BIM4EEB

BIM based fast toolkit for
Efficient rEnovation in Buildings

D4.8 Guidelines for the integration of new tools in the BIM Management System

This project has received funding from European Union's H2020 research and innovation programme under grant agreement N. 820660

Programmes	H2020
Call for Proposal	LC-EEB-02-2018 Building information modelling adapted to efficient renovation
Project Title	BIM based fast toolkit for Efficient rEnovation in Buildings
Acronym	BIM4EEB
Project Grant Agreement	820660

Work Package	WP4
Lead Partner	One Team
Contributing Partner(s)	One Team
Dissemination Level	Public
Type	Other
Due date	30/06/2020
Date	30/06/2020
Version	1.0

DOCUMENT HISTORY

Version	Date	Comments	Main Authors
0.1	10.03.2020	TOC added	Davide Madeddu (One Team), Alessandro Valra (One Team), Diego Farina (One Team), Jacopo Chiappetti (One Team)
0.5	21.04.2020	First draft	Davide Madeddu (One Team), Alessandro Valra (One Team), Diego Farina (One Team), Jacopo Chiappetti (One Team)
0.7	26.05.2020	Second draft	Davide Madeddu (One Team), Alessandro Valra (One Team), Diego Farina (One Team), Jacopo Chiappetti (One Team)
0.8	19.06.2020	Ready for review	One Team
0.9	25.06.2020	Review Done	Andriy Hryshchenko (UCC), Markku Kiviniemi (VTT), Teemu Matasniemi (VTT), Teemu Vesanen (VTT)
1.0 FINAL	30.06.2020	Final Version	Davide Madeddu (One Team), Alessandro Valra (One Team), Diego Farina (One Team), Jacopo Chiappetti (One Team)

BIM4EEB action has received funding from the European Union under grant agreement number 820660.

EXECUTIVE SUMMARY

The BIM Management System (BIMMS) is an open source and interoperable platform with a specific toolkit to optimise the information management (collect, share, update and exchange) during the different stages of a BIM-based renovation process. This makes the process more efficient than traditional one.
The toolkit, developed by the other partners of the consortium in the respective WPs, allows for the information collection of existing buildings and for the support solutions in the early design phases. Furthermore, these factors have positive conseguences on the field construction operations in terms of time, cost, labour, materials, and energy management.
The scope of this deliverable is twofold: (1) explain the integration between the BIMMS and the the toolkit (the existing developed tools), and (2) show how it is possible to integrate new tools to improve the functionalities of the platform.
The deliverable provides a guideline for the integration of tools in the BIMMS explaining the overall structure of the platform, giving details about software architecture, its technology and the technical requirements.
More specifically, in the first section, a brief description of the toolkit functionalities is provided. In the second section, the concept of tool integration and the data exchange management in the BIMMS environment is introduced. The third section shows the principle steps to drive the developers in the integration of their applications in the BIMMS. Finally, some common use cases are described that can be used to implement the most common functionalities using the BIMMS APIs.
The development of the guidelines are congruent with the ongoing work in the parallel tasks in WP4 and in the other WPs by the other developer partners.

PUBLISHING SUMMARY

The aim of this deliverable is the development of guidelines for the integration of new tools in the BIM Management System (BIMMS). The BIMMS platform is composed of set of digital tools (BIM4EEB toolkit) in order to support BIM-based building renovation. The guidelines has been established taking into account several aspects: (1) the advancements in modern architectural models used by web applications with new workflows and standards, (2) the data exchange requirements defined for the whole BIMMS Platform, and (3) the toolkit developed by the other project partners in the respective WPs and , introducing workflows, standards and methods.
The integration of new tools will help to support and to provide attractive solutions for building retrofitting for designers, construction companies, owners, officials and end users.

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

1.1 Scope of the document

This document describes the work carried out in Working Package (WP) 4 - Task 4.5 started in M13 and that will end in M30 after 17 months of duration. The development plan has set two main goals: the first one in M18 (June 2020) where a preliminary toolset for integration testing purposes with the core functionality is released, and the second one in M24 (December 2020) where the final toolset for validation purposes with the complete functionalities will be released. After M24, the application of the BIMMS and the entire toolset in the demo cases scenarios is expected.

The document provides an overview about the tool integration and the software architecture of the BIMMS. The main functionalities of the toolkit in development are introduced with the prerequisites needed to integrate the tools in the platform.

