Project Acronym:	**BIMERR**
Project Full Title:	**BIM-based holistic tools for Energy-driven Renovation of existing Residences**
Grant Agreement:	**820621**
Project Duration:	**42 months**

DELIVERABLE D4.4

BIMERR Building Semantic Modelling tool 1

Deliverable Status:	**Final**
File Name:	**BIMERR_D4.4-v1.00**
Due Date:	**30/04/2020 (M16)**
Submission Date:	**22/05/2020 (M17)**
Task Leader:	**Suite5 (T4.3)**

Dissemination level	
Public	x

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

~~Confidential, only for members of the Consortium (including the Commission Services)~~	

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
UCL	University College London	United Kingdom
NT	Novitech As	Slovakia
FER	Ferrovial Agroman S.A	Spain

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	Bikas	George	Suite5	gbikas@suite5.eu
2	Bountouni	Nefeli	Suite5	nefeli@suite5.eu
3	Chávez-Feria	Serge	UPM	schavez@delicias.dia.fi.upm.es
4	Constantinou	Giannis	Suite5	giannis@suite5.eu
5	Kousouris	Spiros	Suite5	spiros@suite5.eu
6	Tsitsanis	Tasos	Suite5	tasos@suite5.eu
7	Vafeiadis	George	UBITECH	gvafeiadis@ubitech.eu
8	Vergeti	Danai	UBITECH	vergetid@ubitech.eu

REVIEWERS LIST

List of Reviewers (in alphabetic order)				
#	**Surname**	**First Name**	**Beneficiary**	**Contact email**
1	Fenz	Stefan	XYLEM	fenz@xylem.tech
2	Poveda-Villalón	María	UPM	mpoveda@fi.upm.es

REVISION CONTROL

Version	Author	Date	Status
0.10	Suite5	27/03/2020	Draft ToC
0.20	Suite5	10/04/2020	Draft Sections 1, 2
0.30	UPM	20/04/2020	Draft Section 3
0.40	Suite5	22/04/2020	Draft Section 4
0.50	Suite5	27/04/2020	Draft Section 5 and 6
0.60	Suite5	30/04/2020	Revisions to Sections 2, 4, 5
0.70	Suite5	04/05/2020	Draft version to be circulated for Internal Quality Check
0.80	Suite5, All	19/05/2020	Updates to address the comments received during the Internal Quality Check
1.00	Suite5	22/05/2020	Final draft for submission to the EC

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

Acronym	Meaning
API	Application Programming Interface
BIF	BIMERR Interoperability Framework
BIMERR	BIM-based holistic tools for Energy-driven Renovation of existing Residences
MM	Model Mapper
MLM	Model Lifecycle Manager
OMF	Ontology Manager Framework
BO2DM	BIMERR Ontology to Data Model

EXECUTIVE SUMMARY

The purpose of the present documentation for the BIMERR Deliverable D4.4 "BIMERR Building Semantic Modelling tool 1" is to accompany the BIMERR Semantic Modelling Component and formalise the conclusion of the first iteration of the development activities in T4.3 "Building Semantic Modelling Tools Creation". Overall, the BIMERR Semantic Modelling Component is an integral part of the BIMERR Interoperability Framework (BIF) that aims to ensure the semantic mapping and reconciliation of the building-related data that are to be collected from various sources, ranging from the BIMERR applications to legacy systems, while effectively addressing a number of semantic and syntactic interoperability challenges at data and model levels.

In alignment with the BIMERR architecture, the BIMERR Semantic Modelling Component relies on the complementary use of ontologies and data models taking the best of breed of both data modelling paradigms in its Ontology Manager Framework (OMF), its Model Mapper (MM) and its Model Lifecycle Manager (MLM). The three subcomponents that constitute the BIMERR Semantic Modelling Component build on over 15 state-of-the art technologies to deliver the intended functionalities for: (a) the users, i.e. BIMERR applications owners/developers that act as building data providers to the BIF, and (b) the data model and ontology managers that are responsible for managing the creation, evolution and alignment between the BIMERR data models and ontologies.

The present documentation of the BIMERR Semantic Modelling Component (along with its subcomponents) is oriented towards the functionalities it broadly delivers, the technology stacks it builds upon, the APIs it exposes, the installation instructions and usage walkthroughs it offers to its users. As it is just the initial release of the BIMERR Semantic Modelling Component, it does not fully implement all the envisaged functionalities, depends on certain assumptions, imposes a set of restrictions, and requires a more effective integration with the different components within the BIF (that are still under development) in due time.

The final iteration of the BIMERR Semantic Modelling Component, that is anticipated to be released on M30 of the BIMERR project implementation, shall focus on enhancing the end-to-end user experience based on the feedback acquired during the BIF integration, the BIMERR

applications development and the experimentation in the pre-validation sites, while introducing a set of already planned extensions and new functionalities.

1. INTRODUCTION

1.1 SCOPE AND OBJECTIVES

The present deliverable D4.4, entitled "BIMERR Building Semantic Modelling tool 1" constitutes a report of the activities undertaken within the context of Task T4.3 "Building Semantic Modelling Tools Creation" of WP4 "BIMERR Interoperability Framework", towards the delivery of the first version of the Semantic Modelling Tools component of the BIMERR Interoperability Framework (BIF). The BIMERR Semantic Modelling component is practically responsible for the definition, application and maintenance of the BIMERR ontology and data model and their proper synchronization, towards ensuring semantic consistency and coherency among the building-related data exchanges within BIMERR and with any other external systems.

The main objective of D4.4 is to provide a comprehensive overview and a documentation of the first stable release of the Building Semantic Modelling tool. Since this deliverable is of type "Demonstrator", it provides the documentation of the actual software that has been developed (for the initial release) and delivered in accordance with the BIMERR requirements and architecture. In more detail, D4.4 provides an overview of the functionalities of the Building Semantic Modelling component, along with its architecture and the plan for its integration in the BIF. Each of the three subcomponents that constitute the Building Semantic Modelling component, namely the Ontology Manager Framework (OMF), the Model Mapper (MM) and the Model Lifecycle Manager (MLM), 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 during the first 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 (whenever available) to explain in detail each subcomponent's intended use.
- Identifying the accompanying licensing of each subcomponent.

In accordance with the BIMERR DoA [1], the Building Semantic Modelling component will be delivered in two releases, in M16 and M30 of the project implementation. As anticipated, the final release of the Building Semantic Modelling component, shall be built on the outcomes of this deliverable and will contain all the planned functionalities, as well as refinements and enhancements based on the updated outcomes of the design, specification and integration activities performed in WP4, while also taking into consideration the feedback that will be continuously collected during the pre-validation activities of WP8.

1.2 RELATION TO OTHER TASKS/DELIVERABLES

The BIMERR Deliverable D4.4 documents the activities performed in Task T4.3 "Building Semantic Modelling Tools Creation" and its main scope is to report the initial stable version of the Building Semantic Modelling Component. Towards this direction, for the design and implementation of the components described in this deliverable, but also for the creation of this deliverable, the current document receives input from the following deliverables of the BIMERR project:

- 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 skeleton for the BIMERR framework.
- D3.5 "BIMERR system architecture 1st version" [4], where the first version of the BIMERR architecture is provided, defining the BIMERR tools and components along with their functionalities, as well as the specification for information exchange among them.
- D4.2 "BIMERR Ontology & Data Model" [5], where the initial 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. The resulting ontology and data model are utilised by the Semantic Modelling component in order to effectively link information coming from any data source to the BIMERR Interoperability Framework.

The outcome of the activities performed in D4.4 will be also used as input in the following tasks and work packages:

- T4.4 "Building Information Collection and Enrichment Tools Creation" [6], where the Semantic Modelling tool will be utilised for semantic mapping of the data collected in the Building Information Collection and Enrichment component.
- T4.6 "Building Information Query Builder Creation", where the Semantic Modelling tool will provide the necessary "data model"-related input for the development of the Building Information Query Builder component ensuring consistency between the interfaces among the respective components.

In addition, D4.4 will provide significant input and a better understanding of the semantic interoperability repercussions 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". Such applications are also expected to provide feedback and lessons learnt from the real-life application of the Building Semantic Modelling Component towards the delivery of its final release. Finally, D4.4 and the Building Semantic Modelling Component is naturally part of the system level software integration and pre-validation activities performed in WP8 and thereafter in the validation and evaluation activities of WP9.

1.3 STRUCTURE OF THE DOCUMENT

In order to address all the aspects relevant to the scope of T4.3, 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 BIMERR Building Semantic Modelling Tool, its architecture and the plan for its integration in the Building Interoperability Framework (BIF).
- Sections 3–5 provide a comprehensive documentation of the different subcomponents forming the Building Semantic Modelling component, i.e. the Ontology Manager Framework, the Model Mapper and the Model Lifecycle Manager, respectively.
- In Section 6, conclusions are provided along with the release plan for the final iteration of the BIMERR Semantic Modelling component.

2. BIMERR BUILDING SEMANTIC MODELLING COMPONENT

2.1 OVERVIEW

The BIMERR Building Semantic Modelling component constitutes one of the four core components of the BIMERR Interoperability Framework (BIF) and it is instrumental for addressing the semantic interoperability challenges pertaining to the building-relevant data that derive from and are exchanged between the BIMERR applications and external systems. Such a component is responsible for understanding the semantics of the data that are to be handled by the BIF, linking/mapping them with the BIMERR data models and ontologies and effectively handling their lifecycle and alignment.

The Building Semantic Modelling component is composed by three main subcomponents (as depicted in Figure 2-1) which are briefly described in the following lines and individually presented in the Sections 3, 4 and 5.

- The ***Model Mapper*** (MM) aims to ensure semantic consistency between the building data that are extracted from legacy systems or from the BIMERR applications, and the BIMERR data model. Such a consistency is ensured through a semi-automated mapping prediction and configuration process that takes place with the help of the user (e.g. BIMERR application developer). In the Model Mapper, the users can review the automated mapping predictions (that are accompanied by a confidence level scale), and set the applicable mapping and transformation rules, as well as propose new concepts to be added to the BIMERR data model that satisfy the scope of their data.
- The ***Model Lifecycle Manager*** (MLM) is responsible for providing a thorough overview of the BIMERR data models and managing their evolution, from their initial creation and storage to their eventual evolution based on a set of predefined evolution rules and the moderation of the data model managers. Any evolution event is communicated to the Ontology Manager Framework to ensure that the BIMERR data models and ontologies are aligned.
- The ***Ontology Manager Framework*** (OMF) aims to support the various ontology management activities, including the documentation, evaluation, versioning and publishing of the BIMERR ontology while ensuring that both the BIMERR data model and

ontology are updated according to the latest state-of-the-art standards and enriched with further concepts required from the various applications.

The first version of the BIMERR Semantic Modelling component is deployed at: https://bimerr.s5labs.eu/

2.2 ARCHITECTURE

In alignment with the BIMERR deliverable D3.5 [4], Figure 2-1 illustrates the BIMERR Semantic Modelling component's architecture including its main subcomponents, information flows and interactions among them. As already mentioned, the Semantic Modelling component is composed by three core subcomponents, namely the Model Mapper, the Model Lifecycle Manager and the Ontology Manager Framework; each one designed with a distinct role, a distinct scope and a distinct set of core functionalities.

