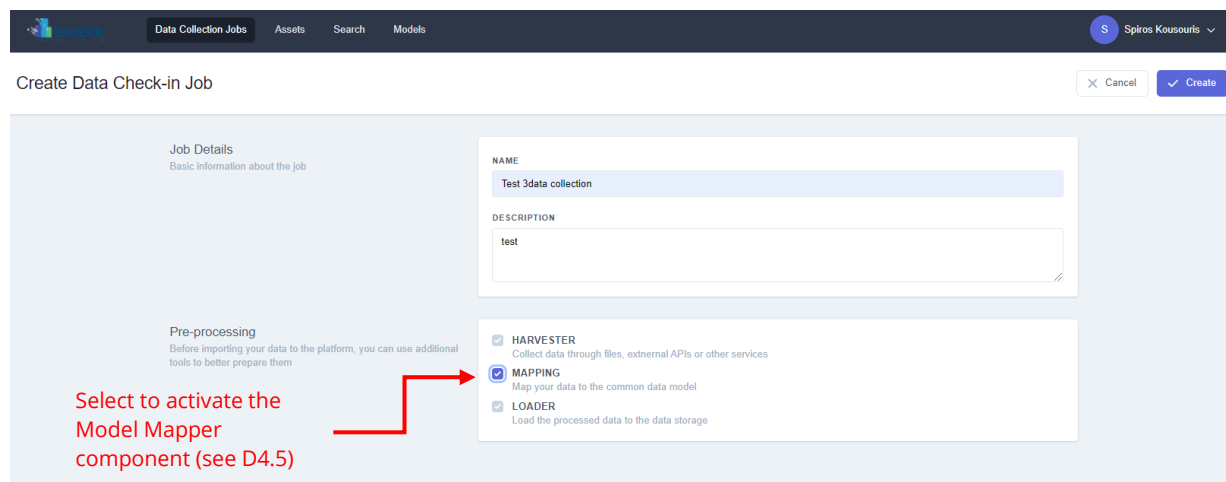
Figure 8-1: View of all user's data collection jobs

On the same screen, and by clicking on the "+ Create" button, users can create a new Data Collection Job.

As shown in Figure 8-2, to create a new data collection job, users shall initially define its basic information such as its name, its description and the pre-processing steps that need to be applied. In the latter case, the Harvester and Loader tools are pre-activated to

properly collect and store the new data, while the Mapping tool is optional. At any time, users can cancel their new Data Check-in Job, by clicking on the “x Cancel” button.

***Note:** When the Mapping option is selected, users can upload their data also through external API or utilising the (internal) API offered by BIF; if the mapping tool is not selected, users can only upload their data only as files (Other type).*



Create Data Check-in Job

Job Details
Basic information about the job

NAME
Test 3data collection

DESCRIPTION
test

Pre-processing
Before importing your data to the platform, you can use additional tools to better prepare them

Select to activate the Model Mapper component (see D4.5)

☒ HARVESTER
Collect data through files, external APIs or other services

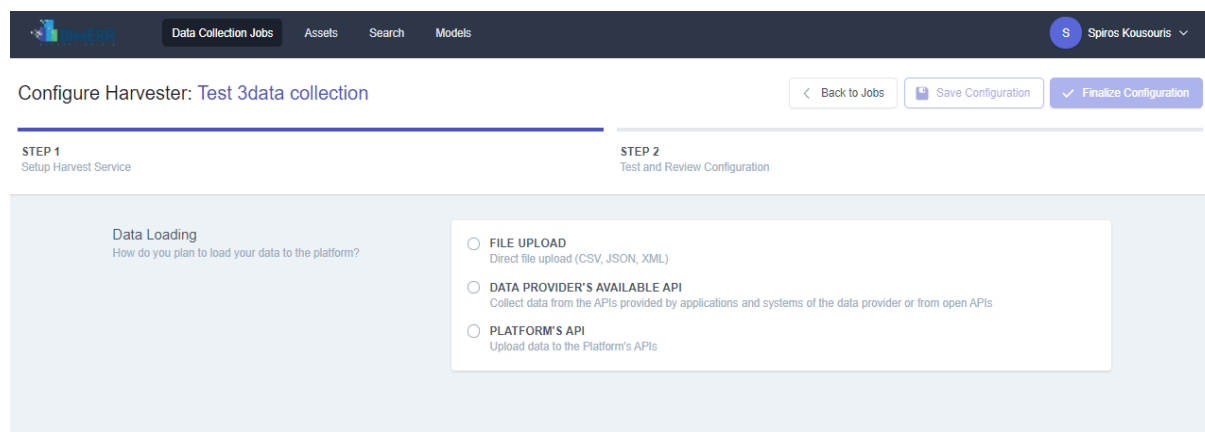
☒ MAPPING
Map your data to the common data model

☒ LOADER
Load the processed data to the data storage

Cancel Create

Figure 8-2: Creation of a new Data Collection Job

Upon the creation of a new data collection job, as shown in Figure 8-3, the users need to select their preferred data collection method.



Configure Harvester: Test 3data collection

Back to Jobs Save Configuration Finalize Configuration

STEP 1
Setup Harvest Service

STEP 2
Test and Review Configuration

Data Loading
How do you plan to load your data to the platform?

☒ FILE UPLOAD
Direct file upload (CSV, JSON, XML)

☐ DATA PROVIDER'S AVAILABLE API
Collect data from the APIs provided by applications and systems of the data provider or from open APIs

☐ PLATFORM'S API
Upload data to the Platform's APIs

Figure 8-3: Data Collection configuration – Data Loading method selection

In general, the users can choose to load their data to BIF: a) through Direct file(s) upload (CSV, JSON, XML, Other), b) via automatic retrieval through API exposed by the data

provider (e.g., BIMERR applications) and c) upload their data utilising the Platform's API (i.e., the internal API offered by BIF).

Upon selecting the preferred data loading method, the users are notified that this cannot change in the future once set (for the specific dataset) and are asked to confirm their selection (Figure 8-4).

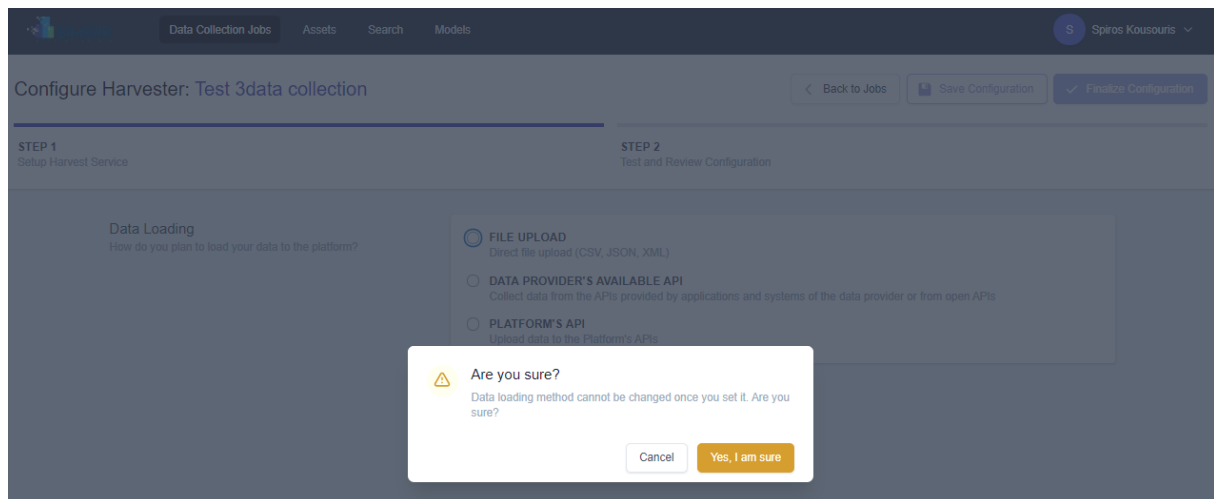
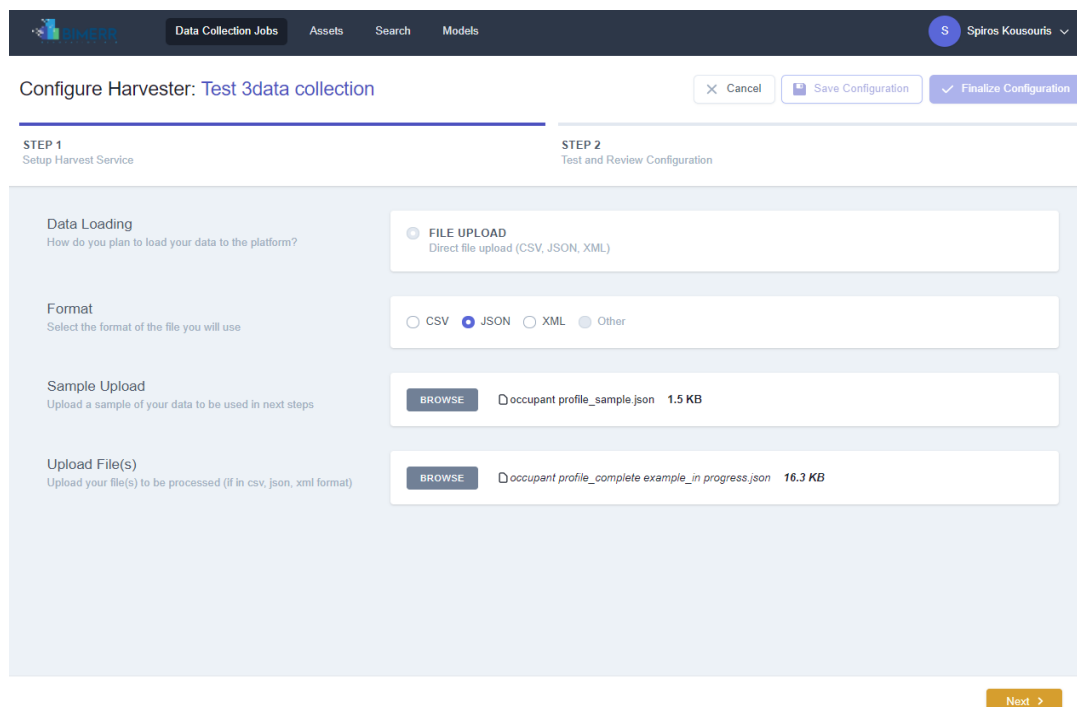


Figure 8-4 Data Collection configuration - Notification for final confirmation of modality selection

8.2 CONFIGURE A DATA COLLECTION JOB FOR FILES UPLOAD

In case the users select to upload file(s) (Figure 8-3), they need to define step-by-step the data collection parameters. As shown in Figure 8-5, initially the format of the file needs to be defined; the users can select among CSV, JSON, XML and Other formats. Depending on the selection, the user is prompted to upload a sample of the data, to be used in the next configuration steps and/or the file(s) to be processed.