The main scope of the document is to give a getting started guide for the integration of developers' applications in the BIMMS. To do that, a brief description of some common use cases can be used as blueprint in the implementation of the functionalities using the BIMMS APIs.

These guidelines are congruent with the ongoing work of the other WP4's tasks. The development has also a strong connection with the activities and the software requirements of the other WPs, and this could affect the current development roadmap of some functionalities to support the other developer partners of other WPs.

1.2 Relevance with the other deliverables

The requirements for the BIMMS tool integration are defined in light of the specifications published in the Task 4.1 with the deliverable D 4.1 “Specifications and overall design of a BIM management system”, , where it is possible to find the essential functionalities for the tools developed in WP5, WP6 and WP7, i.e., how to collect, exchange and structure the information. Moreover, the Task 4.2 in the deliverable D 4.2 “Definition of users’ profiles for accessing the BIM management system” was useful to establish the users’ profiles and their functions allowed for each group of users of the BIMMS.

The guidelines proposed are based on the BIMMS APIs reported in the Deliverable D4.7 “API, Master end user front end”, that contains the end user manual, the APIs descriptions and documentations.

More details about the software architecture, the technical configuration of the platform and the development of the additional services on the BIMMS are available in these deliverables:

- PUBLIC DELIVERABLES:
    - Deliverable D4.7 “API, Master end user front end”, describes the APIs, explains APIs documentation and the End User manual
- CONFIDENTIAL ONLY FOR CONSORTIUM OR ADVISORY BOARD MEMBERS
    - Deliverable D4.3 “CDE Database, Core DB functionalities, Ontological representation, BIM data translation engine to ontology” describes the CDE DB, enriched with the core DB functionalities such as ontological representations of items and their relationships, a translation engine from BIM models to ontology, and etc..
    - Deliverable D4.4 “CDE Services, Interoperability Services, Exchange Layer Services, I/O Protocols and Data specification” deals with the CDE Services Interoperability Services, Exchange Layer Services, I/O Protocols and Data specification.
    - Deliverable D4.5 “Technical configuration of the platform”, as the name suggests, gives indications about hardware and software configuration, OS and DB setup.
    - Deliverable D4.6 “Guidelines for the implementation of the BIM management system” provides the documentation such as user’s manual and guidelines to define objects, relationships and data.

1.3 Structure of the document

The document is structure in the following parts: the first one provides a brief description of the the toolkit functionalities, currently in progress. The second one introduces the concept of tool integration and the data exchange management within the BIMMS environment. The third one suggests recommendations for developers in the BIMMS. Finally, some common use cases are described that can be used to implement the most common functionalities using the BIMMS APIs.

2 The BIM Management System (BIMMS) Toolkit

The BIM Management System (BIMMS) is a platform built around a Common Data Environment (CDE) that store all the data and information gathered through different sources and along the whole building life-cycle, acting a single source of truth (SSOT). The Common Data Environment (CDE) with dedicated interfaces and capabilities works as a central repository enabling the data exchange with connected tools for providing better coordination among users and process phases.

The BIMMS platform gathers a set of digital tools (BIM4EEB toolkit) to support BIM-based building renovation. The Common Data Environment (CDE) allows to collaborate and to store, share and visualise BIM and GIS (Geographic Information System) models, manage data and link the data-streaming from sensors to the models. The set of the tools that are part of the BIM4EEB Toolkit are:

- BIM4EEB Fast Mapping of Buildings Toolkit: provides tools and a methodology that speeds-up the scan-to-BIM process and improves the data visualisation of an existing building by using Augmented Reality (AR);

- BIM4EEB BIMeaser tool: It supports the decision-making process in the early design stage of the renovation process. Using the BIM and linked data from the BIM Management System, BIMeaser enhances the functionalities of the existing energy simulators evaluating several energy refurbishment design options to provide solutions that best fit the requirements while optimising the energy use and comfortable indoor climateconditions.

- BIM4EEB BIM4Occupants tool: provides to Owners and Inhabitants the information related to the renovation activities performed in their building, as well as with personalised information related to their building ambient conditions, comfort preferences and energy profiling;

- BIM4EEB Auteras tool: It supports building services designers to design Room Automation Systems (as part of Building Automation Systems) with a semi-automated process of a functional requirement survey and the generation of function block-based designs;

- BIM4EEB BIMcpd tool: is a software suite that recommends the placement of HVAC, lighting, fire and other devices in the most ideal solution in the constraint checking tool. It will also allow the user to complete Measurement and Verification projects complying with the International Performance Measurement and Verification Protocol (IPMVP).