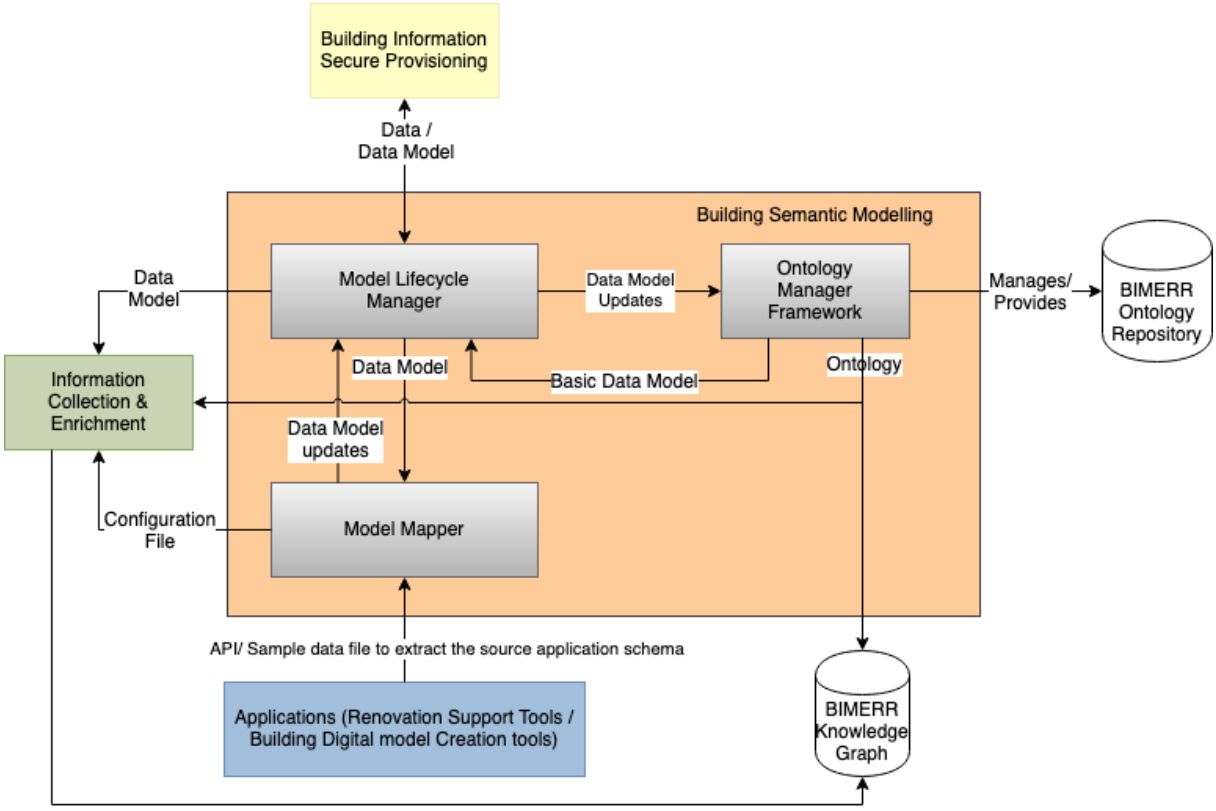

Figure 2-1: Architecture of the BIMERR Semantic Modelling component

Once the user has initiated a Data Collection job in the BIF Information Collection & Enrichment component, the Model Mapper is involved and undertakes the semi-automatic mapping of the source data that are collected and shall be exchanged through the BIF, to the underlying BIMERR data model. The Model Mapper derives the underlying schema from the available sample, predicts the mappings of the external data concepts to the BIMERR concepts, while enabling users to complement the mapping arrangements and add custom transformation rules. Upon a successfully completed mapping configuration, the Model Mapper populates the "Configuration File" with the data model mapping information which are made available to the Building Information Collection component.

The Model Lifecycle Manager's role is to import the basic data models it has received from the Ontology Manager Framework, to enrich the data model information (in accordance with the data model information described in the BIMERR Deliverable D4.2 [5]), to expose the BIMERR data models to any BIF component that requires them, as well as to oversee the evolution of the BIMERR data model. The Model Lifecycle Manager also receives requests for data model

updates from the Model Mapper and effectively manages them with the help of the BIMERR data model administrator. In addition, this component communicates with the OMF to ensure that the stored data models and ontologies are aligned; whenever a new concept is introduced, both the respective BIMERR data model and the ontology are updated from these two components, respectively.

The Ontology Manager Framework offers a documentation and publication tool that extracts the ontology metadata and generates the documentation from the relevant ontology metadata properties which is made available to the BIMERR Knowledge Graph and the BIMERR Ontology Repository. In addition, the Ontology Manager Framework provides a JSON serialisation of the ontology to the Model Lifecycle Manager and, in turn, receives the updates from the BIMERR data models that need to be incorporated in the BIMERR ontologies network.

2.3 INTEGRATION PLAN IN BIF

The Semantic Modelling Component that is documented in this deliverable is intended to be fully integrated in the BIMERR Interoperability Framework by its draft release that is expected on M18. To this end, the following integration plan is expected to be applied.

Table 2-1: Integration Plan of the Semantic Modelling Component in the initial BIF release

Integration Activity	M16	M17	M18	M19
Initial integration and testing of Building Information Collection & Enrichment – Building Information Semantic Modelling Components				
Further improvements on the Building Information Collection & Enrichment – Building Information Semantic Modelling Components based on actual data samples from the BIMERR applications and, if needed, for integration in overall BIF for M18				
Integration of additional BIMERR data models and ontologies				
Early integration with the BIMERR Building Information Query Builder and Building Information Secure Provisioning Component				

Integration Activity	M16	M17	M18	M19
Full integration of the first version of the integrated BIMERR Interoperability Framework				

3. ONTOLOGY MANAGER FRAMEWORK

3.1 OVERVIEW

The Ontology Manager Framework is a collaborative environment suite to support several ontology management activities, including the documentation, evaluation, versioning and publishing of the BIMERR ontology. The OMF documentation tool extracts the ontology metadata and generates the documentation from relevant ontology metadata properties in the form of HTML pages.

The evaluation module consists of: a) a common pitfall detector, and b) an ontology verification module that will check whether the proposed ontological requirements are satisfied by the ontology. The versioning tool is based on a Git-based system to store the history (current and previous versions) of an ontology. The publishing component helps to publish an ontology according to best practices (i.e., content negotiation, permanent URI) so that the ontology is available both in human oriented and machine-readable formats under a unique URI. Finally, it incorporates the ontology to data model converter, which transforms the OWL coded ontology into a JSON serialized data model.

The Ontology Manager Framework is composed by two main modules: 1) OnToology, a web application to support the ontology development process, and 2) BO2DM, a web service that performs the transformation process from an ontology to a data model. Each module is discussed in detail in the following subsections.

3.1.1 OnToology

OnToology is a web framework that supports many of the ontological activities carried out during the ontology development process. This tool tracks ontologies stored in GitHub repositories, producing a series of actions any time a new change is made on their implementation. These actions involve: 1) the generation of HTML documentation, 2) generation of an evaluation report that include pitfalls found on the implementation, 3) the publications of the ontologies and its related resources.

3.1.2 BO2DM

The BIMERR Ontology to Data Model converter is a web service, which given an OWL-coded ontology model performs a series of transformations to finally return a JSON serialized data model. This converter should receive an enriched version of the original ontology in order to extract all the metadata required by the data model. Some of this metadata fields can be directly extracted from the ontology implementation, however there are other fields that need extra annotations on each ontology element. A detailed list of the metadata terms required by the data model is shown in Table 3-1 in alignment with the BIMERR deliverable D4.2 [5]. This table also includes a short description of each term and if they need to be populated on the data model side or during the ontology implementation. Finally, Table 3-1 also indicates the equivalent annotation property for each metadata term used during the enrichment process, or the ontology resource used to extract the required metadata field.

Table 3-1: Metadata List

Metadata on Data Model	Equivalent Annotation Property or Ontology Resource used	Extracted From	Description	Populated By
Definition	rdfs: comment	Original Ontology	A brief overview that acts as an account of a concept's contents	Automatically extracted from the ontology
Type	Extracted from restrictions on: owl:ObjectProperty owl:DatatypeProperty	Original Ontology	The data type to which the concept's data comply, e.g. string, integer, boolean, datetime, etc	Automatically extracted from the ontology and applied to children nodes.
Related_terms	rdfs: labelskos:altLabel	Original Ontology	A set of related terms (e.g. synonyms) that can be alternatively used to represent the concept	Manually annotated by the data model admin in the Model Lifecycle Manager
standards	bm:isDefinedByStandard	Enriched Ontology	A list of standards in which the specific concept is modeled, along with their complementary information, i.e. the exact concept used in such a standard, its type (e.g. element, attribute) and its use (required/optional) that applies for attributes	Partly extracted from the ontology complemented by the data model admin in the Model Lifecycle Manager
Date_added	dc: created	Enriched Ontology	The date when the specific concept was added in the data model	Used for provenance and alignment between

Metadata on Data Model	Equivalent Annotation Property or Ontology Resource used	Extracted From	Description	Populated By
				the ontology and the respective data model
Date_deprecated	bm: deprecated	Enriched Ontology	The date when the specific concept became obsolete in the data model	Used for provenance and alignment between the ontology and the respective data model
Version	owl: versionInfo	Enriched Ontology	The version of the model when the concept was last modified	Used for provenance and alignment between the ontology and the respective data model
Children	Properties or attributes of type: owl: ObjectProperty owl: DatatypeProperty	Original Ontology	Grouped information for concepts that are conceptually classified under a concept	Automatically extracted from the ontology
Facet	---	---	Complementary information for a concept, essentially grouping the following metadata: cardinalityMax, ordered, sensitive, transformation, measurementType, measurementUnit, timezone	Partly extracted from the ontology and complemented by the data model admin in the Model Lifecycle Manager
cardinalityMax	Max cardinality extracted from: owl: FunctionalProperty owl:maxQualifiedCardinality	Original Ontology	The maximum number of expected appearances of a concept in the data	Automatically extracted from the ontology
Ordered	bm: ordered	Enriched Ontology	An indication of whether ordering is needed for multiple appearances of the same concept.	Automatically extracted from the ontology
Sensitive	bm: sensitive	Enriched Ontology	An indication whether the specific concept models personal or sensitive data	Automatically extracted from the ontology
transformation	bm: transformation	Enriched Ontology	Information for the transformation rules that are related to a specific applicable standard. It practically contains the function that is required for the transformation and the	Manually annotated by the data model admin in the Model Lifecycle Manager

Metadata on Data Model	Equivalent Annotation Property or Ontology Resource used	Extracted From	Description	Populated By
			parameters / concepts that are involved.	
measurementType	bm: measurementType	Enriched Ontology	An indication of the measurement type that is applicable to a concept, e.g. referring to distance, temperature, etc.	Manually annotated by the data model admin in the Model Lifecycle Manager
measurementUnit	bm: measurementUnit	Enriched Ontology	The baseline measurement unit for the specific concept and measurement type	Manually annotated by the data model admin in the Model Lifecycle Manager
timeZone	bm: timeZone	Enriched Ontology	The timezone to which the data refer by default.	Manually annotated by the data model admin in the Model Lifecycle Manager
codeList	bm: codeList	Enriched Ontology	A link to the code list that should be typically used for the data that refer to the specific concept	Manually annotated by the data model admin in the Model Lifecycle Manager
codeType	bm: codeType	Enriched Ontology	The type of code list that is applied for the specific concept	Manually annotated by the data model admin in the Model Lifecycle Manager

Table 3-2 shows complementary information regarding the namespaces and the corresponding prefixes.

Table 3-2: Ontology Namespaces and Prefixes

Prefix	Ontology namespace
owl	http://www.w3.org/2002/07/owl#
dc	http://purl.org/dc/elements/1.1/
bm	https://bimerr.iot.linkeddata.es/def/bimerr-metadata#
skos	http://www.w3.org/2004/02/skos/core#
rdfs	http://www.w3.org/2000/01/rdf-schema#

The complete transformation pipeline is depicted on Figure 3-1. The process starts with the enrichment step, whereby means of an ontology editor, users can import both the original ontology and the extra metadata fields (also in the form of an OWL-encoded ontology). Once the editor merges both ontologies the user can start the edition step, generating as a final result the enriched version of the ontology. Note that this enriched version is stored as a separated file that imports the original ontology. The original ontology is not edited in this process, but the changes are reflected in the enriched version due to the import mechanism. The BO2DM service takes as input this enriched ontology and triggers a series of conversion steps to finally generate the data model. Any changes produced to the original ontology will propagate to the enriched one that at the same time will materialize those changes in the data model through the BO2DM converter. The same alignment process is followed in the reverse direction, where any changes committed to the data model should be communicated to the converter, which will retro propagate those modifications to the enriched ontology. If the changes correspond to the metadata fields added during the enrichment process, those updates will be kept at the intermediate level. However, if the modifications affect some core element of the ontology, such as a concept or its attributes, the updates will also be performed over the original ontology in a manual way.