Configure Harvester: Test 3data collection

Cancel Save Configuration Finalize Configuration

STEP 1
Setup Harvest Service

STEP 2
Test and Review Configuration

Data Loading
How do you plan to load your data to the platform?

Format
Select the format of the file you will use

Sample Upload
Upload a sample of your data to be used in next steps

Upload File(s)
Upload your file(s) to be processed (if in csv, json, xml format)

FILE UPLOAD
Direct file upload (CSV, JSON, XML)

☐ CSV ☒ JSON ☐ XML ☐ Other


BROWSE 1.5 KB

BROWSE 16.3 KB

Next >

Figure 8-5: Configure data collection for Files

By clicking next, and upon completion of the files upload, the users can view the data sample they have uploaded, as shown in Figure 8-6.


Data Collection Jobs
Assets
Search
Models
S Spiros Kousouris

Configure Harvester: Test 3data collection
Cancel
Save Configuration
Finalize Configuration

STEP 1
Setup Harvest Service

STEP 2
Test and Review Configuration

ADDED FILES

occupant profile_complete example_in progress.json
16.3 KB


Data Sample
The details of the data sample that was uploaded

```
[
{
  "Building ID": "Building_1",
  "Building Description": "A building which contains 2 spaces, both residences and offices.",
  "Building Type": "Mixed",
  "Space ID": "S0",
  "Space Description": "Residence for 1 occupant",
  "Space Type": "ResidentialOwn",
  "Space MaxNumOccupants": 1,
  "OccupantID": "O1",
  "Occupant Name": "OccupantO1S0",
  "BehaviorID": "B_Therm1",
  "Behavior Description": "When temperature is outside the thermal comfort range set to 22.5 deg.C",
  "Need_ID": "S",
  "Need Type": "Thermal",
  "maximum temperature": 95,
  "Meeting ID": "M1",
  "Meeting Duration": "120",
  "Meeting Start": "29/04/2020 18:00",
  "Meeting End": "29/04/2020 18:02"
}
]
```

< Previous

Figure 8-6: Configure data collection for Files – Review of uploaded data sample (if applicable)

Upon review of their data sample, the users can save their configuration; once the data upload is completed, they are notified that the data sample has been saved successfully (Figure 8-7).


Data Collection Jobs
Assets
Search
Models
S Spiros Kousouris

Configure Harvester: Test 3data collection
< Back to Jobs
Save Configuration
Finalize Configuration

STEP 1
Setup Harvest Service

STEP 2
Test and Review Configuration

Data Sample
The details of the data sample that was uploaded

```

{
  "Building ID": "Building_1",
  "Building Description": "A building which contains 2 spaces, both residences and offices.",
  "Building Type": "Mixed",
  "Space ID": "S0",
  "Space Description": "Residence for 1 occupant",
  "Space Type": "ResidentialOwn",
  "Space MaxNumOccupants": 1,
  "OccupantID": "O1",
  "Occupant Name": "OccupantO1S0",
  "BehaviorID": "B_Therm1",
  "Behavior Description": "When temperature is outside the thermal comfort range set to 22.5 deg.C",
  "Need_ID": "5",
  "Need Type": "Thermal",
  "maximum temperature": 95,
  "Meeting ID": "M1",
  "Meeting Duration": "120",
  "Meeting Start": "29/04/2020 18:00",
  "Meeting End": "29/04/2020 18:02"
},
{
  "Building ID": "Building_1",
  "Building Description": "A building which contains 3 spaces, both residences and offices."
}

```

< Previous

Success
Harvester configuration saved successfully

Figure 8-7: Configure data collection for Files - Notification for successful saving of the data upload

Once the configuration is saved, the users can finalize their configuration by clicking on the appropriate button. A pop-up window appears, confirming the successful completion of the Harvester configuration. The users now can select to proceed with the Mapping configuration or return to the main screen with all their available jobs. It needs to be noted that the mapping configuration is documented in detail in D4.5.

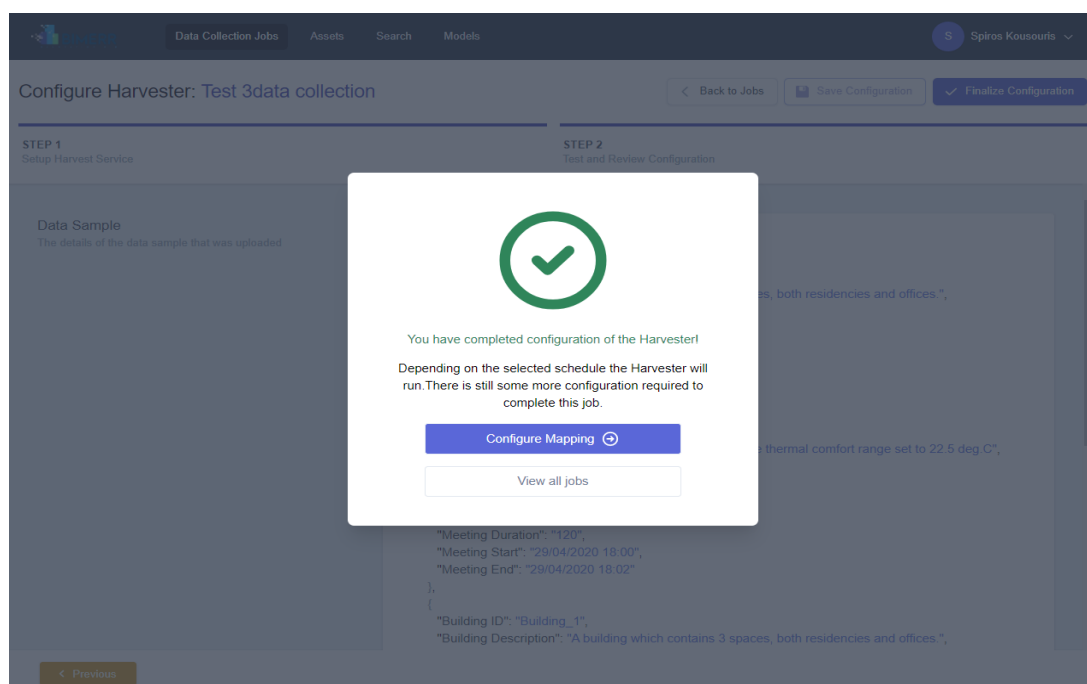


Figure 8-8: Configure data collection for Files - Notification for completed configuration of the Harvester

8.3 CONFIGURE A DATA COLLECTION JOB THROUGH DATA PROVIDERS' AVAILABLE API

In case the users select to upload their data to BIF via the "Data Provider's available API (i.e external API) mode (see Figure 8-3), they must first configure the Harvester as shown in Figure 8-9 In this first step, the users shall specify the authentication details, the method, URL & query body of the request, the pagination options, any request parameters and extra required headers, the retrieval settings and processing schedule and lastly the preferred error handling strategy for their data.

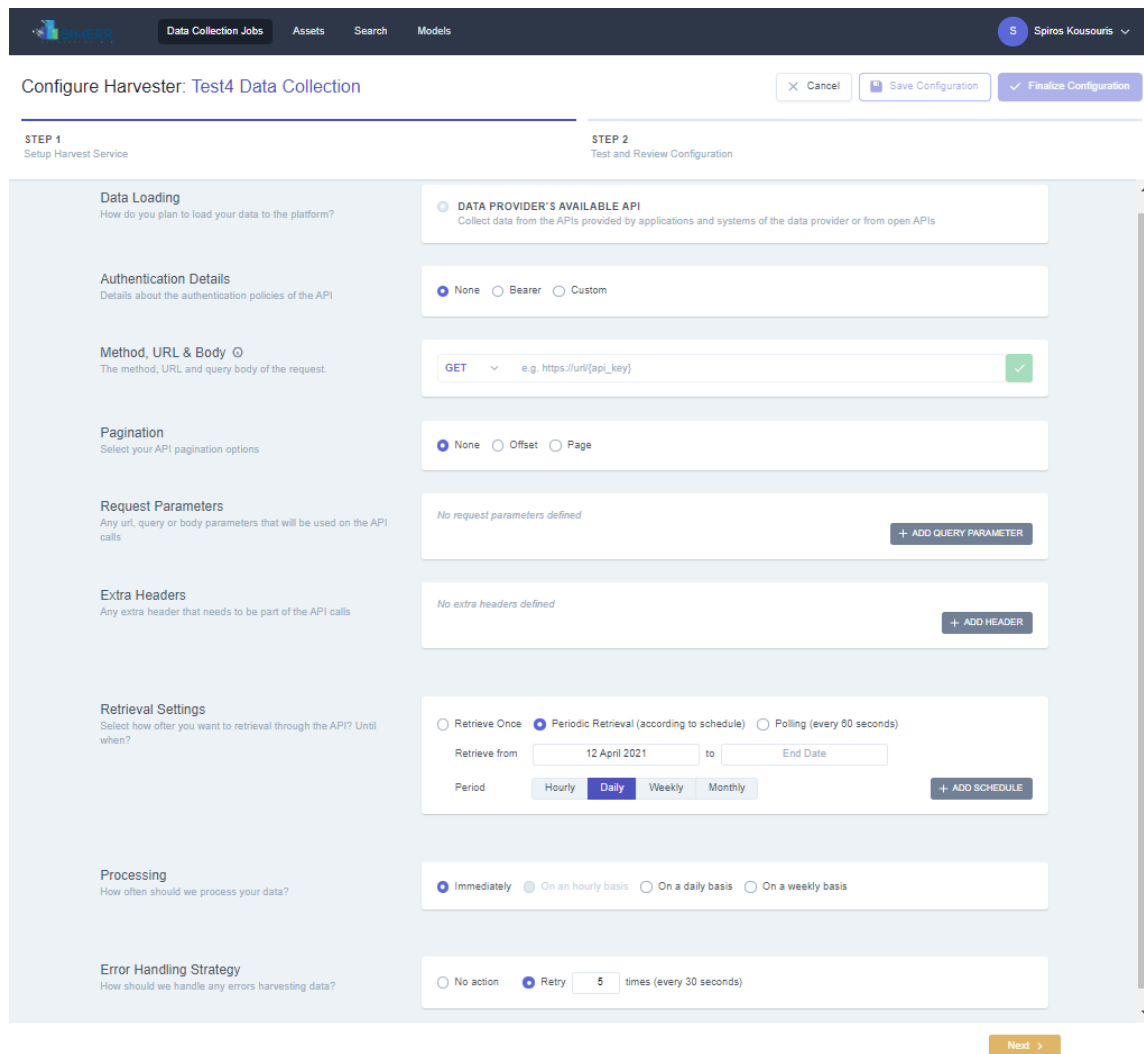
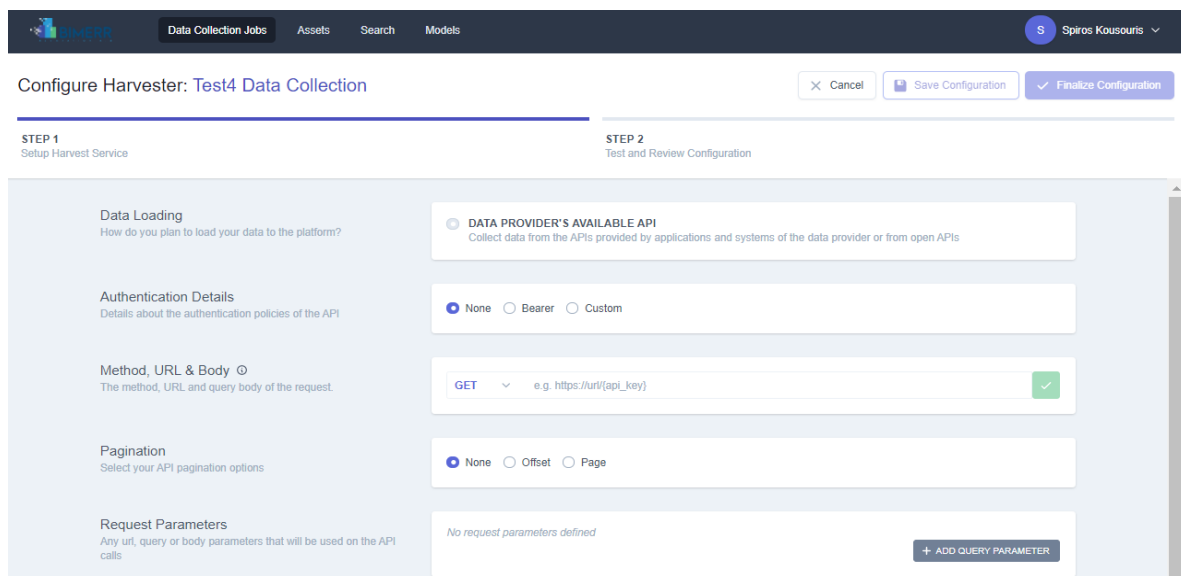