- BIM4EEB BIMPlanner tool: It provides a digital environment to share up-to-date information about the plans and progress of site operations with all participants of a renovation project.

The BIMMS software architecure allows to integrate new tools to extend and further improve the functionalities of the platform. Any tool that will takes advantabe by exploiting the data managed in the BIMMS can be integrated in the platform. Developers can connect their applications to extends the data in any manner that could improve the renovation works activities and the process business logic. The existing toolkit is an example of tools that will used in a typical renovation workflow and give an idea of the wide range of applications that will be integrated.

3 Tool Integration

Integration means to bring together or incorporate parts into a whole. Integration does not mean to merge tools and functionalities in a unique platform but enabling data exchange between applications. With tools integration the user can enter data once and distribute it to multiple connected applications reducing errors, manual intervention and assuring consistency across different platforms. Tool integration is applied in different industry and businesses contexts, because helps to supply different interfaces to manage different data flows.

There are some benefits coming from tool integration directly related to the development business that have positive effects also for the users interaction. The development of the tools can be independent, using different programming languages, without software and library dependencies between tools. This positively affect the deploy in terms of software development platforms, user interface development, business logic and resource management. The developer can focus on user needs, and the tool can be updated, scaled and extended in the future to fulfil the new requirements.

In the context of the BIM4EEB project, the architecture, engineering and construction industry is usually high demanding context in terms of data exchange, functionalities and integration. A tool integration strategy shall be able to support designers in the design and planning phase, construction companies to efficiently carry out the work and service companies to provide attractive solutions for building retrofitting. Moreover the tools will be used by public and private owners for decision making and asset management and by professionals and technical roles to design, plan, construct and manage the building retrofit. In this scenario the data exchange complexity emerge, and the toolkit will be able to deal not only with different datasets, standards, but also with complex user requirements in terms of data delivery and management.

In this context the BIMMS integrates the BIM4EEB toolkit, as set of tools that works as basic instrument for increasing semantic interoperability between software and stakeholders involved along the overall renovation process (design, planning, construction, performance assessment and management). New applications can be developed to integrate and expand the existing set of tools, exploiting the data shared in the BIMMS. The BIMMS is developed as open architecture to promote any addiction, extension and future upgrade of the existing functionalities using common and interoperable frameworks and services.