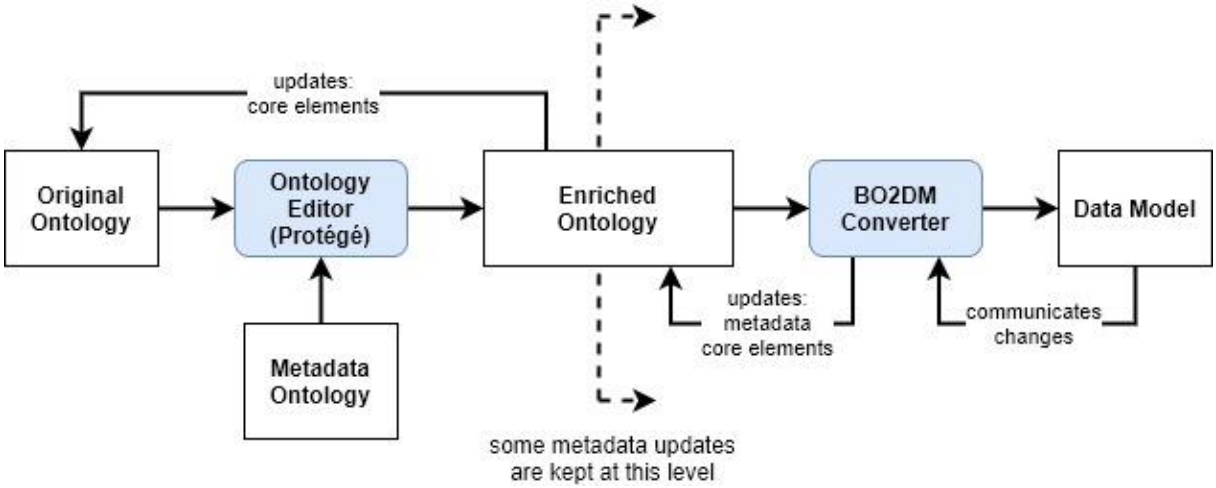

Figure 3-1: Ontology to Data Model Conversion Pipeline

The BO2DM converter's internal functionality is shown in Figure 3-2. The transformer initially takes the enriched ontology encoded in the OWL language to produce a JSON-LD serialization of the model. This format helps during the parsing procedure, re organizing the ontological

elements in a dictionary structure and making the value of each element and its associated metadata available through the use of "keys". Following this serialization, the Concepts Identification module identifies all the elements that correspond to the type `owl:Class`, and at the same time filters blank nodes. For each concept detected, all its associated metadata are identified. The second step involves the internal Children Identification stage, composed by the Attributes and Relations Identification submodules. The Attributes Identification component extracts all the datatype properties attached to each concept and incorporates them as children within the data model structure. A similar process is carried out in the Relations Identification sub module where object properties become children of a concept if this concept is on the source side of those properties (property domain side). In both sub modules, the metadata extraction procedure is performed for each of the elements identified. The converter performs this series of steps until all the ontology concepts are parsed and their respective children identified and populated, finally obtaining a first version of the data model. Before the converters output the data model, the Children Inheritance module evaluates the subclass relationships between concepts and copies the children set of the parent concept into the children structure of the subclass concept, generating the final data model.

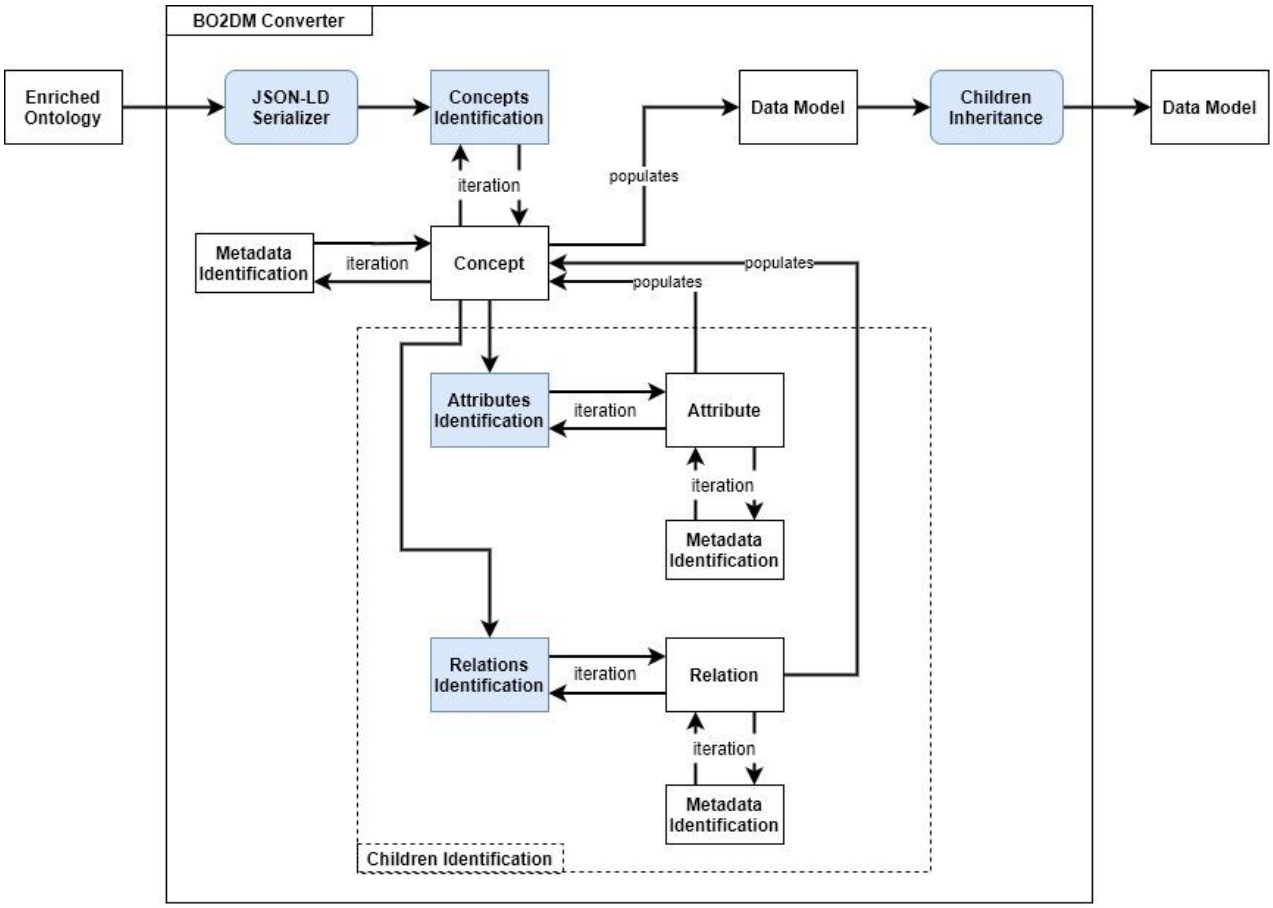

Figure 3-2: BIMERR Ontology to Data Model Converter

3.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The OnToology framework takes advantage of existing technologies in order to perform the versioning and hosting activities. Specifically, GitHub is used as the main storing infrastructure, hosting both the ontology implementation an all its complementary resources. Additionally, GitHub provides the means to handle multiples versions of the ontology, generated during the implementation sprints. Finally, it also provides the opportunities to manage issues and improvement requirements from users or ontology developers through the GitHub issues section.

OnToology is composed by four layers, 1) the presentation layer, which is a GUI web interface that allows the user the registering of their repositories, 2) the logic layer, which monitors changes in the ontology repository, and selects and activates the appropriate external and internal services that make up the framework suite, and 3) the persistence layer, implemented as an internal database containing a list of the users information, their tracked repositories, and their state in the processing pipeline.

The logic layer is further divided into the following sublayers:

- The Change Monitor layer is in charge of tracking the changes produced on the ontology repository. This process is done by a previous user confirmation while registering the GitHub repository in OnToology for the first time. As a result of this confirmation a web-hook link is established between GitHub and OnToology, allowing the notification to OnToology any time a new update is made to the ontology repository.
- The Orchestration layer selects which components of the framework suite will be activated when a change is detected by the tool. This selection is based on the configuration made by the user.
- The Integration layer executes the selected services, integrated by OnToology, and generates all the appropriate resources for each ontology model. The functionalities provided by this layer are the following:
  - The generation of documentation for each ontology. This information is provided in the form of HTML documentation by WIDOCO[1], which is a standalone application that extracts all the metadata properties from ontologies, generates equivalent serializations of the model according to W3C best practices, and provides with a provenance detailing the history of your vocabulary changes.
  - The evaluation of the ontology by means of OOPS![2], a web application that is able to detect 33 types of common semantic and syntactic errors in ontologies. Each pitfall is accompanied with a title, a description of the error, and an enumeration of all the ontology components affected by this error. Furthermore, the pitfalls are categorized into critical, important, or minor errors.

[1] https://github.com/dgarijo/Widoco/
[2] http://oops.linkeddata.es/

- The publication of the ontology under a permanent w3id URI. OnToology uses the w3id services to point to the location of the published ontology and their associated resources with content negotiation. Another alternative is to download a bundle with all the generated content to allow the user to publish the ontology on a server under its control.

The Ontology Manager Framework has been developed using Python 3.6[3] as the main programming language. Table 3-3 indicates all the libraries and software packages used in the OnToology framework, meanwhile Table 3-4 shows the same information but for the BO2DM component.

Table 3-3 List of Libraries used in OnToology

Name of the Library / Software	Version	License
Django	1.11.28	BSD 3-Clause
Pymongo	2.7.2	Apache 2.0
RDFLib	4.2.2	BSD 3-Clause
Requests	2.22.0	Apache 2.0
PyGithub	1.35	MIT
Selenium	2.52.0	Apache 2.0
Mongoengine	0.14.3	MIT
Widoco	1.4.13	Apache 2.0
OOPS!	---	GPL-3

Table 3-4 shows all the libraries and software packages used in the BO2DM component.

Table 3-4. List of Libraries used in BO2DM

Name of the Library / Software	Version	License
Flask	1.1.2	BSD 3-Clause
Flask Swagger UI	3.25.0	MIT
RDFLib	4.2.2	BSD 3-Clause
RDFLib JSON-LD	0.4.0	BSD 3-Clause

[3] https://www.python.org/

3.3 API DOCUMENTATION

OnToology provides all the appropriate documentation on its web site,[4] including detailed instructions of usage, and explanations of the internal functionality. On the other hand, the converter's documentation is provided in Swagger.[5] This web interface gives examples of the type of requests available and the parameters needed.

3.4 ASSUMPTIONS AND RESTRICTIONS

Each of the aforementioned OMF modules has its own set of limitations. These limitations are being handled as restrictions over the input data or operational modes in which the tools can be executed; or assumptions the tools made in order to simplify their functionality.

3.4.1 OnToology

One of the main limitations of OnToology is the missing support of repositories under a GitHub organization and private repositories. If the ontological OWL implementation is under an organization repository, it needs to be forked and then fed into OnToology with the new URL (your-username/the-repo-name).

Additionally, it only works with the master branch, which means that modifications result of development activities carried out in another branch need to be merged into the master branch first, before OnToology can track the changes.

The size of ontologies is another aspect to take into consideration when it comes to the evaluation activity. The evaluation of very large ontological models may not work due to OOPS!