Figure 8-9: Data collection through external APIs - Harvester configuration

As shown in Figure 8-10 below, when specifying the authentication details, three options are available:

1. *None*, when advanced authentication is not required. In this case the tokens or API keys may be indicatively passed through the base URL or the extra headers.



Configure Harvester: Test4 Data Collection

STEP 1: Setup Harvest Service

Data Loading
How do you plan to load your data to the platform?

Authentication Details
Details about the authentication policies of the API

Method, URL & Body
The method, URL and query body of the request.

Pagination
Select your API pagination options

Request Parameters
Any uri, query or body parameters that will be used on the API calls

DATA PROVIDER'S AVAILABLE API
Collect data from the APIs provided by applications and systems of the data provider or from open APIs

None **Bearer** **Custom**

GET e.g. https://url/{api_key}

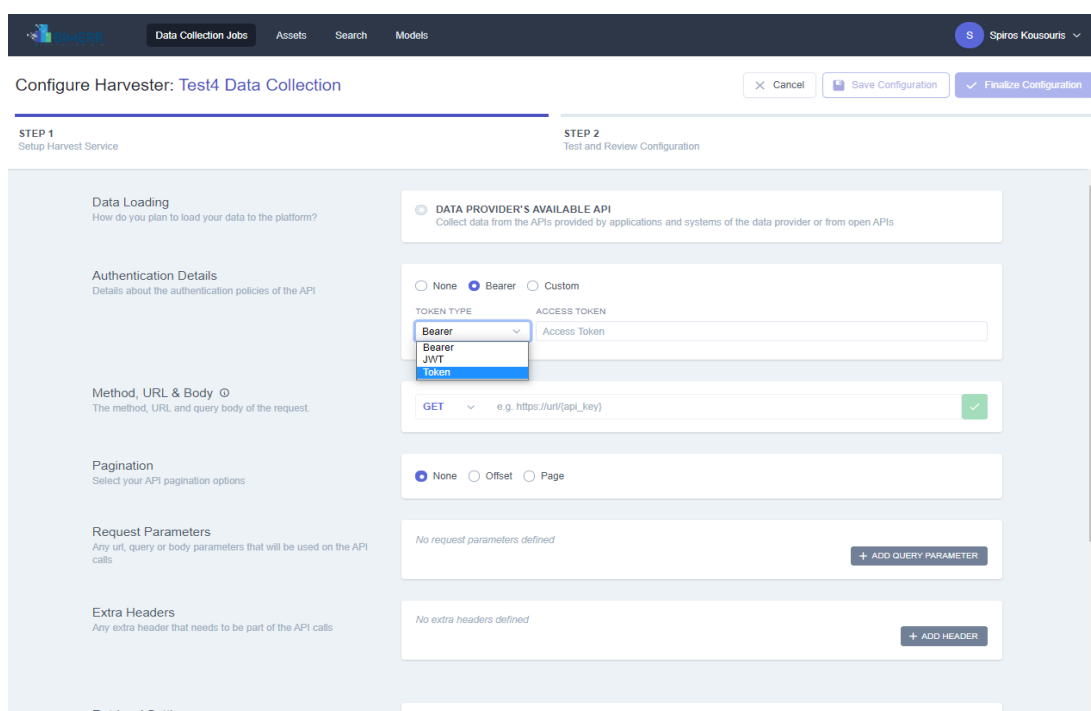
None **Offset** **Page**

No request parameters defined

+ ADD QUERY PARAMETER

Figure 8-10: Data collection through external APIs - No authentication required

2. *Bearer authentication*, which allows users to select the token type among Bearer, JWT and Token and provide the access token, as shown in Figure 8-11. By default, all tokens are considered as sensitive and are stored in an encrypted form, separately from the BIF data for security purposes.



Configure Harvester: Test4 Data Collection

STEP 1: Setup Harvest Service

Data Loading
How do you plan to load your data to the platform?

Authentication Details
Details about the authentication policies of the API

Method, URL & Body
The method, URL and query body of the request.

Pagination
Select your API pagination options

Request Parameters
Any uri, query or body parameters that will be used on the API calls

DATA PROVIDER'S AVAILABLE API
Collect data from the APIs provided by applications and systems of the data provider or from open APIs

None **Bearer** **Custom**

TOKEN TYPE **ACCESS TOKEN**

Bearer **Access Token**

Bearer **JWT** **Token**

GET e.g. https://url/{api_key}

None **Offset** **Page**

No request parameters defined

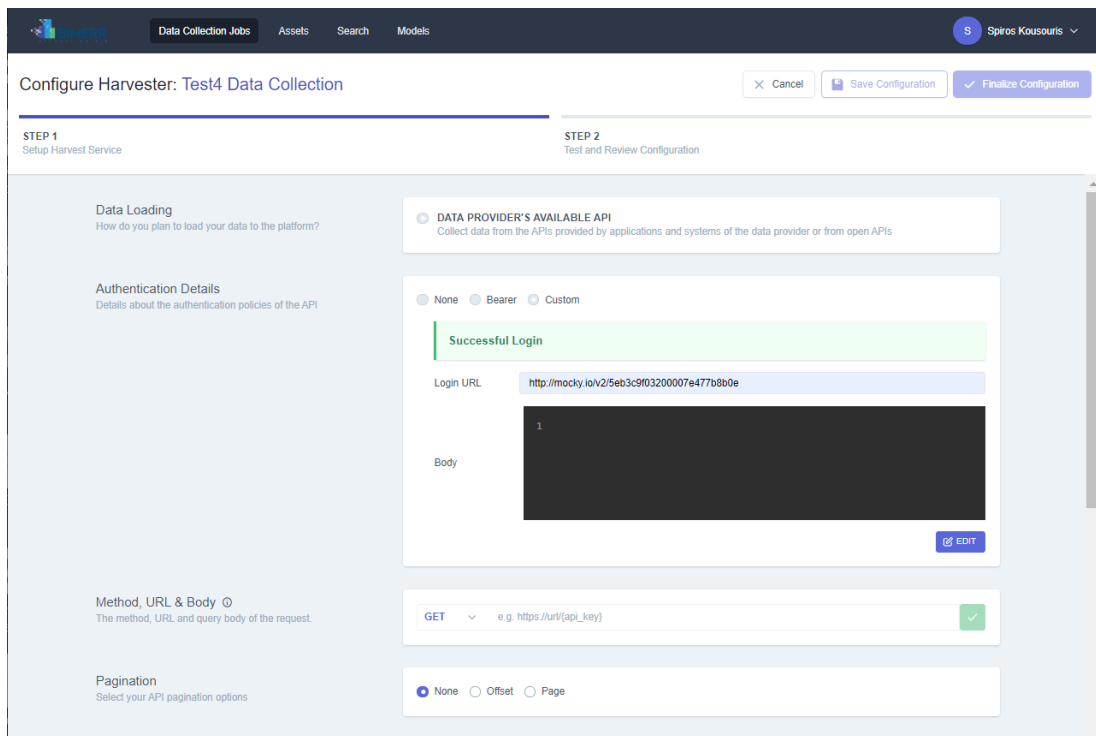
+ ADD QUERY PARAMETER

No extra headers defined

+ ADD HEADER

Figure 8-11: Data collection through external APIs - Bearer Authentication options

3. *Custom authentication*, which addresses the cases when a login URL and the request body needs to be filled in in order to retrieve the access token. A test login is performed (as shown in Figure 8-12) and the sensitive parameters that are retrieved (e.g., access token, refresh token, etc.) are securely stored in the BIF vault.