3.1 The BIMMS CDE Interoperability and Exchange Layer Services

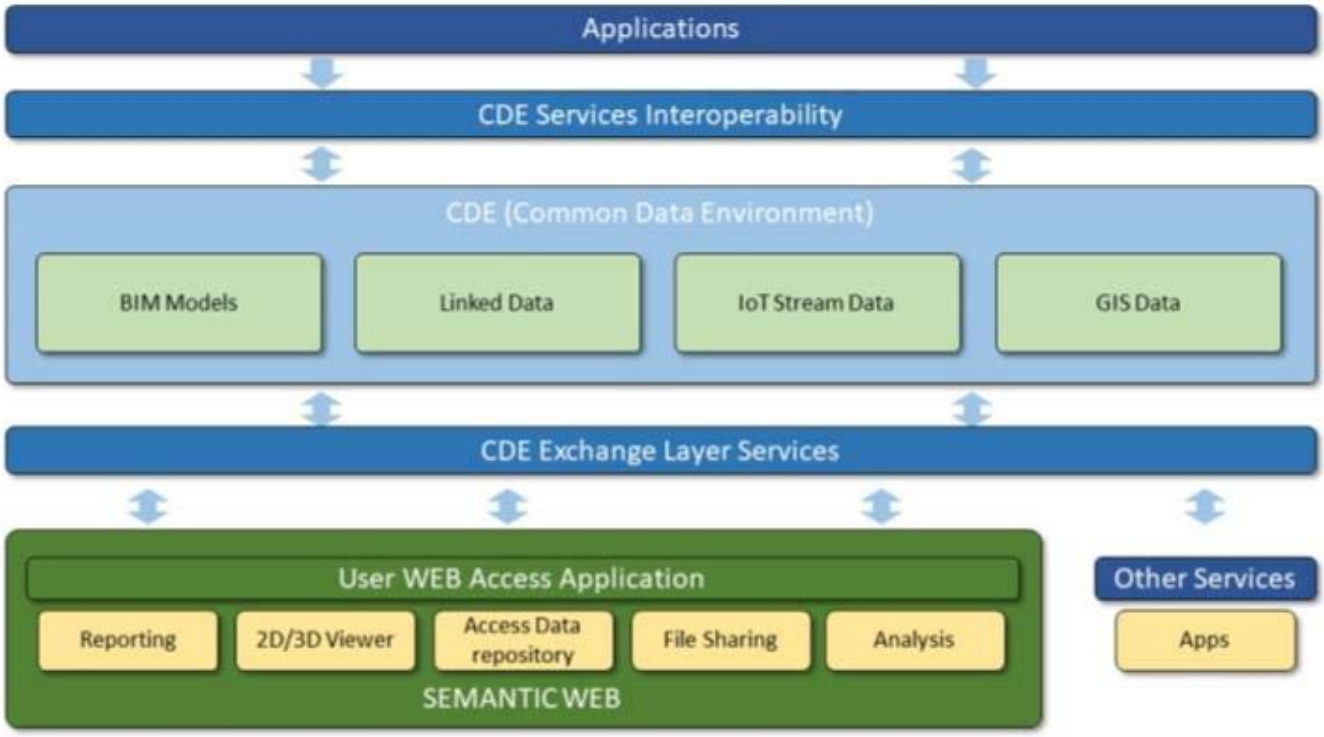


Figure 1 – The main BIMMS modules

Figure 1 shows the architectural system highlighting the user web applications and tools developed by the other partners in the respective WPs. For more details, refer to D4.7 "API, Master End-User Front End", paragraph 2 The BIMMS Exchange Layer Services.

4 Prerequisites to integrate new tools in the BIMMS

4.1 Tool registration

Any tool that want to connect to the BIMMS must be registered. The registration will allow to define a unique name of the application, a brief description of the application and a image to be used as icon. All this information shall be sent to the BIMMS Responsible party and allow them to insert the tool in the tool list of the BIMMS Web portal and make available the tool for the interested users.

Further developments that will improve the registration process can be planned to include more features. These improvements will be agreed with other WPs Developers Partners.

As improvement One Team propose to include a workflow where the developer that wants to create an application that will connect to the BIMMS must be registered first to the BIMMS as Developer Role. The Developer then will register the application using a specific web page in the BIMMS Web Portal.

Within registration the developer register the application defining:

- Tool name: will be the name that will appear in the BIMMS Tools list and will be shown to the users;
- Tool description: will be the brief description of the application that will appear in the BIMMS Tool list and will be show to the user;
- Developer name: will be the name of the responsible developer to be contacted for support or enquiry;
- Developer email: will be the email to contact the responsible developer;
- Call-back: will be the domain where the authentication process will set the authorization code.

The registration process automatically assign an applicationID and a applicationSecret that will be used for the authentication process of the tool in the BIMMS.

OAuth allows external applications to request authorization to a user's data. It allows users to grant and revoke API access on a per-application basis and keeps users' authentication details safe.

All developers need to register their application before getting started. A registered application will be assigned a client ID and client secret. The secret is used for authentication and should never be shared.

4.2 Authentication

In this chapter are explained the authentication processes used by the tools to access to the BIMMS data. It is important to distinguish the Authentication process of the applications from the standard authentication and log in process of the generic user that access to the BIMMS by the Web Portal. Indeed the applications to access to the BIMMS data must be authorized first by the BIMMS Exchange Layer Services authentication and then by the user that confirm the use of his or her data in the application.

Users in the BIMMS can be authorized using the form-based authentication.

The applications in the BIMMS can be authorized with two kind of authentication processes:

- Basic authentication;
- OAuth authentication;

A basic authentication was implemented to support the development environment giving a fast and easy system to authenticate applications in the BIMMS. With this strategy in the first phase of the development of the applications the developer Partners can put the effort to the development of data exchange services.

Furthermore the OAuth authentication was implemented to support the development and production environment, and will be available for the next development phases of the tools and the final release of the BIMMS Exchange Services.

Some terms in the following paragraphs are related to the definitions of the users and the roles used in the BIMMS and described in full detail in the Deliverable D4.6 where there also explained the relationship with the functionalities related to each role. For the purposes of the Authentication process the list below summarize some of these definitions:

- a Organization is any public or private company or organization that want to implement the BIMMS;
- a User is any people that can register to the BIMMS through the BIMMS Web Portal available at https://bim4eeb.oneteam.it/BIMMS/;
- a Role is a definition of a set of permissions that allow a user to access to specific functionalities of the BIMMS.