[4] http://ontoology.linkeddata.es/
[5] https://converter.bimerr.iot.linkeddata.es/swagger/

Web service timing out. However, the HTML documentation and the diagrams generation will not be affected.

The publication in a given organization server involves a manual process to deploy the files from the GitHub repository to the organization web server which includes authentication. The publication under w3id permanent identifiers can't be set for paths, only the identifier of the ontology can be chosen.

The ontology conceptualization and enrichment are manual steps. The updates of the enriched ontologies are semi-automatic, depending on the changes reported by the data model.

3.4.2 BO2DM

The lack of clear transformation rules to convert the BIMERR ontology into a JSON serialized data model, makes it necessary to introduce some assumptions to generate a data structure as close as possible to the expected data model.

First, all the attributes attached to a specific concept will become children of this concept. This assumption corresponds with the fact that most of the children elements within the data model correspond to aspects of the concepts that can be described by datatype properties.

Second, all the relations between concepts will be incorporated as children within the data model. However, the target concepts associated to those properties will not be nested into a second level of children. They will be handled using pointers or references to concepts that will be on the first level of the data structure tree.

3.5 INSTALLATION INSTRUCTIONS

Each module of the OMF platform is deployed as a web service, meaning that they do not require the installation or downloading of any component in order to use them. OnToology requires an initial setup to register the ontological repositories, however, besides that no other step is required.

3.6 USAGE WALKTHROUGH

The initial step consists of registering the ontological repository to be tracked by OnToology through the web user interface, as shown in Figure 3-1. This step assumes that the repository hosting the OWL code is a user GitHub repository and does not correspond to an organization. In case the repository is from an organization, the user can fork the repository and provide the URL to OnToology. In case it is the first time the user accesses the service, OnToology will redirect the user to GitHub to confirm tracking permissions (See Figure 3-2). Once the authorization is confirmed, OnToology will start tracking any changes produced on the ontology.

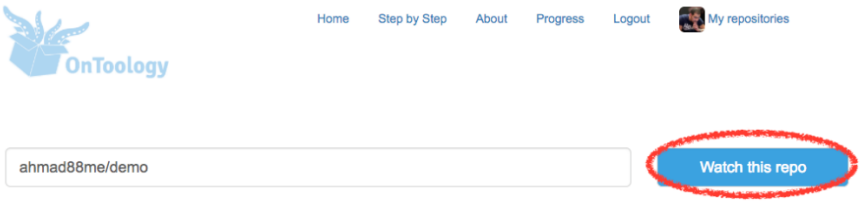

Figure 3-1. Ontology registering

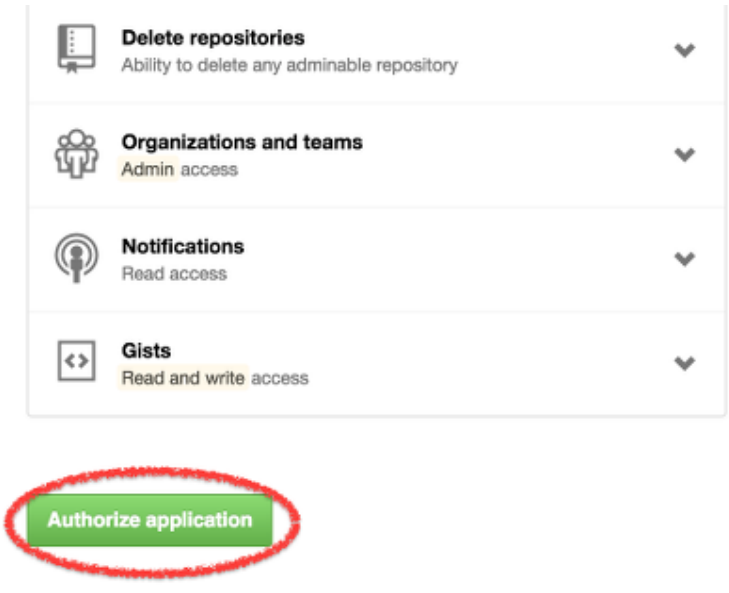

Figure 3-2. GitHub permissions confirmation

Every time ontology updates are pushed to the remote tracked repository, due to new requirements or issues detected by ontology users or domain experts, OnToology will automatically generate or re-regenerate the resources and will create a pull request, as shown in Figure 3-3, with those resources, i.e., the HTML documentation, diagrams, and the validation report.

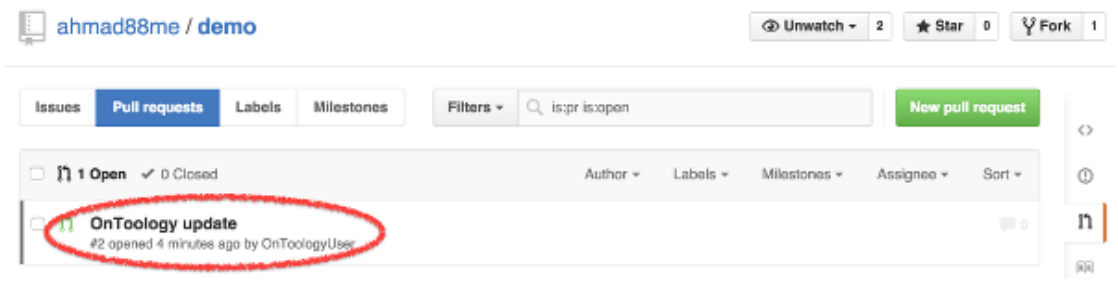

Figure 3-3. Pull request generation

Once the pull request appears on the ontology repository, the ontology manager has to review the changes and merge the appropriate pull request to accept the generated resources, as depicted in Figure 3-4. By confirming the merge, a new folder called OnToology with all the resources will be located inside the ontology repository, as shown in Figure 3-5.

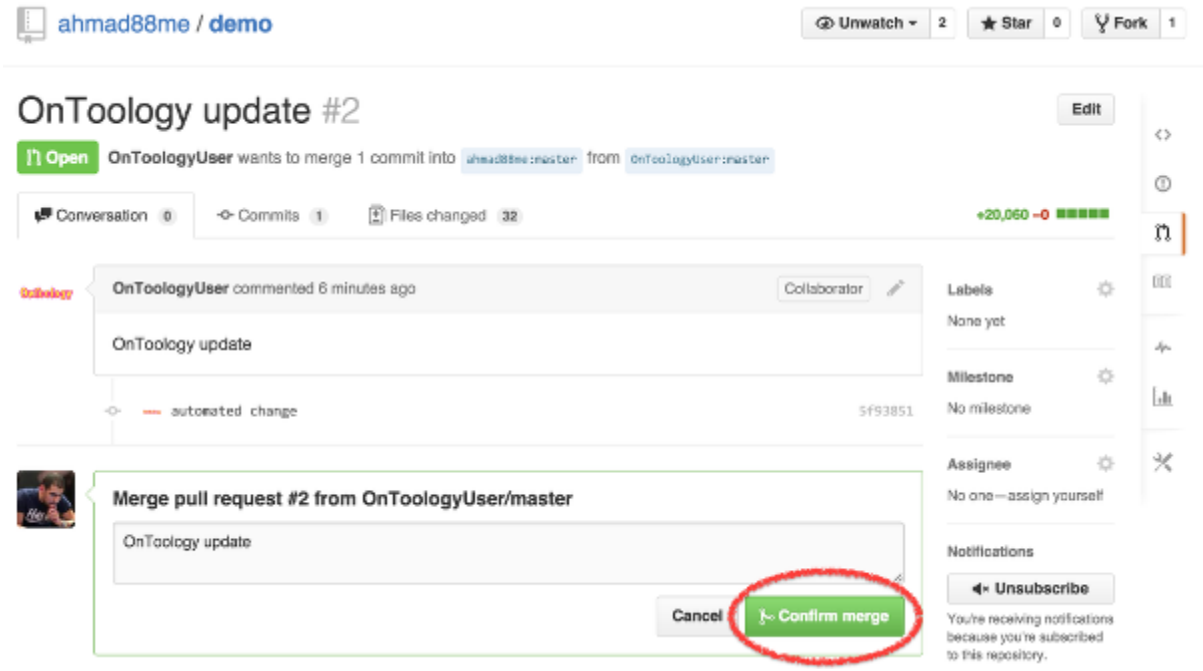

Figure 3-4. Pull request merge confirmation

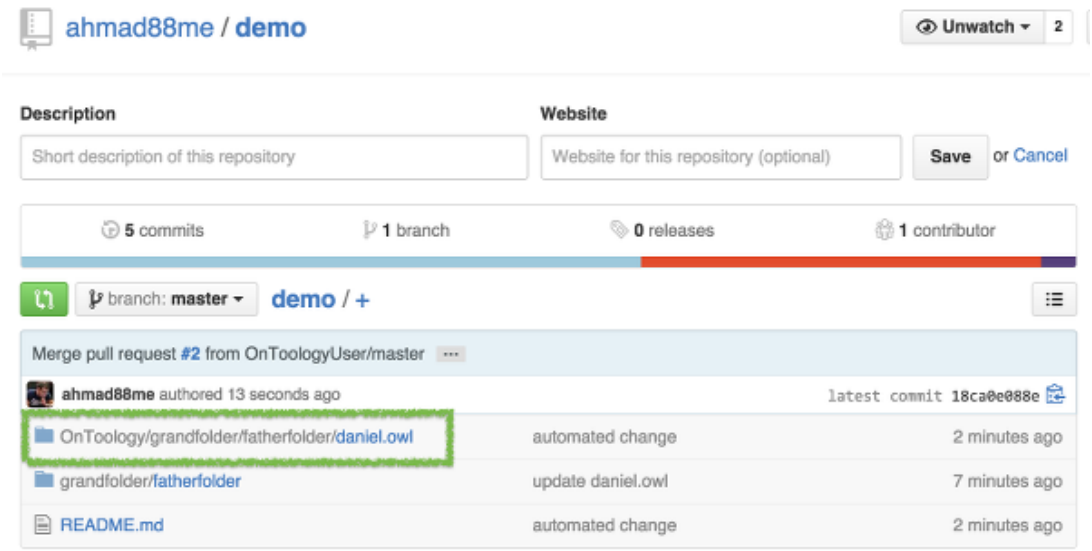

Figure 3-5. Creation of OnToology resource folder

Even though OnToology provides users with ready-to-publish core documentation extracted from the analyzed ontologies, most of the times users need to further customize parts of this initial HTML documentation to include examples or additional description about the functionality of the ontology. Once all the HTML sections have been customized, users may want to publish the ontology online. Whether it is the first time the users publish their ontology or it is just updating information, OnToology can handle the publication process by the permanent URI mentioned in Section 3.2, or the users can download all the HTML documentation to publish the ontological model in their own server. In the case of BIMERR, the latter option is selected.

The BO2DM tool is also tracking the repository that stores the enriched version of the ontological models. The converter generates the data model on demand. When a request comes from any of the semantic modelling tools or a specific user, the module retrieves the respective ontology and generates its JSON-serialized counterpart.

3.7 LICENSING

OnToology and BO2DM are licensed under the Apache 2.0[6] license.

[6] https://www.apache.org/licenses/LICENSE-2.0

4. MODEL MAPPER

4.1 OVERVIEW

The functionalities of the Model Mapper subcomponent are of great significance for the interoperability between the BIMERR applications as data previously modeled following different standards and models will be harmonized under a common data model, which will facilitate their sharing among the BIMERR applications. The appropriately configured data model mapping processes ensure the smooth and well-structured "data collection" of application data (i.e. uploading an application's data to the BIF) and sensor data through the Middleware, as well as the compatibility of the interoperating applications.

If the user has declared that the data to be retrieved by a data collection job shall undergo the mapping stage, instead of being handled as an object, the Model Mapper becomes involved in the "data collection" preparation and configuration step. Its core functionalities entail the mapping prediction and then the semi-automatic mapping configuration through the population of a "Configuration file" with data model mapping information, following a semi-automated process. The mapping information, which is typically included in the "Configuration file", defines how the underlying data model of the data to be exchanged through the BIF should be mapped to a selected BIMERR data model.