The screenshot shows the 'Configure Harvester: Test4 Data Collection' interface. It has a dark blue header with navigation tabs: 'Data Collection Jobs', 'Assets', 'Search', and 'Models'. A user profile 'Spiros Kousouris' is in the top right. The main area is divided into two steps: 'STEP 1 Setup Harvest Service' and 'STEP 2 Test and Review Configuration'. The 'Authentication Details' section is active, showing 'DATA PROVIDER'S AVAILABLE API' with a description. Below this, there are radio buttons for 'None', 'Bearer', and 'Custom' (selected). A green box labeled 'Successful Login' is visible. The 'Login URL' field contains 'http://mocky.io/v2/5eb3c9f03200007e477b8b0e'. The 'Body' field is a large black redacted area. Below the body field is an 'EDIT' button. The 'Method, URL & Body' section shows a dropdown for 'GET' and a text field with 'e.g. https://url(api_key)' and a green checkmark. The 'Pagination' section has radio buttons for 'None' (selected), 'Offset', and 'Page'.

Figure 8-12: Data collection through external APIs – Custom Authentication options

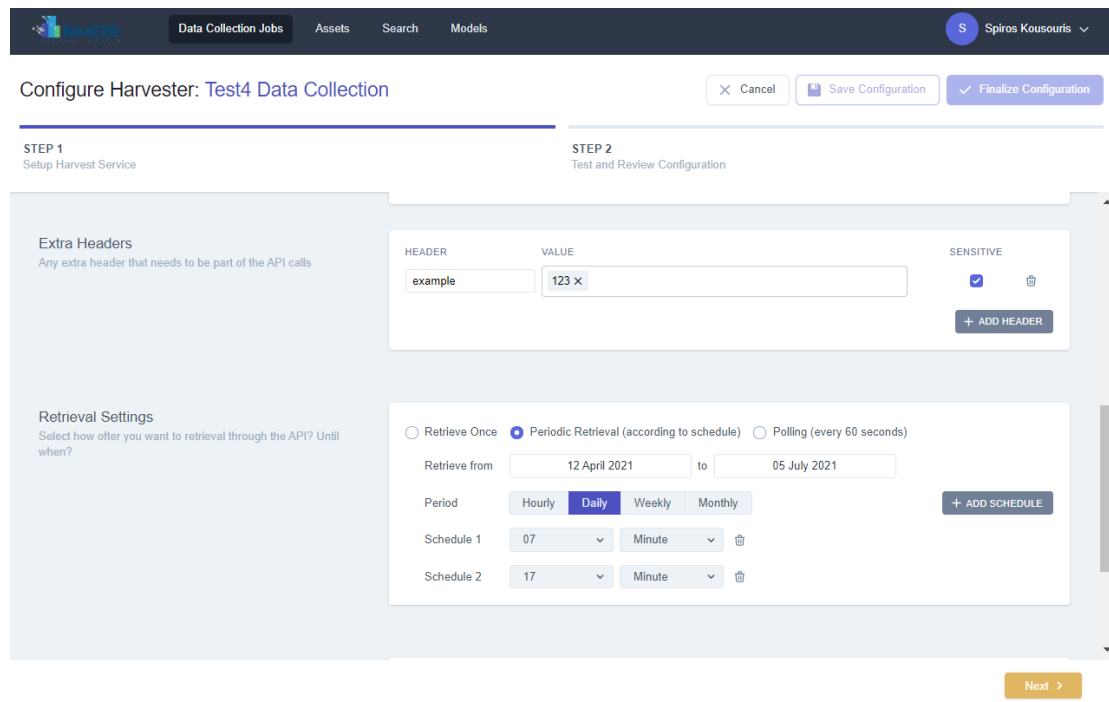


Figure 8-14: Data collection through external APIs - Extra Headers & Retrieval Settings

Figure 8-13: Data collection through external APIs - Authentication, Method, Pagination and Query Parameters

If option 1 (i.e., none authentication) is selected, as shown in Figure 8-13 above, the users shall define their preferred method (POST or GET) and specify the full API path according to the instructions provided, as well as the appropriate API pagination options if required (i.e., how many items are included in each page and the starting page/value).

By parsing the full API path, the query parameters are identified and displayed. The users then need to provide the values of the parameters and indicate whether any parameter is sensitive (in order to be stored in the BIF vault). In case the users have selected the POST method, the query body needs to be also filled in.

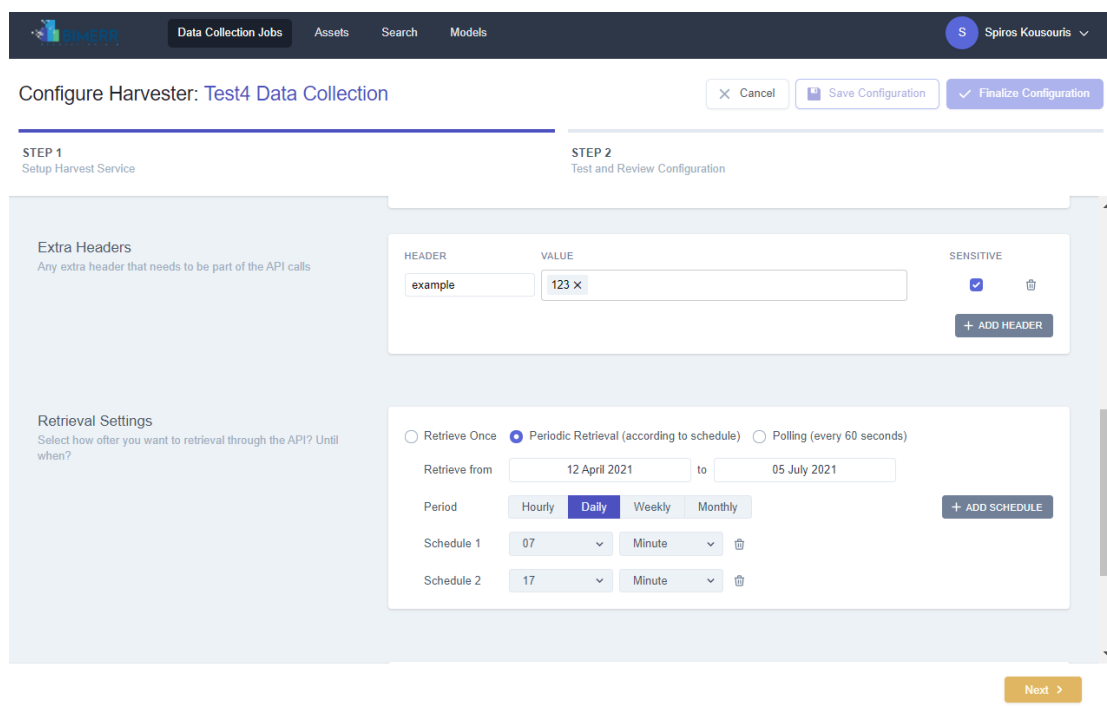


Figure 8-14: Data collection through external APIs - Extra Headers & Retrieval Settings

Following and as shown in Figure 8-14 above, the users are prompted to provide any extra headers that need to be part of the API call and to set the data Retrieval settings according to the following options:

1. *Retrieval of data once*, the users are prompted to define the exact retrieval date.
2. *Periodic retrieval of data*: the users are prompted to set their desired retrieval schedule by specifying the retrieval period and the intervals (i.e., hourly, daily, weekly and monthly) for the selected period. Moreover, the specific hour/day of retrieving the data can be set for each selected interval. According to the selected schedule, shown in Figure 8-14, data will be collected daily at 07:00 and 17:00, between April 12th and July 5th, 2021.
3. *Polling*, where data automatically retrieved every sixty (60) seconds.

Lastly, as shown in Figure 8-15 below, the users shall also define how often the data handling and processing should happen (e.g., immediately, on an hourly, daily or weekly basis) and define the desired error handling strategy that should be applied (e.g., no action or undertake a specific number of retries every x seconds).

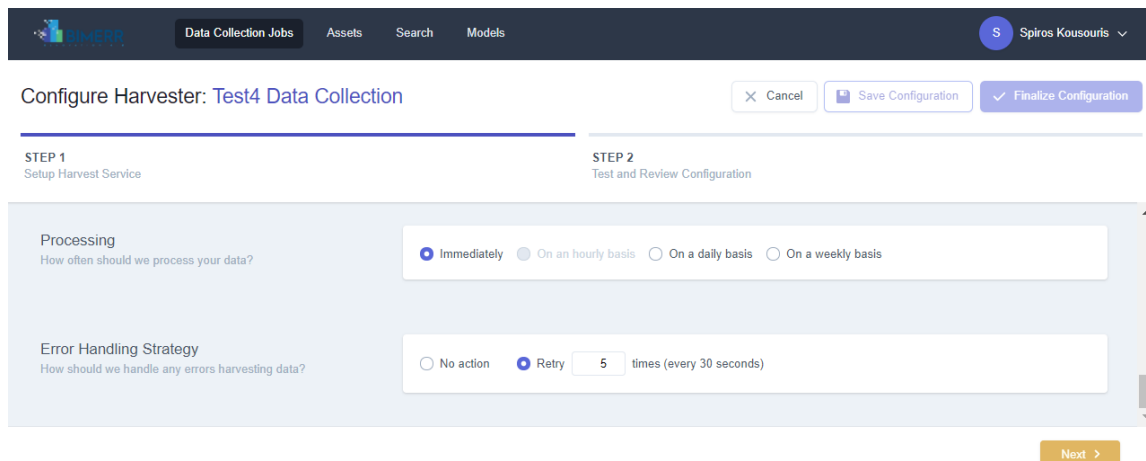


Figure 8-15: Data collection through external APIs – Processing & Error Handling Strategy

By clicking next, the users move on to the second step of the API data collection configuration, as shown in Figure 8-16, where they can test and review their configuration prior to finalizing it.