4.2.1 User Form-based authentication

A standard user will register in the BIMMS and then login using a Form-Based authentication. For security reason the connection between the user (the client) and the BIMMS (the server) is secured using HTTPS over TSL.

The process of authentication follow these steps

- A client requests access to a protected resource.
- If the client is unauthenticated, the server redirects the client to a login page.
- The client submits the login form to the server.
- If the login succeeds, the server redirects the client to the resource. If the login fails, the client is redirected to an error page.

The sensitive information is stored in the BIMMS and encrypted using Triple DES (3DES) symmetric-key block cipher.

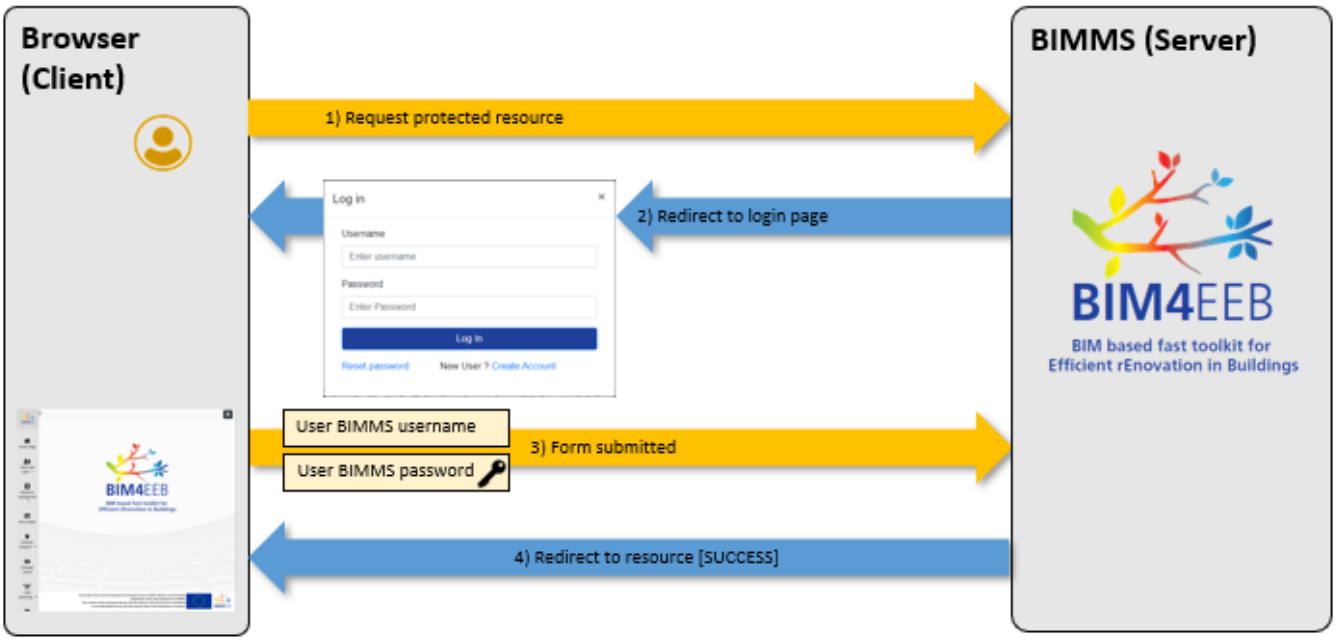


Figure 2 – User Form-based authentication

4.2.2 BIMMS Application Basic authentication

The Application Basic authentication process allows external applications to access to the BIMMS. Using Basic authentication is possible to interact with the data stored in the BIMMS without having to store sensitive information in the tools.

The tool's developer need to register the BIMMS before getting started. After registration the developer user is set as guest user role and can not access to the BIMMS data. The guest user can only access to their profile management settings to change his or her password.

The developer must be authorized by the Project Administrator that will assign the Application role and will enable the access to the Projects data stored in the BIMMS. Once authorized, the application can login and access to the data stored in the BIMMS using the registered developer user and the password.

The user that use the application must insert an Application Token to authorize the tool to access to his or her data stored in the BIMMS. The Application Token is a personal code string automatically generated and stored in the BIMMS with encryption. The user can retrieve this code accessing to his or her account profile on application tools management page.