More specifically, the functionalities provided by the Model Mapper are the following:

- **Automatic calculation of mapping predictions based on a set of matching techniques**: The Model Mapper shall combine the information provided by the users and shall extract the underlying data model from the data sample that has been uploaded. With the help of various techniques ranging from fuzzy matching (i.e. matching the user input field to model fields based on the names and the related terms of the leaf nodes using levenshtein distance) and elastic matching (i.e. matching the user input field to model fields based on various text fields of the model's leaf nodes using advanced queries on the indexed data models) to sample-based matching (applying different algorithms for learning from the sample contents) and standards-based matching (that considers the predefined mapping of the data model to existing data models for the concept at hand), the Model Mapper

provides possible mappings from the external data model concepts to the BIMERR concepts. These predictions are accompanied by calibrated confidence scores.

- **Manual mapping configuration confirmation and updates by the users**: The Model Mapper shall provide an intuitive user interface to the users, where they shall navigate through the mapping predictions, providing the necessary mapping and transformation details or updating them to the correct concepts. Through the Model Mapper, the users shall identify the mappings or select alternative, related concepts for any unidentified concepts (for which the mapping prediction confidence score was not above a certain threshold). In this context, the Model Mapper shall also offer the option to define data types, measurement units and define any other required transformations (e.g. for datetime fields). The final mapping configurations are stored in the "Configuration File".
- **Easy navigation to the BIMERR data models**: The Model Mapper shall facilitate the users in understanding the different concepts that appear in the BIMERR data models in order to make prediction reconciliations and manual mappings in a more effortless manner.
- **User-driven proposition of new concepts**: In case the concepts in the underlying BIMERR data models do not effectively address the needs of a BIMERR application, the Model Mapper shall allow for suggestions for new concepts in an easy manner. Once created, the suggestions are forwarded to the Model Lifecycle Manager subcomponent, in order to be semi-automatically handled and further processed by the model administrator, whenever needed.

4.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The Model Mapper builds on state-of-the art technologies across 3 layers: the Presentation Layer, containing the Model Mapper User Interface that is developed in VueJS[7] and TailwindCSS[8]; the Business Logic Layer, containing the different packages of the Model Mapper Backend that are based in the Flask micro web framework[9]; and the Data Access Layer that essentially refers to the BIF Storage and Indexing that has been set up in the context of the

[7] https://vuejs.org/
[8] https://tailwindcss.com/
[9] https://flask.palletsprojects.com/en/1.1.x/

Building Information Collection and Enrichment component and utilizes ElasticSearch[10] and PostgreSQL[11], for the Model Mapper needs.

Such layers along with the different technologies are depicted in the following figure.

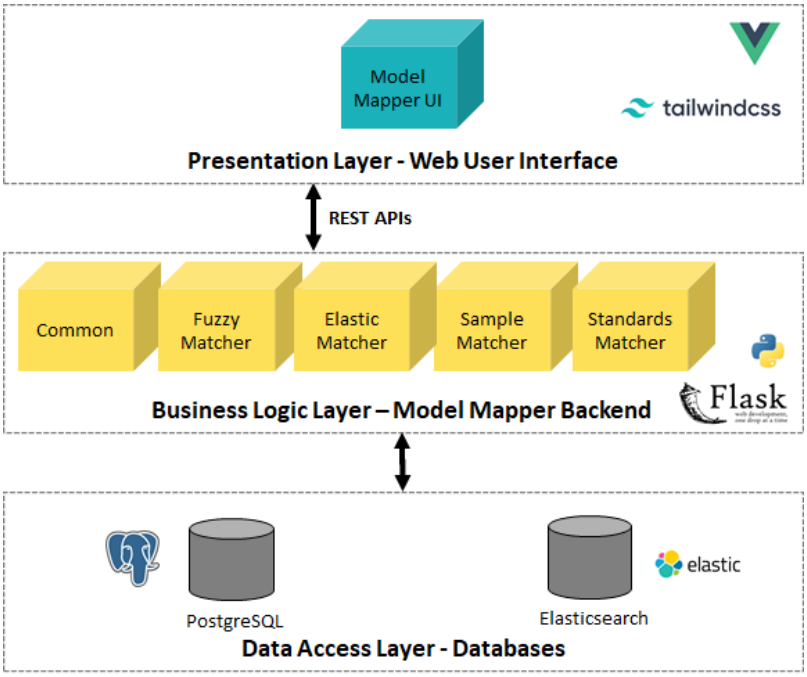

Figure 4-1: Architecture of the Model Mapper under the BIMERR Semantic Modelling component

The Model Mapper is written in Python 3.8.2[12] and utilizes the following open source technologies as defined in **Error! Reference source not found.**.

Table 4-1: Technologies and llibraries used in the Model Mapper, along 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)

[10] https://www.elastic.co/
[11] https://www.postgresql.org/
[12] https://www.python.org/

Name of the Library	Version	License
Elasticsearch	7.6.0	Elastic License
Elasticsearch DSL	7.6.0	Apache License 2.0
Vue.js	2.6.11	MIT
TailwindCSS	-	MIT
NumPy	1.18.1	BSD
Pandas	1.0.3	BSD 3-Clause
scikit-learn	0.22.1	BSD 3-Clause

4.3 API Documentation

All APIs that accompany the 1st version of the Model Mapper are documented in Swagger[13] and indicatively include: (a) GET /matching-prediction, and (b) POST /matching-configuration.

4.4 Assumptions and Restrictions

In the draft release of the Model Mapper, where the development activities for the different BIMERR applications are still ongoing, a number of assumptions (that in certain cases, represent restrictions for the Model Mapper) were taken:

- The Model Mapper will receive the data sample and data structure in a JSON format that may be flattened or non-flattened (in a nested structure), depending on the building data collection parameters.
- The Model Mapper will allow users to select a single data model for each data collection job, so it is not possible to map the data to concepts belonging to different BIMERR data models. This is an assumption taken since the BIMERR ontologies and data models represent different domains and the data that are to be uploaded by the applications are not expected to be cross-domain.

[13] https://swagger.io/

- The Model Mapper will only return "single" results per concept that appears in the source data, without offering any alternatives in the BIMERR data model to which a source concept could be also potentially mapped. Such an assumption was taken in order to offer a better user experience without overloading or confusing the users.
- The Model Mapper may predict mapping of the same concept of the applicable BIMERR data model to multiple source concepts. Such a decision was taken not only since there are concepts that may appear multiple times in the data, but also because the users should review the "best-predicted" mappings and select the most appropriate concept to their needs.
- The combination of underlying matching techniques needs extension and fine-tuning once actual data that are to be exchanged through the BIF are mapped with the help of the Model Manager.
- The sample-based, machine learning techniques require significant training per data model in order to make reliable predictions since they currently yield the lowest confidence scores. Such training was not possible without the actual BIMERR applications, but a pool of general-purpose training data will be created by the data model administrators once data start being collected in the BIF. Such training data practically refer to a variety of data values that typically accompany the different concepts, rather than large volumes of data samples, and will be derived during the pre-validation and early validation activities.

4.5 INSTALLATION INSTRUCTIONS

The Model Mapper is served as a web application and does not require the installation of any component by the user. Detailed instructions for the Model Mapper deployment are provided in the related private code repo and all subcomponents are already available as Docker containers to speed up the process.

4.6 USAGE WALKTHROUGH

As part of the data collection job definition (that is described in detail in the BIMERR Deliverable D4.6 [6]), the users (e.g. the BIMERR Application owner/developer) need to select the Mapping step as depicted in Figure 4-2 in order for the Model Mapper to be activated. In case the users skip the mapping step, any data uploaded though this data collection job will be treated as a single object, meaning that in later stages it will not be possible to perform other tasks requiring mapping, such as queries on these data.

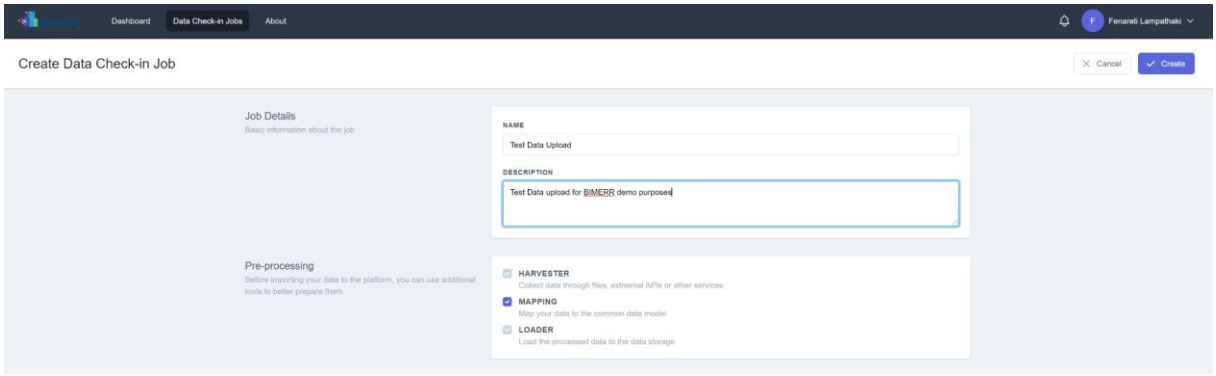

Figure 4-2: Activate the Model Mapper during the "Data Collection" job definition

As depicted in Figure 4-3, by selecting a data collection job in the list, the steps that have been activated are displayed and the users can create or update their configuration. When the users select the Mapping step, they may access the Model Mapper interface.

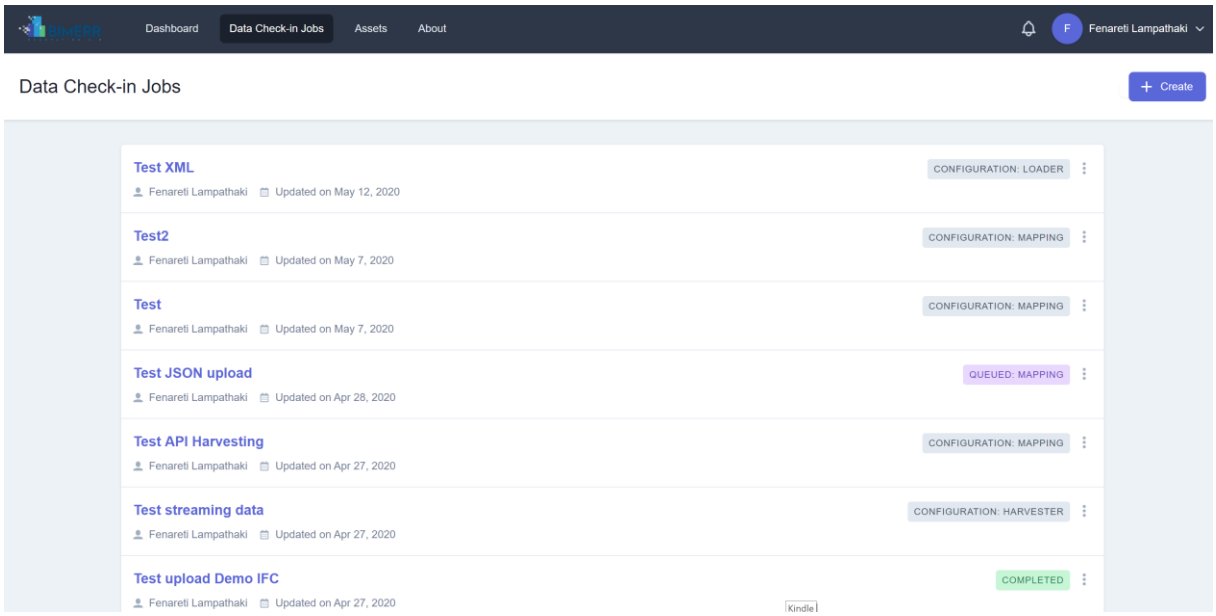

Figure 4-3: View the collection and mapping steps associated with each "Data Collection" job

In the Model Mapper interface that is viewed in Figure 4-4, the users are initially able to provide information regarding the main domain to which the data refer (e.g. Occupancy Profiling), the standards associated with the specific data model if applicable (otherwise the specific selection is not visible), and the core concept (e.g. Building Space) that represents the data. The domains directly refer to the titles of the different BIMERR ontologies and data models while the categories reflect the main "parent" concepts in the respective BIMERR data model.