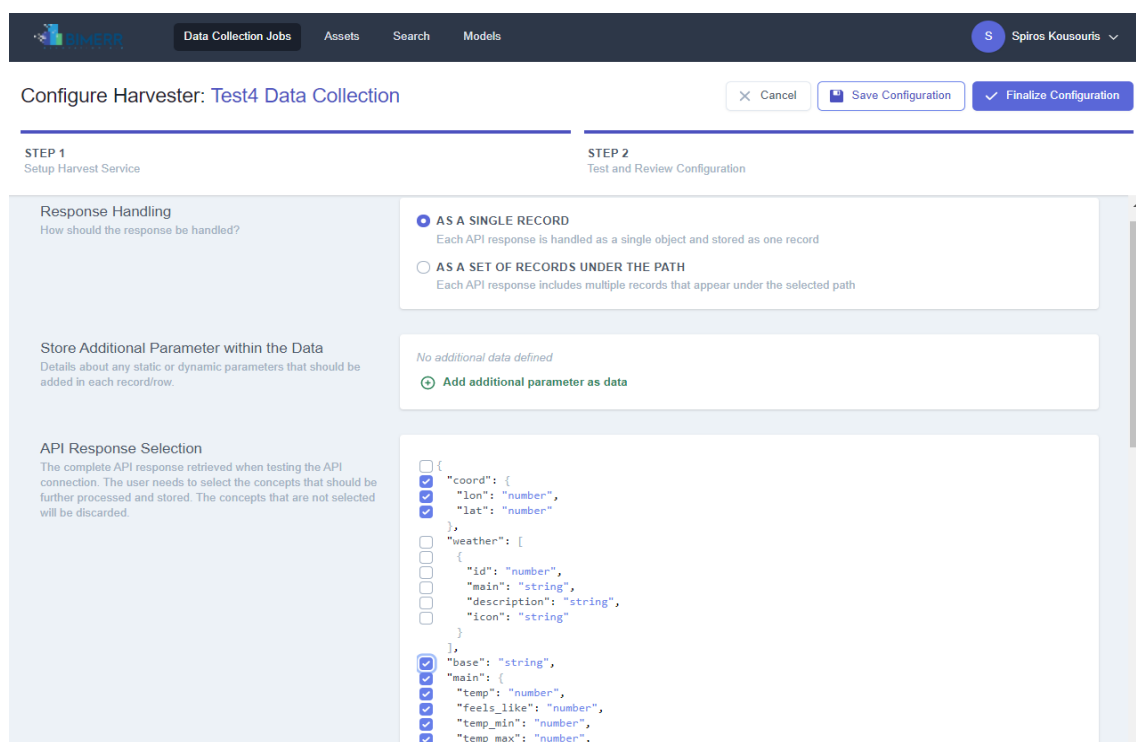


Figure 8-16: Data collection through external APIs - Response handling, storage of additional parameters and API response selection

In this step, users can select how the API response will be handled, i.e., each API response handled as a single record or multiple records are included in each response and need to be placed under a given path. Moreover, if required, the users can add additional parameters (static or dynamic) that shall be added in each record/row of the data.

While the complete API response is displayed when testing the API connection, the users are enabled to select the part/concepts of the API response that should be stored and further processed (see Figure 8-16). Any remaining data are to be discarded from the BIF.

At the bottom of the same screen, as shown in Figure 8-17, the users can view a summary of the API response they have selected along with the retrieved data that will be permanently stored. The users can save the configuration, in order to revisit it later and/or finalize it, in order to initiate the actual data collection execution according to the retrieval settings that have been set.

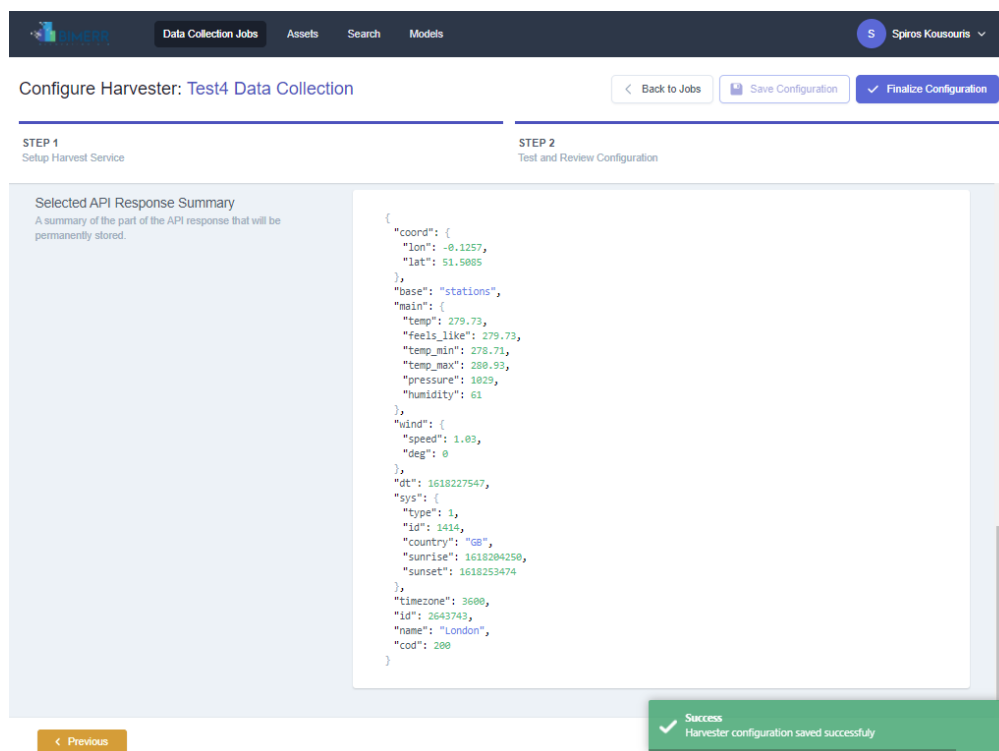


Figure 8 17: Data collection through external APIs – View and save sample of selected API response part

Upon finalizing the specific job, a pop-up window confirms that the Harvester configuration is completed Figure 8-18 and the data collection activities are updated. The

users can now select to proceed with the Mapping configuration or to see their created jobs.

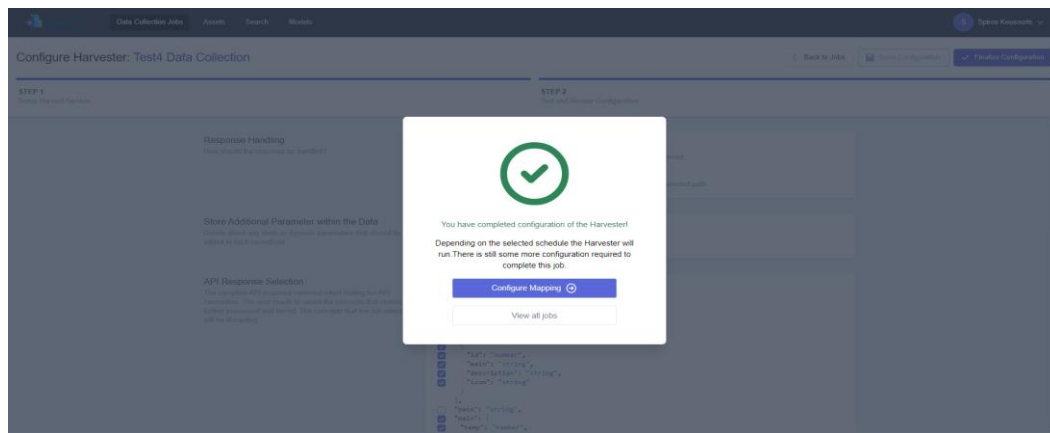


Figure 8-18: Data collection through external APIs - Notification for completing Harvester configuration

8.4 CONFIGURE A DATA COLLECTION JOB THROUGH THE BIF APIs

In case the users selects to upload their data to BIF (see Figure 8-3) via the Platform's API (i.e. via the internal API offered by BIF), they must first configure the Harvester as in the previous data loading methods.

In this first step, as shown in Figure 8-19, the users shall specify the type of data to be uploaded (e.g., Text only or Text and Binary), the preferred processing schedule (i.e., immediately, on an hourly, daily or on a weekly basis), and upload a sample of their data to be utilised in the second step of the Harvester configuration.

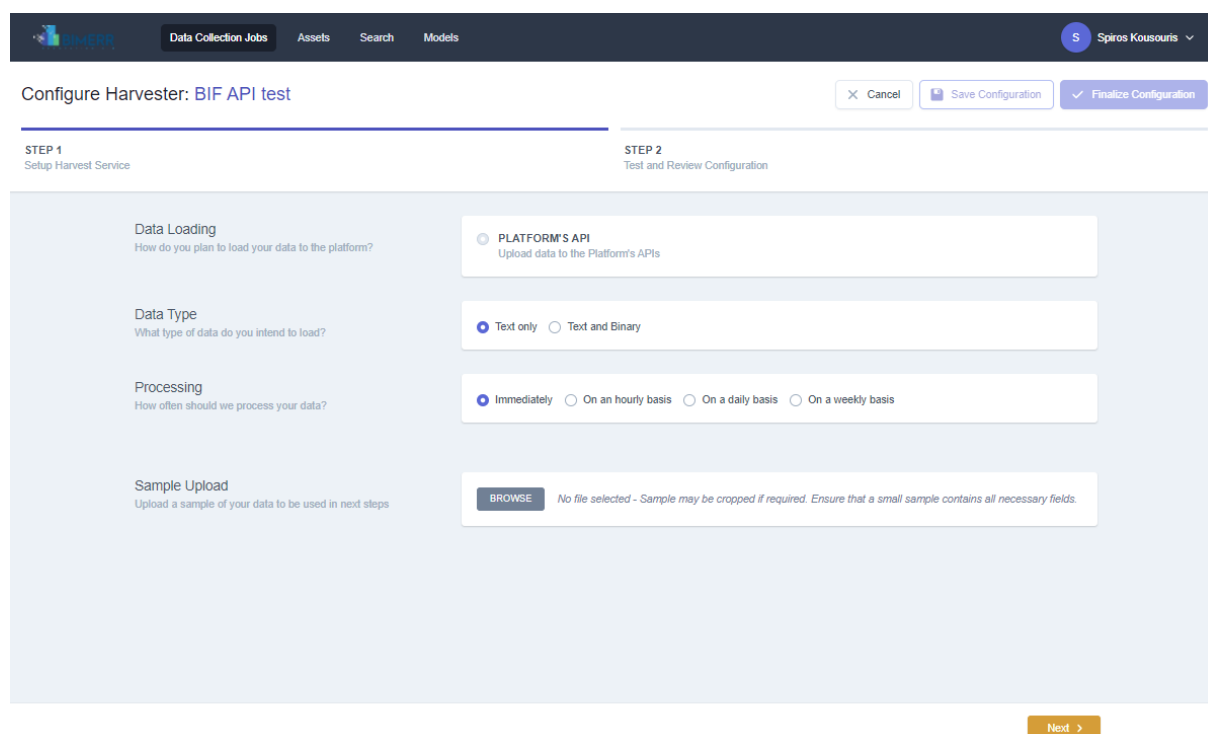



Figure 8-19: Data collection through BIF's API - Harvester configuration (text only)

Once the users upload their data sample, as shown in Figure 8-20 below, they can view the API method (i.e., POST) and the full URL.