The Application access to the BIMMS user data with no access to the BIMMS user credentials or other sensible data. The tool (the client) and the BIMMS (the server) exchanged data using specific API methods that will return information with anonymous application token (only the user knows the application token passcode).

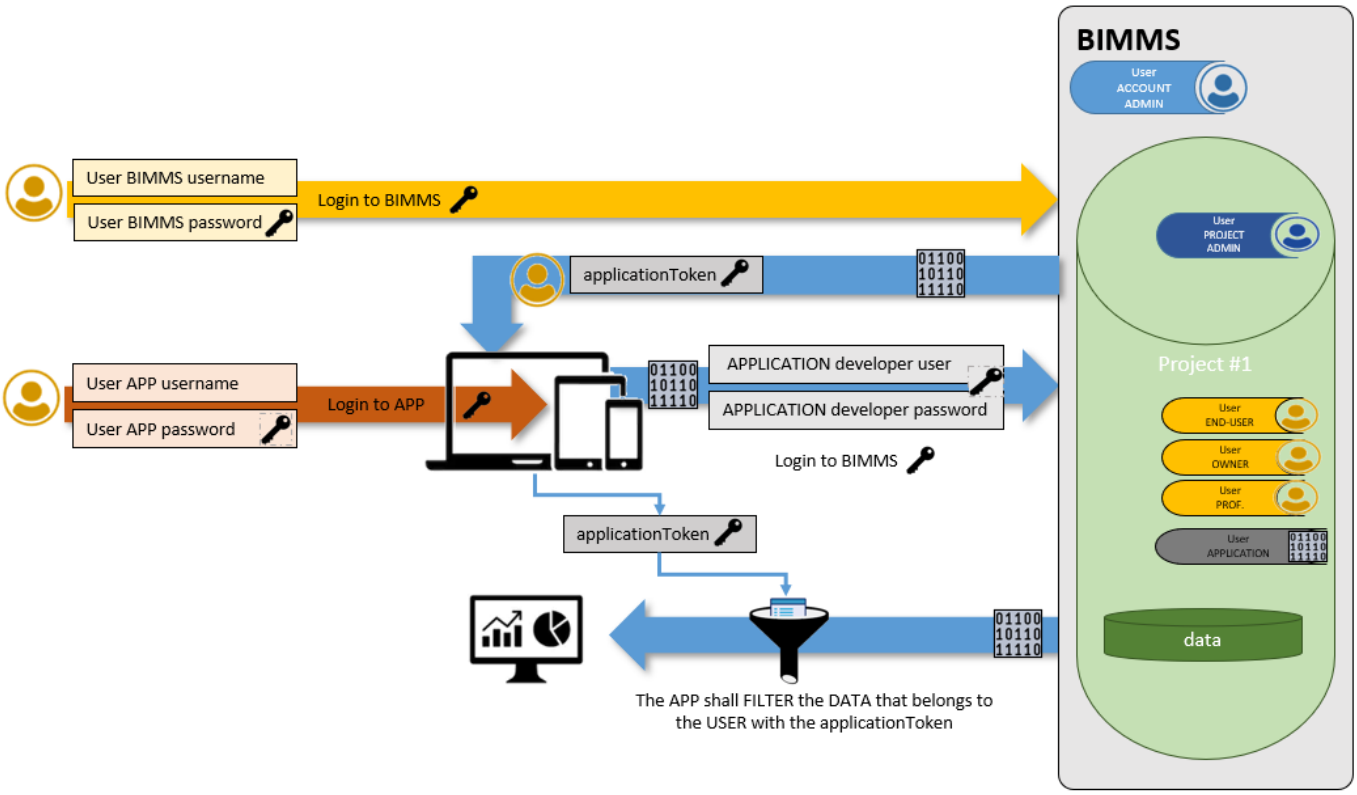


Figure 3 – BIMMS Application Basic authentication

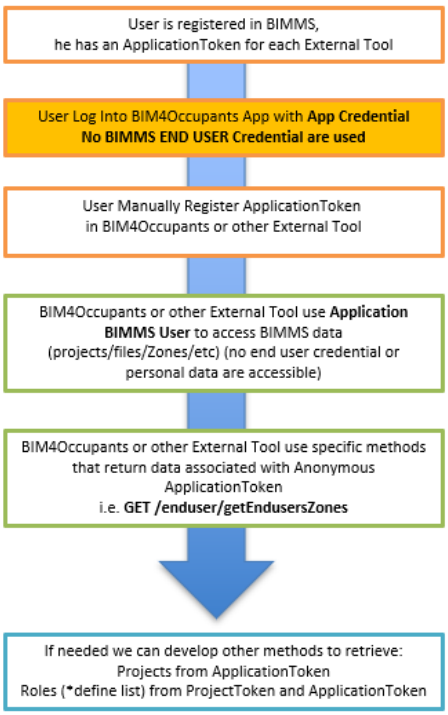


Figure 4 – BIMMS Application Basic authentication with APIs methods to filter the data that belongs to the user.

4.2.3 Application OAuth authentication

OAuth allows external applications to request authorization to a user's data. It allows users to grant and revoke API access on a per-application basis and keeps users' authentication details safe. Using OAuth is possible to interact with the data stored in the BIMMS without having to store sensitive information in the tools.

All developers need to register their application before getting started. A registered application user will be assigned a developer token over his login and password. The token is used for authentication and should never be shared.

When OAuth is initiated, the user is prompted by the application to log in on the BIMMS and to give consent to the requesting application. A user can opt out of the scopes requested by the application.

After the user accepts or rejects the authorization request, the BIMMS redirects the user to a URL specified by the application. If the user authorized the application, the URL query string will include an authorization code (end-user application token) and the scope accepted by the user. Apps should check which scopes a user has accepted. Applications complete the authorization process by exchanging the given authorization code. These tokens are used by applications to obtain and modify BIMMS resources on behalf of the authenticated user.