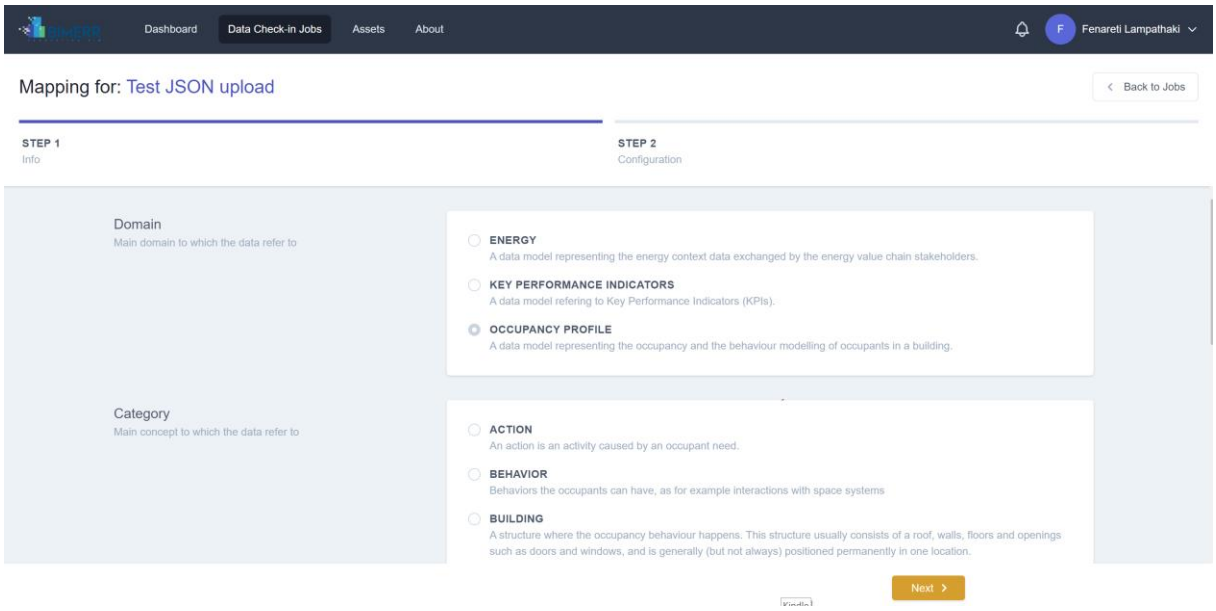

Figure 4-4: Model Mapper Step 1 – Mapping Information Provision

The Model Mapper combines such mapping information with the underlying source data model as extracted by the sample of the data collection job. As depicted in the Mapping Playground in Figure 4-5, the users view the initial predictions that have been provided by the Model Mapper from the source concepts of the data that the BIF shall collect through this data collection job, to the target concepts of the respective BIMERR data model (that was selected in Step 1). Such predictions are accompanied by confidence scores with color coding (e.g. green for high confidence, yellow for medium confidence and red for low confidence on the provided mappings). The users view the data type of each concept and can easily spot any data type mismatches that are highlighted with red. They are able to remove any wrong mapping (by selecting the "x" button on the top-right of each concept in the Playground) and filter the mappings to quickly navigate to the predicted / corrected / undefined / invalid / selected mappings. In the Data Model (left side of the screen in Figure 4-5), the users can view the respective data model and navigate to the details of each concept by selecting it in order to cross-check whether the mapping is correct.

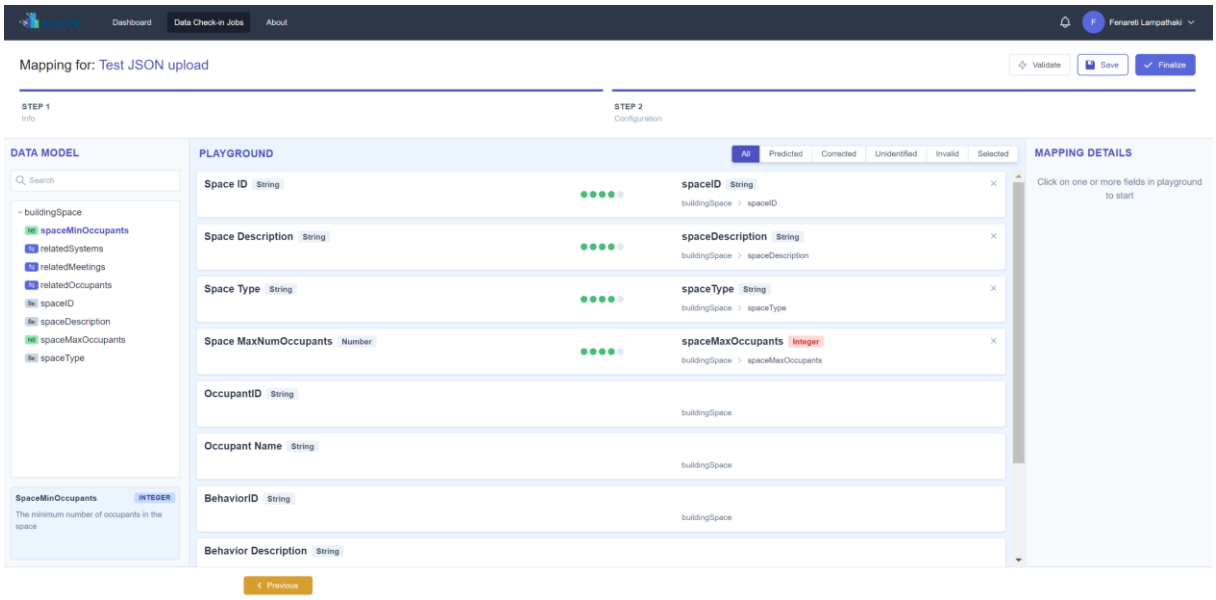

Figure 4-5: Model Mapper Step 2 – Mapping Playground Overview

By selecting a specific concept (that is highlighted in blue), the users can view the Mapping Details that are associated with the specific concept, and the sample values, as depicted in Figure 4-6. For example, in case the concept has a numeric data type, the users need to define the measurement unit with which the data comply in order for the mapping transformation functions to transform them to the baseline measurement unit followed in the respective BIMERR data model. If the concept refers to a datetime data type, the users need to define the datetime format from a large selection of supported formats (e.g. YYYY.MM.DD hh:mm:ss, DD/MM/YYYY hh:mm AM/PM, YYYYMMDDhhmmss) and the timezone to which the data refer.

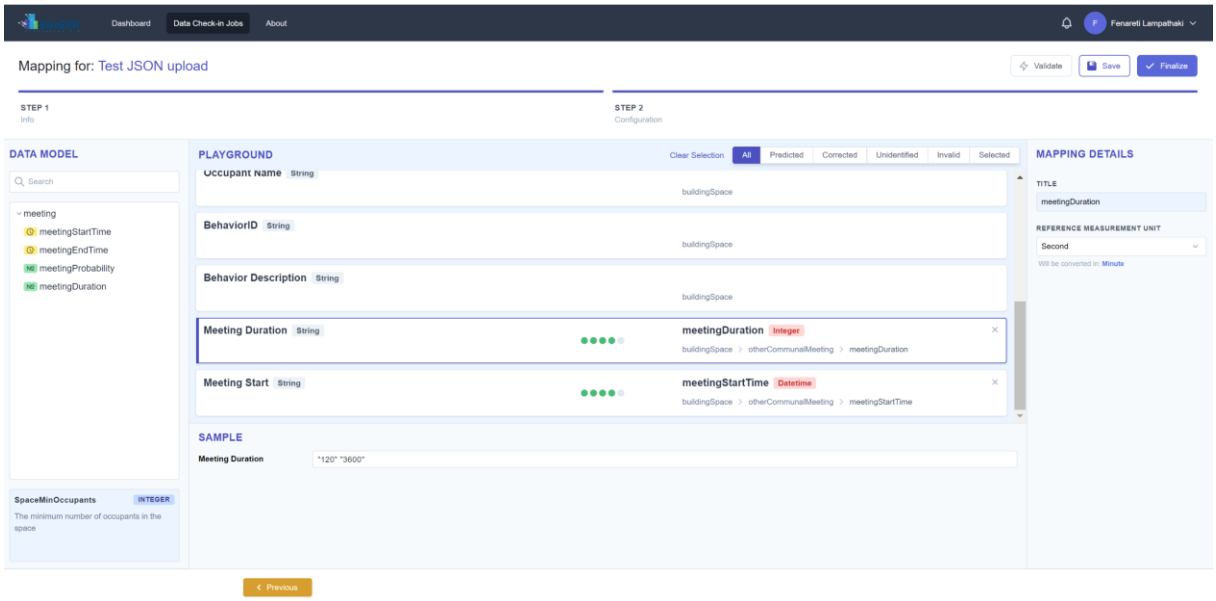

Figure 4-6: Model Mapper Step 2 – View/Edit Mapping Details for a selected concept

If the users select two or more concepts in the Playground, they are urged to identify the related concept to which they refer. Upon selecting the respective concept, they need to define the appropriate prefix or add their own (if this allowed by the respective BIMERR data model) as depicted in Figure 4-7. The users then either set the specific related concept or request for an updated prediction for the selected concepts.

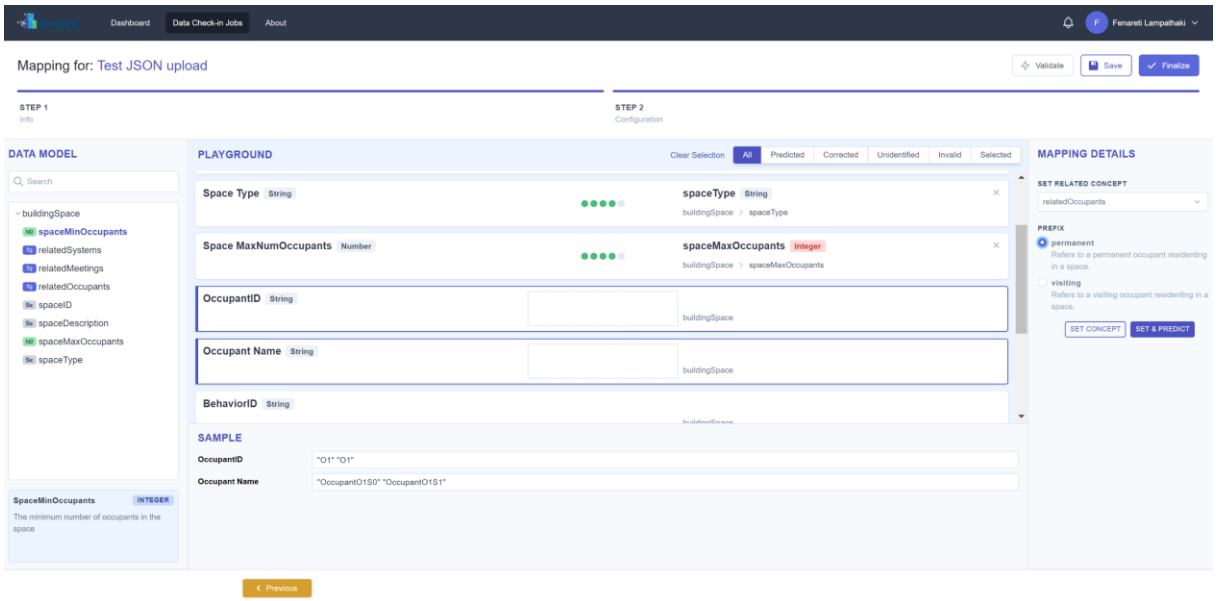

Figure 4-7: Model Mapper Step 2 – Set related concept

Upon reviewing the new predictions that have been provided step-by-step, the users may view the new concept that was selected (i.e. Occupant instead of the buildingSpace that was visible in the previous figures) in the Data Model in Figure 4-8.