Instructions are also provided to the users on how to use the provided POST endpoint. Additionally, the users can view the details of the data sample that they uploaded.


Data Collection Jobs
Assets
Search
Models
S Spiros Kousouris

Configure Harvester: **Test5 Data Clection BIF 'sAPI**
Cancel
Save Configuration
Finalize Configuration

STEP 1
Setup Harvest Service

STEP 2
Test and Review Configuration

Method & URL
The API method and the full URL

Instructions
How to use the POST endpoint

Data Sample
The details of the data sample that was uploaded

POST
https://bimerr.s5labs.eu/api/upload/968c4016-ba27-4047-9bd6-e721cc272f44

Authentication
In order to use the generated API, you should be authenticated. To do so, you need to:
1. Use an already generated access token with [upload](#) scope or [generate a new one](#). This token will be used to authenticate your requests.
2. Add the created access token into an **X-API-TOKEN** header in your request.
3. Insert the data you wish to upload to the API, into the body of your request, in JSON format.

```
[
  {
    "Building ID": "Building_1",
    "Building Description": "A building which contains 2 spaces, both residencies and offices.",
    "Building Type": "Mixed",
    "Space ID": "S0",
    "Space Description": "Residence for 1 occupant",
    "Space Type": "ResidentialOwn",
    "Space MaxNumOccupants": 1,
    "OccupantID": "O1",
    "Occupant Name": "OccupantO1S0",
    "BehaviorID": "B_Thermal",
    "Behavior Description": "When temperature is outside the thermal comfort range set to 22.5 deg.C",
    "Need_ID": "5",
    "Need Type": "Thermal",
    "maximum temperature": 95,
    "Meeting ID": "M1",
    "Meeting Duration": "120",
    "Meeting Start": "29/04/2020 18:00",
    "Meeting End": "29/04/2020 18:02"
  },
  {
    "Building ID": "Building_1"
  }
]
```

< Previous

Figure 8-20: Data collection through BIF's API – Instructions for data loading (text) and overview of data sample

In the event the users want to send Text & Binary data, they shall select the appropriate data type (see Figure 8-21) and upload their text data sample. As in the case of sending Text only, the Method & URL are provided along with instructions in order for the users to send the binary file through the generated API.

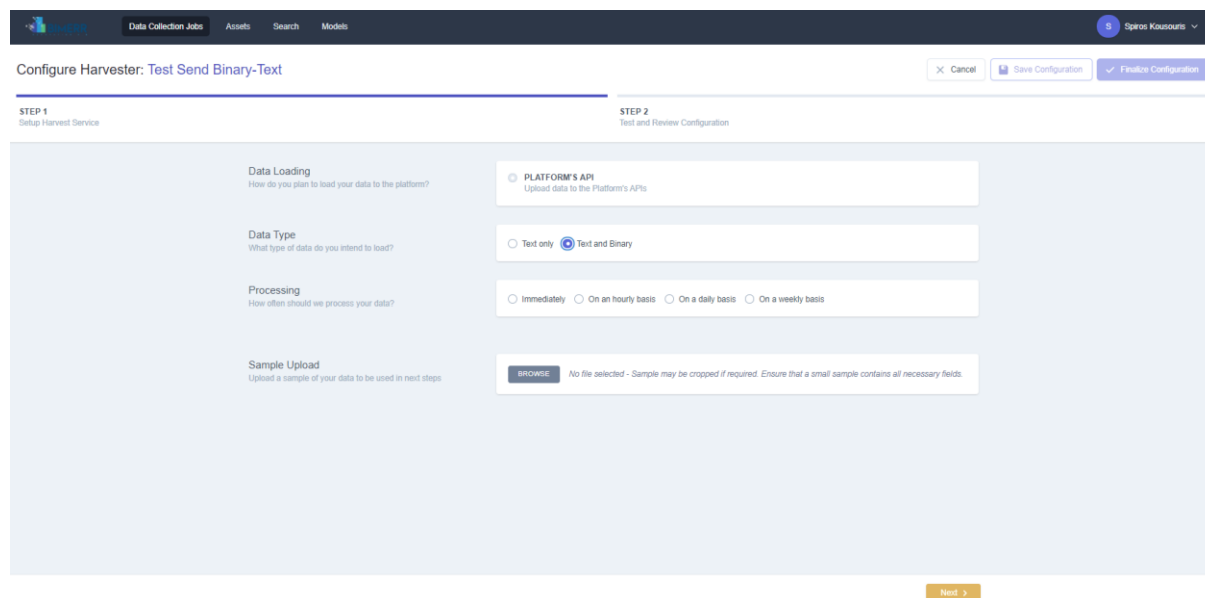


Figure 8-21: Data collection through BIF's API - Harvester configuration (Text and Binary)

Similar to the aforementioned data loading methods, the users can save the configuration (see Figure 8-22), in order to revisit it later and/or finalize it (see Figure 8-23), in order to initiate the actual data collection execution.

Configure Harvester: Test5 Data Clection BIF 'sAPI

STEP 1 Setup Harvest Service | STEP 2 Test and Review Configuration

Method & URL
The API method and the full URL

Instructions
How to use the POST endpoint

Data Sample
The details of the data sample that was uploaded

POST | <https://bimerr.s5labs.eu/api/upload/966c4016-ba27-4047-9bd6-e721cc272f44>

Authentication
In order to use the generated API, you should be authenticated. To do so, you need to:

1. Use an already generated access token with `upload` scope or [generate a new one](#). This token will be used to authenticate your requests.
2. Add the created access token into an `X-API-TOKEN` header in your request.
3. Insert the data you wish to upload to the API, into the body of your request, in JSON format.

```
{
  "Building ID": "Building_1",
  "Building Description": "A building which contains 2 spaces, both residences and offices.",
  "Building Type": "Mixed",
  "Space ID": "S0",
  "Space Description": "Residence for 1 occupant",
  "Space Type": "ResidentialOwn",
  "Space MaxNumOccupants": 1,
  "OccupantID": "O1",
  "Occupant Name": "OccupantO1S0",
  "BehaviorID": "B_Therm1",
  "Behavior Description": "When temperature is outside the thermal comfort range set to 22.5 deg.C",
  "Need_ID": "5",
  "Need Type": "Thermal",
  "maximum temperature": 95,
  "Meeting ID": "M1",
  "Meeting Duration": "120",
  "Meeting Start": "29/04/2020 18:00",
  "Meeting End": "29/04/2020 18:02"
},
{
  "Building ID": "Building_1",
```

Success
Harvester configuration saved successfully

< Previous

Figure 8-22: Data collection through BIF's API - Notification for successful saving of the data upload

Configure Harvester: Test5 Data Clection BIF 'sAPI

STEP 1 Setup Harvest Service | STEP 2 Test and Review Configuration

An error has occurred!
Request failed with status code 400

Method & URL
The API method and the full URL

Instructions
How to use the POST endpoint

Data Sample
The details of the data sample that was uploaded

8bd6-e721cc272f44

You have completed configuration of the Harvester!

Depending on the selected schedule the Harvester will run. There is still some more configuration required to complete this job.

Configure Mapping

View all jobs

< Previous

Figure 8-23: Data collection through BIF's API - Notification for completing Harvester configuration

8.5 PROVISION OF AN ASSET'S METADATA

When a data collection job is successfully completed, (following any of the aforementioned ways and as described in D4.5), the users can now view it through the Asset Menu. By clicking on the Asset Tab, see Figure 8-24 and Figure 8-25, all their data assets successfully collected will be visible. The new uploaded dataset (e.g., Test binary-text) will be initially marked as “Incomplete”.

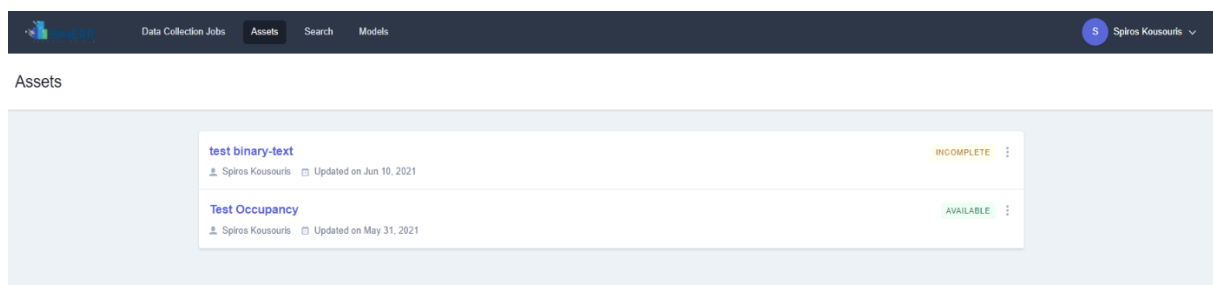


Figure 8-24: View of user's uploaded Asset's

By selecting the data asset of interest, the users can view an overview of their data asset and are also prompted to edit their asset (see Figure 8-25) if it is marked as incomplete.