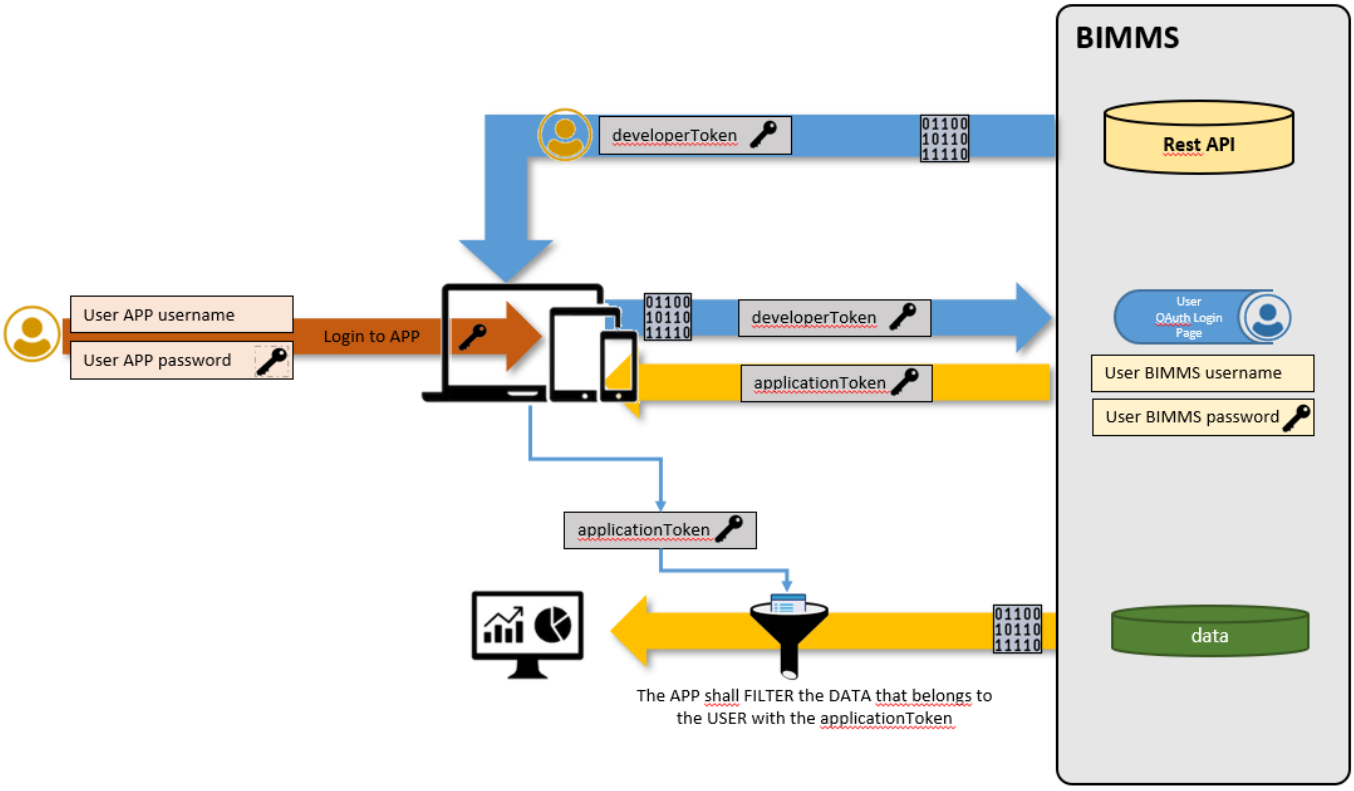


Figure 5 - BIMMS Application OAuth authentication to filter the data that belongs to the user

5 API Use case scenario

The BIMMS REST API Service allow to put, edit and get data from and to the BIMMS. To work with API the user will have to register to the BIMMS. Applications must be authorized to get access to the data using the Authentication methods described in Paragraph 4.2 of this Deliverable.

The list of the REST API is available on the BIMMS Web Portal under the Endpoints in the left side menu. The Endpoints page give a brief documentation of the methods and also the accesso to the Swagger UI service. The Swagger UI is a open source service supported by Smartbear, that help developers and end users to visualize and interact with the API resources without having any of the implementation logic in the place (Swagger, 2020). The Swagger UI allow to familiarize with the BIMMS API by submitting HTTP requests and getting the responses before write any line of code in development environment and make easy the backend implementation and the client side consumption.

The APIs have been organized into homogeneous groups, the different API groups are dedicated to different development areas; users enabled to use the API, access all the interoperability features of the services at the moment.

The API groups of the BIMMS system are as follows:

- OTMySQL: They are all the APIs of the BIMMS system that use the MySQL Database as the data source, the central database of the BIMMS system. The macro group are divided into sub-areas:
    - User API: these are all the APIs that allow access to authentication services, the definition of user access profiles and the projects accessible to the user who uses the API;
    - Project API: these are all the APIs that allow access to the structure of each projects and graphs associated with the project;
    - IFC API: these are all the APIs that allow you to access the IFC files and structure (remembering that BIMMS also converts IFC files into data within the SQL database tables in order to optimize access to data);
    - Resource API: are all the APIs that interact with the resources (documents) of the BIMMS system;