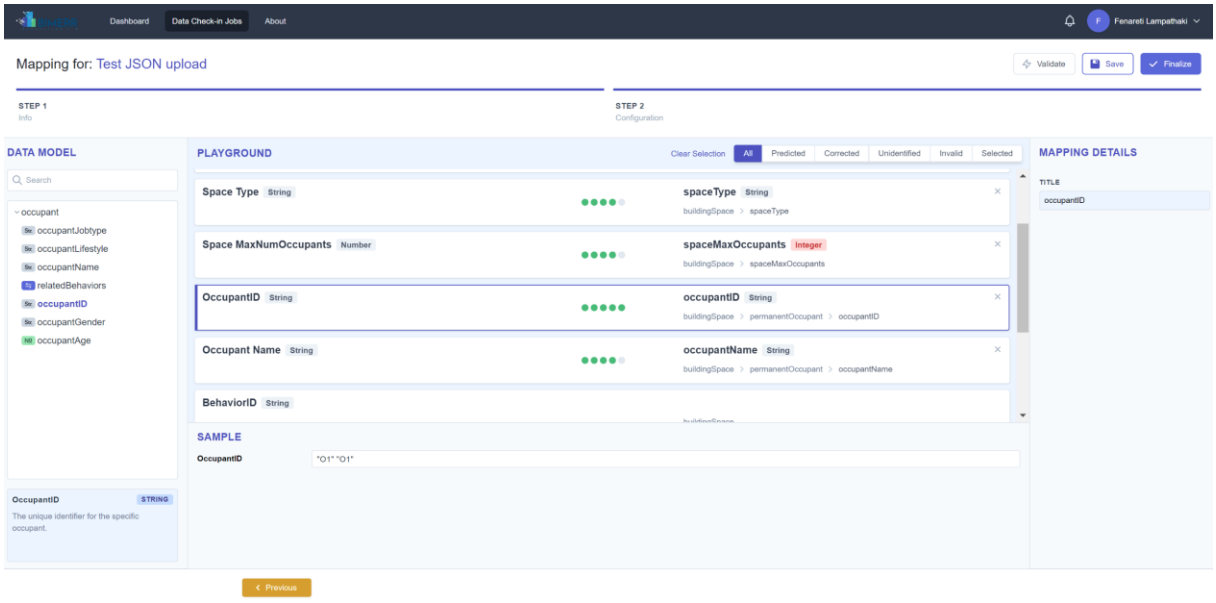

Figure 4-8: Model Mapper Step 2 – View new mappings for related concept

In case the users want to manually provide a mapping or update the automatically generated mappings, they may select a source concept from their data in the Playground and drag and drop a concept from the Data Model as depicted in Figure 4-9.

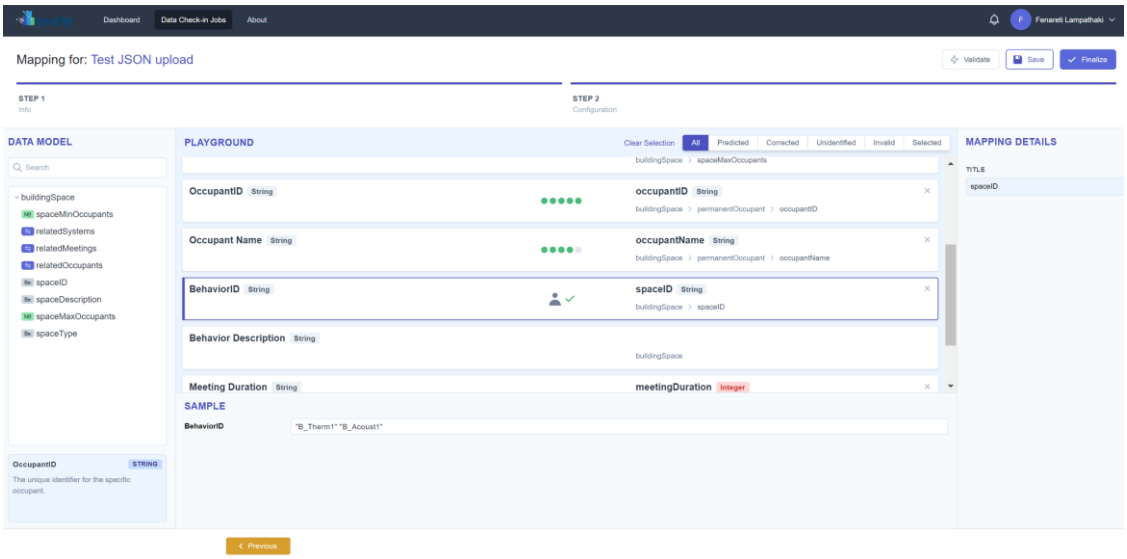

Figure 4-9: Model Mapper Step 2 – Create manual mappings for concepts

At any point, the users can validate the mappings that have been provided and save the progress made. In case there are any mapping validation errors, they are immediately highlighted to the users as depicted in Figure 4-10.

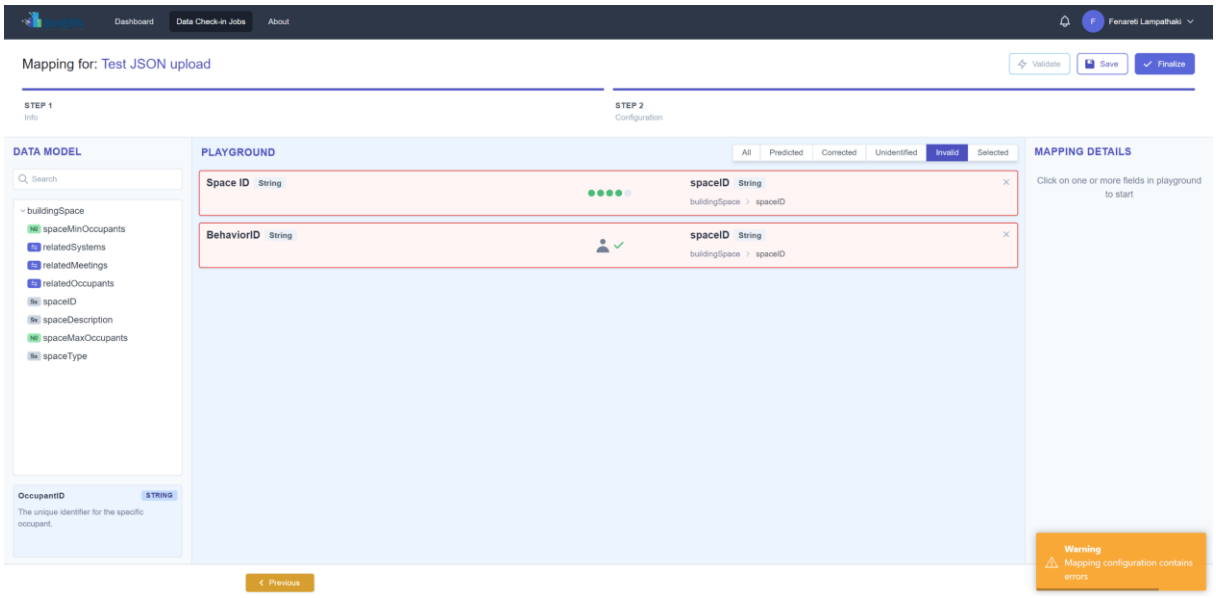

Figure 4-10: Model Mapper Step 2 – Validate mapping and invalid mappings detection

Regarding concepts that were not possible to be mapped to an existing BIMERR concept, they are indicated as 'Unidentified'. The users shall manually provide the mapping for the

unidentified concepts to BIMERR concepts or select alternative related concepts to which the mapping prediction should be executed, otherwise they are omitted from the final mapping configuration file and will be discarded from the actual mapped data that will be eventually uploaded in the BIF storage (with the help of the Building Information Collection & Enrichment Component).

In addition, the users can also propose through the Model Mapper subcomponent the update of an existing concept or the addition of a new concept in the BIMERR data model by providing its title, description, data type, related concept and mappings to existing models as depicted in Figure 4-11. The proposed concept is forwarded to the Model Lifecycle Manager, where it is handled by the data model administrator.

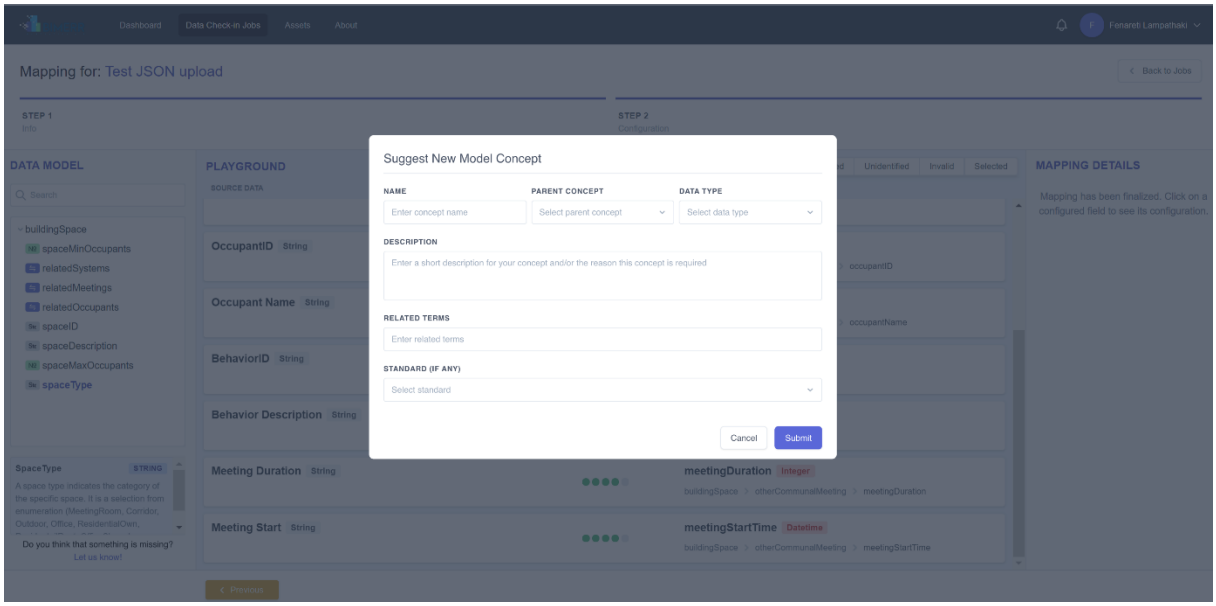

Figure 4-11: Model Mapper Step 2 – Propose a new concept

The users are able to save the mapping configuration in the "Configuration File" and revisit it at a later stage. If the mapping stage is "finalized", it cannot be edited any more by the users as the specific data collection job proceeds to the mapping execution stage.

4.7 LICENSING

The BIMERR Model Mapper is a closed source component.

5. MODEL LIFECYCLE MANAGER

5.1 OVERVIEW

The Model Lifecycle Manager of the BIF Semantic Modelling Component offers a set of features for the effective management, storage and evolution of the BIMERR data model by the BIMERR data model administrator. Such a component is also responsible for ensuring that the BIMERR data model is aligned with the BIMERR ontology in cases when the data model or the ontology are updated or enriched with concepts as required by the BIMERR Applications that exchange data through the BIF, in collaboration with the Ontology Manager Framework.