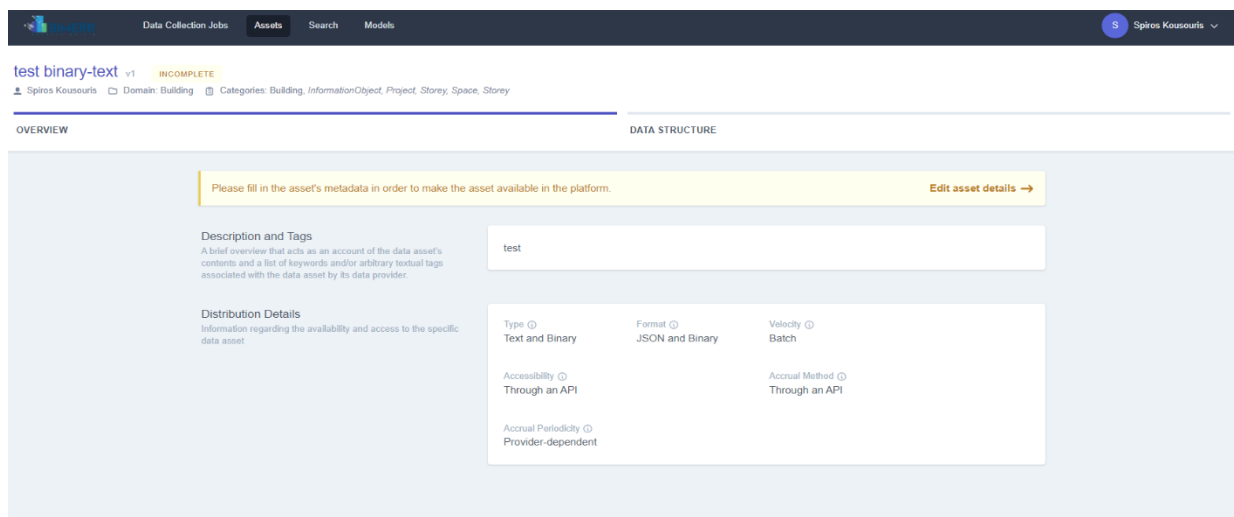
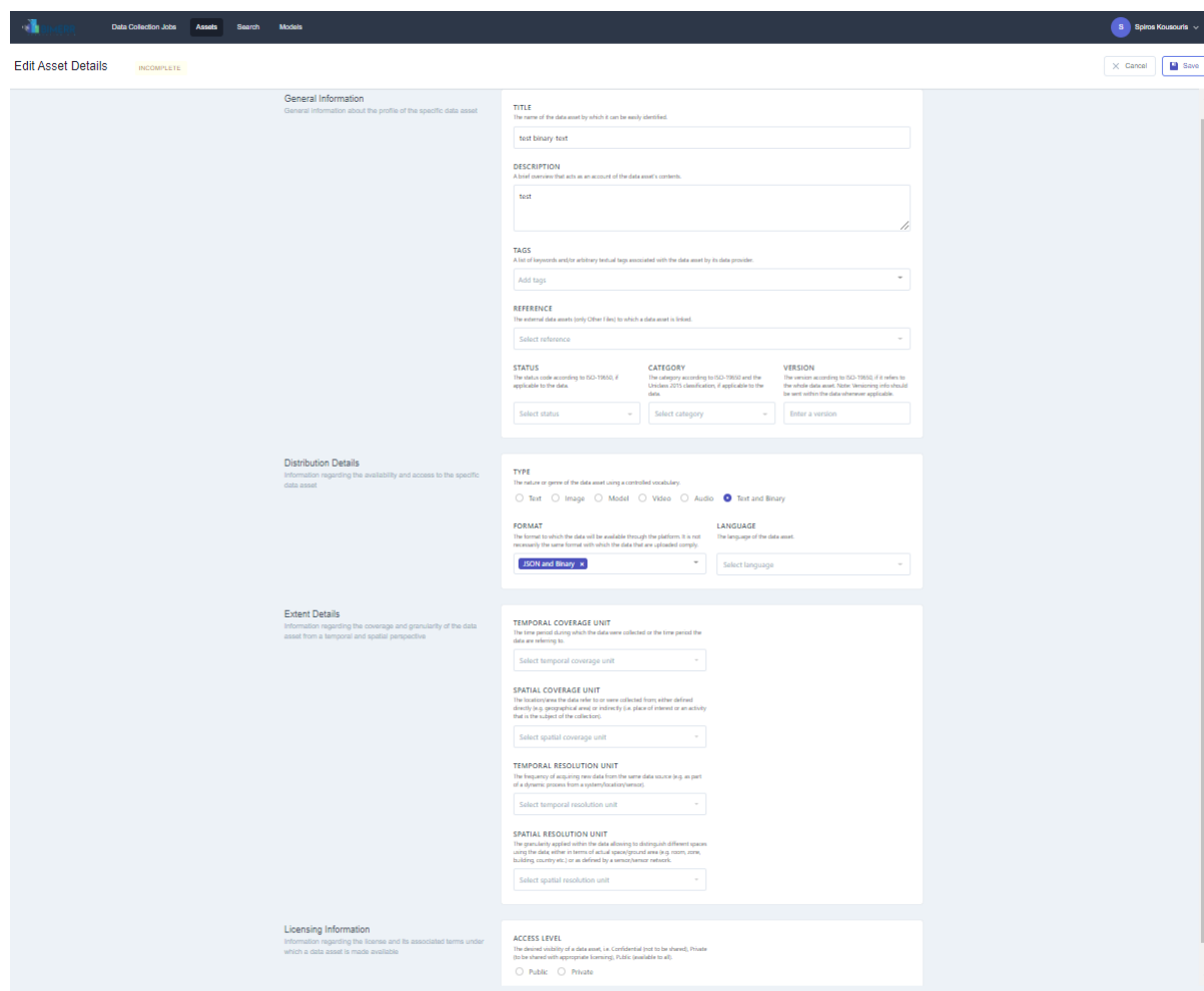


Figure 8-25: Overview of data asset

By clicking on the “Edit Asset details” button (Figure 8-25), the users are now prompted, as shown in Figure 8-26 below, to edit the assets basic metadata (title, description, tags and reference to external data assets), as well as define some additional metadata (status,

category and version) required in ISO19650 (if applicable). The users shall also provide further information on their data asset distribution (type, format and language), its coverage and granularity extent (temporal and spatial coverage and resolution units), as well as define its licensing information (access level, copyright owner and license). For more information of the configuration of the access policies in an Asset's metadata, the readers are referred to D4.9, section 5.1.



Edit Asset Details INCOMPLETE

General Information
General information about the profile of the specific data asset

TITLE
The name of the data asset by which it can be easily identified.
text binary text

DESCRIPTION
A brief overview that acts as an account of the data asset's contents.
text

TAGS
A list of keywords and/or arbitrary textual tags associated with the data asset by its data provider.
Add tags

REFERENCE
The external data assets (only Other if flag) to which a data asset is linked.
Select reference

STATUS
The status code according to ISO 19650, if applicable to the data.
Select status

CATEGORY
The category according to ISO 19650 and the Uniclass 2015 classification, if applicable to the data.
Select category

VERSION
The version according to ISO 19650, if it refers to the whole data asset. Note: Versioning info should be used when the data is not applicable.
Enter a version

Distribution Details
Information regarding the availability and access to the specific data asset

TYPE
The nature or genre of the data asset using a controlled vocabulary.
☐ Text ☐ Image ☐ Model ☐ Video ☐ Audio ☒ Text and Binary

FORMAT
The format to which the data will be available through the platform. It is not necessary the same format with which the data that are uploaded comply.
JSON and Binary

LANGUAGE
The language of the data asset.
Select language

Extent Details
Information regarding the coverage and granularity of the data asset from a temporal and spatial perspective

TEMPORAL COVERAGE UNIT
The time period during which the data were collected or the time period the data are referring to.
Select temporal coverage unit

SPATIAL COVERAGE UNIT
The boundaries the data refer to or some defined from either defined directly (e.g. geographical area) or indirectly (i.e. place of interest or an activity that is the subject of the collection).
Select spatial coverage unit

TEMPORAL RESOLUTION UNIT
The frequency of acquiring new data from the same data source (e.g. as part of a dynamic process from a system/technology/sensor).
Select temporal resolution unit

SPATIAL RESOLUTION UNIT
The granularity applied within the data allowing to distinguish different spaces using the data, either in terms of actual space/ground area (e.g. rooms, zones, building, country etc.) or as defined by a sensor/network network.
Select spatial resolution unit

Licensing Information
Information regarding the license and its associated terms under which a data asset is made available

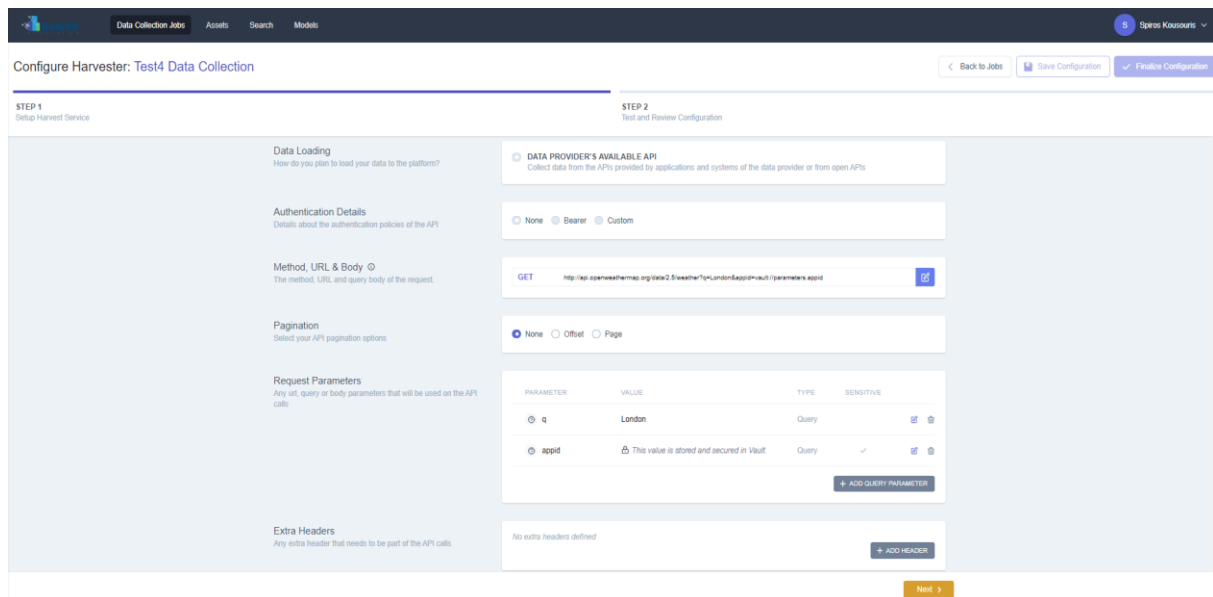
ACCESS LEVEL
The desired visibility of a data asset, i.e. Confidential (not to be shared), Private (data shared with appropriate licensing), Public (available to all).
☐ Public ☐ Private

Figure 8-26: Edit Asset Details

By providing such information and once their asset is marked as "Available" in the main Asset Menu (see Figure 8-24), the data asset will appear in the Search menu (as described in detail in D4.9).

8.6 UPDATE OF THE DATA COLLECTION JOB

As mentioned previously, a data collection configuration can be later revisited by the user; (see Figure 8-27). Most of the parameters are not anymore editable, while any sensitive parameters (e.g., API key) are not displayed, but there is an indication that they are secured in the BIMERR BIF Vault.



Configure Harvester: Test4 Data Collection

STEP 1: Setup Harvest Service

STEP 2: Test and Review Configuration

Data Loading
How do you plan to load your data to the platform?

Authentication Details
Details about the authentication policies of the API

Method, URL & Body
The method, URL, and query body of the request

Pagination
Select your API pagination options

Request Parameters
Any url, query or body parameters that will be used on the API calls

Extra Headers
Any extra header that needs to be part of the API calls

DATA PROVIDER'S AVAILABLE API
Collect data from the APIs provided by applications and systems of the data provider or from open APIs

None ☒ Bearer ☐ Custom

GET <http://api.openweathermap.org/data/2.5/forecast?lat=London&appid=aut/parameters/appid>

None ☒ Offset ☐ Page

PARAMETER	VALUE	TYPE	SENSITIVE
q	London	Query	<input type="checkbox"/>
appid	This value is stored and secured in Vault	Query	<input checked="" type="checkbox"/>

No extra headers defined

Next >

Figure 8-27: Updating the data collection configuration for external APIs – Part 1

The users can only edit the non-sensitive query parameters, as well as edit the retrieval and processing settings, as shown in Figure 8-28.

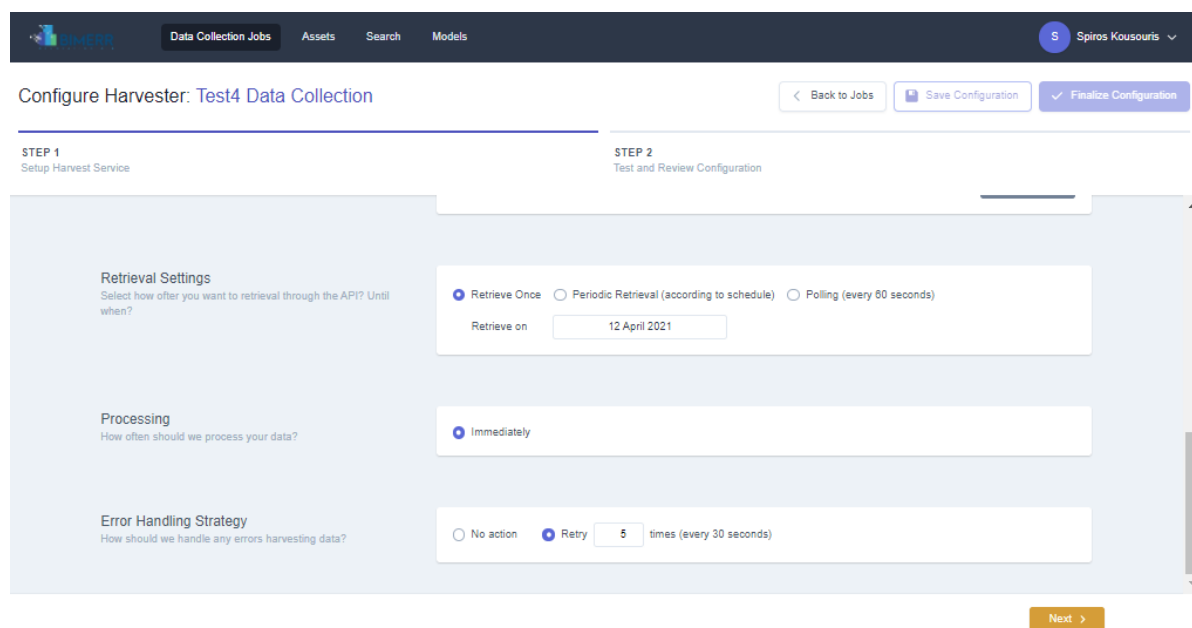


Figure 8-28: Updating the data collection configuration for external APIs – Part 2

For consistency with the already collected data, if the respective data collection job has been finalized and executed at least once, the users cannot perform any change on the part of the API response that is selected to be stored as viewed in Figure 8-27.

By clicking next, the users are directed to the second step, (see Figure 8-29) where upon saving their configuration the respective file(s) are uploaded, and a notification is displayed if the configuration has been successfully updated.

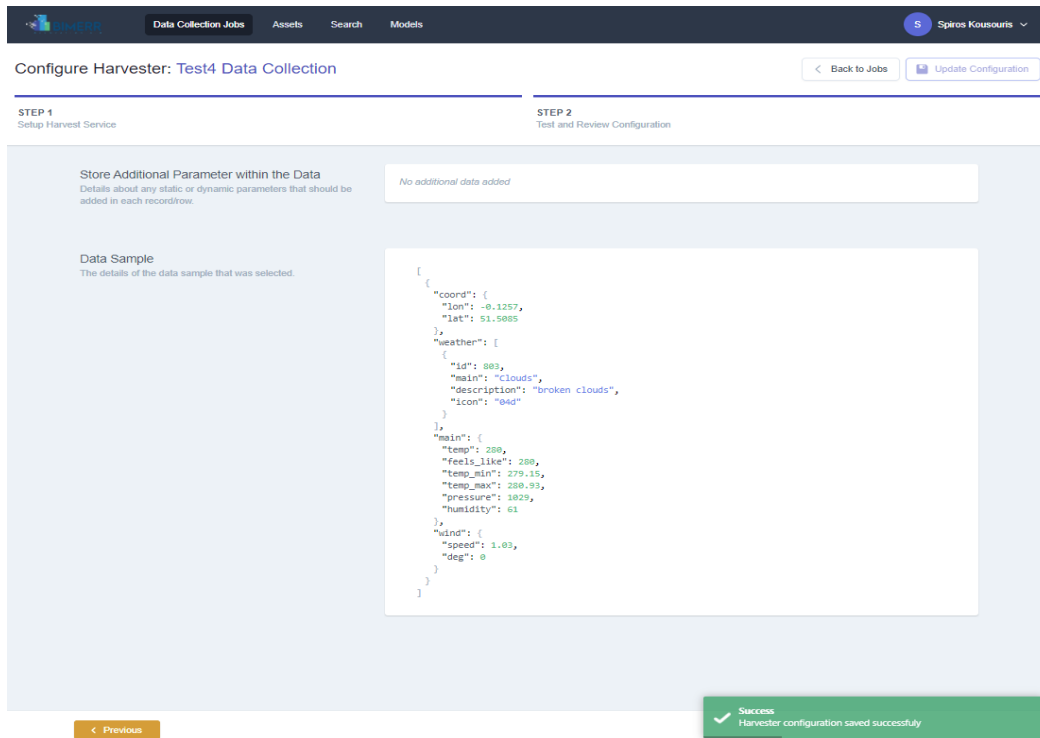


Figure 8-29: Updating the data collection configuration for external APIs – Part 3

Upon finalizing the specific job, a pop-up window confirms that the Harvester configuration is completed (Figure 8-30) and the data collection activities are updated. The users can now select to proceed with checking the Mapping configuration or to see their created jobs.

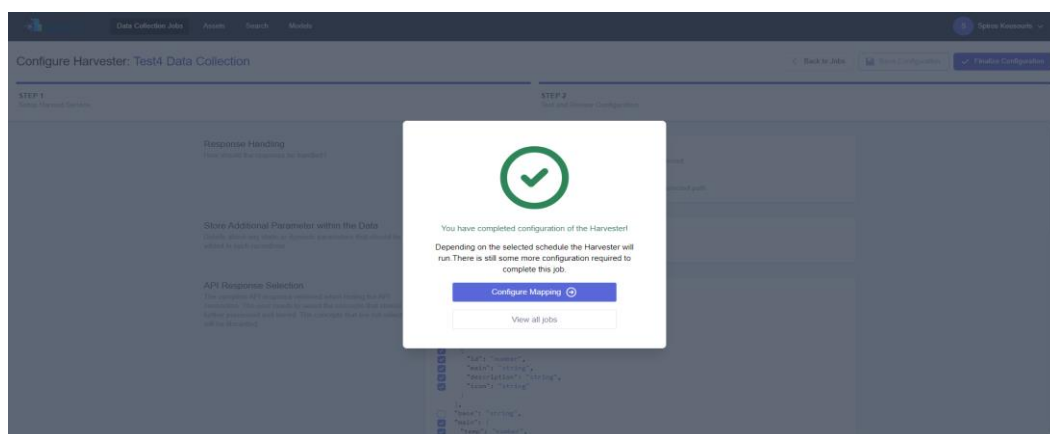


Figure 8-30: Data collection through external APIs - Notification for completing Harvester configuration

9. CONCLUSIONS

The BIMERR Information Collection and Enrichment Component, in conjunction with the BIMERR Semantic Modelling Component (that is documented in the BIMERR Deliverable D4.5 [6]), lie at the core of the BIMERR Interoperability Framework when it comes to building data collection, semantic mapping and reconciliation, harmonization and storage from the data providers' perspective.

As documented in this deliverable (D4.7), the core functionalities of the BIMERR Information Collection and Enrichment Component are delivered through its five subcomponents, namely the Data Ingester & Fetcher, the Data Handler, the Data Storage & Indexing, the Master Controller and the Knowledge Graph Generator, which have been developed as planned both in terms of back-end processing requirements and front-end user experience.

This final version of the BIMERR Information Collection and Enrichment Component has been built upon testing, evaluation and the feedback received: (a) by the different BIMERR components during the BIF integration activities carried out in WP4; and (b) by the BIMERR applications developed in WP5, WP6 and WP7, during the preliminary integration activities carried out in WP8.

ANNEX I: BIBLIOGRAPHY

- [1] BIMERR (2018) Description of Action (DoA)
- [2] BIMERR (2019a) D3.1 - Stakeholder requirements for the BIMERR system
- [3] BIMERR (2020b) D3.6 - BIMERR system architecture 2nd version
- [4] BIMERR (2020b) D4.2 - BIMERR Ontology & Data Model 1
- [5] BIMERR (2021a) D4.3 - BIMERR Ontology & Data Model 2
- [6] BIMERR (2021a) D4.5 - BIMERR Building Semantic Modelling tool 2
- [7] BIMERR (2020b) D4.6 - BIMERR Information Collection & Enrichment Tool 1
- [8] BIMERR (2020a) D4.8 -Integrated BIMERR Interoperability Framework 1
- [9] BIMERR (2021a) D4.9 - Integrated BIMERR Interoperability Framework 2
- [10] ISO 19650-1:2018 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 1: Concepts and principles, UK: BSI Standards Publication