More specifically, the functionalities provided by the Model Lifecycle Manager are the following:

- **Navigation to the data model's concepts hierarchy and details**: The component provides to the data model administrator a user-friendly model navigation interface, where the administrator can manage the BIMERR data model concepts' hierarchy, as well as the individual concepts and their details.
- **"Spontaneous" and "upon-request" data model evolution events (create-update-deprecate) moderated by the data model administrator**: The Model Lifecycle Manager allows the data model administrator to update the data models' concept on their own volition (e.g. considering a new data model that emerged) or respond to the suggestions for new concepts provided by users in the Model Mapper. The data model administrator should perform any updates to existing concepts or additions of new concepts, through the user interface provided by the Model Lifecycle Manager. When the administrator has proceeded with a decision for a specific suggestion, the Model Lifecycle Manager sends an update to the Model Mapper regarding the progress and outcome of the suggestion at hand.
- **Validation of updated concepts to ensure the model's consistency**: Upon updating the BIMERR data model, the Model Lifecycle Manager shall validate that the changes introduced by the data model administrator have not generated any unforeseen disruptions, conflicts or inconsistencies with existing concepts (e.g. by introducing a new concept with a synonym name with an existing concept).

- **Persistence of the updated data model based on appropriate versioning conventions for consistency and traceability purposes**: Depending on the type of changes that have been introduced and the extent to which they are considered as "breaking", i.e. non backwards compatible, the Model Lifecycle Manager automatically applies appropriate versioning conventions (in terms of major and minor versions) for consistency and traceability purposes. The updated concepts are stored in the Data Storage and Indexing component (that is introduced in the Building Information Collection and Indexing component in D4.6 [6], but is applicable to the whole BIF).
- **Alignment between the BIMERR ontologies and data models**: The Model Lifecycle Manager allows importing an automatically extracted data model from the respective ontology through the BO2DM subcomponent of the Ontology Manager Framework (as described in Section 3). In addition, the Model Lifecycle Manager on its behalf will forward the new or updated concepts to the Ontology Manager Framework, in order for the component to update accordingly the respective ontology concepts and attributes and ensure the ontology-data model alignment.

5.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The Model Lifecycle Manager builds on state-of-the art technologies across three layers: the Presentation Layer, containing the Model Lifecycle Manager User Interface that is developed in VueJS[7] and TailwindCSS[8]; the Business Logic Layer, containing the different packages of the Model Lifecycle Manager Backend that are based in the Nest NodeJS web framework[14]; and the Data Access Layer that essentially refers to the BIF Storage and Indexing that has been set up in the context of the Building Information Collection and Enrichment component and utilizes ElasticSearch[10] and PostgresSQL[11], for the model lifecycle management needs.

Such layers along with the different technologies are depicted in the following figure.

[14] https://nestjs.com/

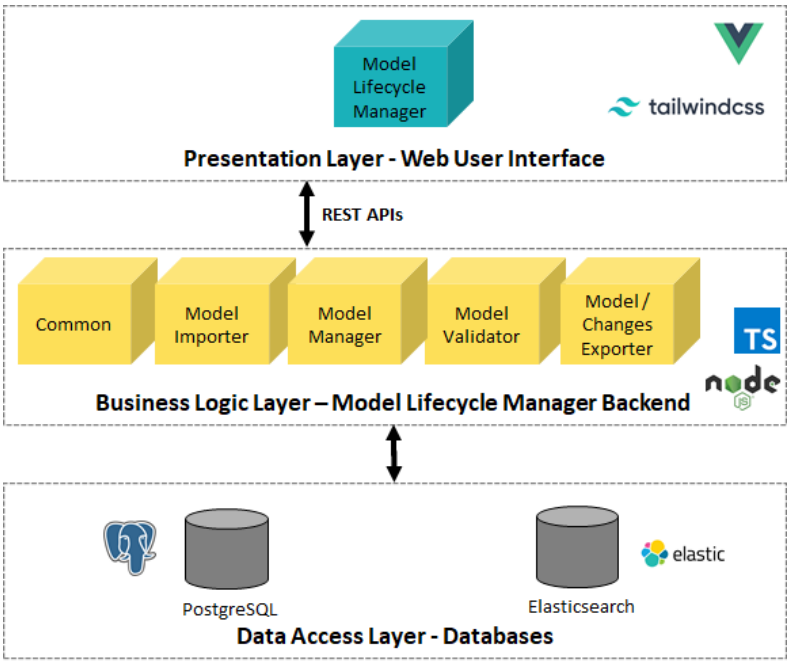

Figure 5-1: Architecture of the Model Lifecycle Manager under the BIMERR Semantic Modelling component

The Model Lifecycle Manager is written in Typescript[15] and utilizes the following open source technologies as depicted in **Error! Reference source not found.**-1:

Table 5-1: Technologies and libraries used in the Model Lifecycle Manager, along their licenses

Name of the Library	Version	License
Nest NodeJS Web Framework	7	MIT
TypeORM	-	MIT
PostgreSQL	12.2	PostgreSQL License (similar to BSD/MIT)
Elasticsearch	7.6.0	Elastic License
Vue.js	2.6.11	MIT
TailwindCSS	-	MIT

[15] https://www.typescriptlang.org/

5.3 API Documentation

The APIs that accompany the Model Lifecycle Manager will be documented in its final release.

5.4 Assumptions and Restrictions

In the first release of the Model Lifecycle Manager, where the development activities for the different BIMERR applications are still ongoing, a number of assumptions (that in certain cases, represent restrictions for the Model Lifecycle Manager) were taken:

- Each data model is self-standing and does not contain any relations to other data models. In case the data handled by the different BIMERR applications require a different approach in practice once all data models are available, such a decision will be reconsidered in the final release of the Model Lifecycle Manager.
- The alignment between the BIMERR ontologies and data models cannot be automatically ensured due to the inherent differences between the formats utilized (i.e. OWL and JSON). In order to ensure that the BIF will run according to the functionalities planned for semantic data mapping and transformation, the BIMERR data models require significant complementary information to be provided by the data model administrator. In parallel, when a change is introduced in the BIMERR data models, it is propagated to the ontology, but the ontology manager needs to also approve it as described in section 3.
- Over time, the BIMERR applications need change and certain evolution events, i.e. *addition* (of a new concept, a new sub-concept or a new standard), *update* (of metadata or sub-concepts) or *deletion* (of a concept, a sub-concept or their associated metadata), are anticipated as defined in the BIMERR Deliverable D4.2 [5]. It needs to be noted that different action is taken depending on the significance and the backwards compatibility of the foreseen event. Such actions range from direct adoption (through the *Propagate*

action) to compulsory validation by the data model administrator (in the *Prompt* action) and to prohibition of the specific evolution event (in the Block action).

- Deletion of certain concepts is not literally performed in the BIMERR data models as the concepts shall always be only deprecated in order to ensure backwards compatibility for data that have been already collected in the BIF.

5.5 Installation Instructions

The Model Lifecycle Manager is intended to be served as a web application and does not require the installation of ay component by the user. Detailed instructions for the deployment are provided in the related private code repo and all subcomponents are already provided as Docker containers to speed up the process.

5.6 Usage Walkthrough

The user interface of the Model Lifecycle Manager is planned to be delivered in the final version of the BIMERR Semantic Modelling Component. In the following lines, the user experience of its functionalities is described and it shall be complemented with the related screenshots of the Model Lifecycle Manager in the BIMERR Deliverable D4.5.

The users (i.e. data model administrators) can navigate through the concepts of the BIMERR data model, view their definitions, and understand the relations and details, through the provided user interface. The data model administrators will eventually need to perform updates on the BIMERR data model. Such a need may emerge from the evolution of popular external data models and standards, which are updated with new concepts, while rendering some other obsolete. Another possible scenario where the update of the BIMERR data model may be required, is when the application users have submitted their suggestions for new concepts and updates through the Model Mapper subcomponent (Figure 4-11).

In both cases, the data model administrators decide to propagate or reject the updates based on the methodology described in deliverable D4.2 - "BIMERR Ontology & Data Model 1" [5]. They check if this data model update introduces conflicts with existing concepts and whether the model will remain backwards compatible afterwards (i.e. existing mappings will not become invalid). If the update is expected to introduce compatibility issues, the administrators must evaluate the extent of additional actions required (e.g. transformations) to ensure compatibility of existing mappings and assess whether it is an acceptable cost compared to the value added by the update. If the suggestion gets accepted, the administrators either update an existing concept accordingly, or add the new concept under the related core concept in the BIMERR data model and complete the necessary details and metadata. The application owner/developer is informed regarding the outcome of her request, once it is closed. If the request has been rejected, the application owner/developer is instructed to use one of the existing BIMERR concepts for her needs or submit again a revised request in order to solve the conflicts that occurred.

Whenever changes occur to the data model or the ontology, triggered by a user, the introduction of a new standard or domain etc., they have to be bilaterally communicated and adopted. The ontology and data model versioning convention used in BIMERR facilitates this 'communication'. An update is indicated as 'minor', implying that no further action was required to ensure backwards compatibility for the existing BIMERR mappings, or 'major' for any concept update that required a series of further actions. The actual alignment of the BIMERR data model and the ontology, whenever a new version is available, is performed by a semi-automated rule-based process, which makes use of certain metadata, such as version and date_added. The Model Lifecycle Manager provides the updated concepts to the Ontology Manager Framework, which in turn can proceed with the update and alignment of the ontology.

5.7 LICENSING

The BIMERR Model Lifecycle Manager is a closed source component.

6. CONCLUSIONS AND PLAN FOR FINAL ITERATION

The BIMERR Semantic Modelling Component, along with the BIMERR Information Collection and Enrichment Component (that is documented in the BIMERR Deliverable D4.6 [6]), constitute 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 (D4.4), the core functionalities of the BIMERR Semantic Modelling Component along its three subcomponents, namely the Ontology Manager Framework, the Model Mapper and the Model Lifecycle Manager, have been developed as planned in terms of back-end processing requirements and front-end user needs. Such an initial version is currently being tested and evaluated in the context of the BIF integration activities towards its first release.

New features will be developed according to the BIF evaluation and feedback by the BIMERR applications and the pre-validation sites. Based on the current feedback from the initial demos in the online BIMERR plenary meeting on M16, the following features and extensions have been already planned for the final release of the BIMERR Semantic Modelling Component:

- Ontology Manager Framework – final release:
    - Automate the ontology publishing step to the web server by means of webhooks.
    - Design and implement the ontology updating process when changes in the data model occur.
    - Generate the enriched version of the already implemented ontological models, needed by the BO2DM converter.
- Model Mapper – final release:
    - Management of "multiple" predictions per concept (from the same data model and additional data models)
    - Improvements of the matching techniques upon experimentation with actual building data exchanged through the BIMERR applications

- Better notifications to the user for the inclusion of new concepts to the BIMERR data models of interest for a specific data collection job
- Support for links/hooks to existing code lists for certain data types, if anticipated by the respective BIMERR data model
- Export of the mapping configuration as a downloadable file

- Model Lifecycle Manager – final release:
  - Intuitive user interfaces for the different back-end functionalities
  - Semi-automatic evolution actions enforced once a new concept is proposed
  - Effective management of relations between different data models
  - Export of the BIMERR data models (in full or their changes)

ANNEX I: BIBLIOGRAPHY

[1] BIMERR (2018) Description of Action (DoA)
[2] BIMERR (2019a) D3.1 - Stakeholder requirements for the BIMERR system
[3] BIMERR (2019b) D4.1 - Report on Semantic Alignment & Linking of EEB-related Ontologies
[4] BIMERR (2020a) D3.5 - BIMERR system architecture 1st version
[5] BIMERR (2020b) D4.2 - BIMERR Ontology & Data Model
[6] BIMERR (2020b) D4.6 - BIMERR Information Collection & Enrichment Tool 1