    - Sensors API: are all the APIs that allow the upload and access to raw data of the sensors and associated measurements;
    - Enduser API: are all the APIs that allow the management of associations between EndUser (Owner or Inhabitants) and the IFC Zones, this in order to associate the apartments with the individual end-users;
    - Occupancy API: these are all the APIs that allow access and management of the definition of the occupancy profiles of each Zone;
    - Activity API: are all the APIs that allow the management (reading and creation) of the Activities related to the Ifc Zones;
    - API Alerts: they are all the APIs that allow the management of the Alerts (reading and creation) of the Alerts linked to the Ifc Zones.
- OTSPARQL: These are the APIs dedicated to querying the data stored in Virtuoso's Graph Database. The APIs are always protected by authentication for users authorized to use the APIs of the BIMMS system;

- OTVirtuoso: These are APIs, protected by authentication, which allow to load / delete data (Triples) within the Virtuoso Graphs Database, including the creation and deletion functions of the Graphs (created by connected user);
- OTGeoLinkedData: These are APIs, protected by authentication, dedicated to the querying of geographic data stored in Virtuoso's Graph Database, associated with the resources of the BIMMS;
- OTRdf: These are APIs, protected by authentication, dedicated to loading an RDF file within Virtuoso Graph databases.

The following scenarios explain how to use the API methods to retrieve specific data from BIMMS to complete the most common workflows.

5.1 Setup, user registration and user role assignment

The BIMMS manages the users and the roles that can access to the platform. Users must be registered first to the BIMMS. The access to the BIMMS web portal is public at this URL https://bim4eeb.oneteam.it/BIMMS/. Any user can and register to the BIMMS following the login button on top right of the page and sign in. The BIMMS host the registering data of the users in an encrypted database. No user sensible data will be shared to other partners.

A Role define a set of permissions that allows users to access to specific functionalities of the BIMMS. A user can have more than one role. In the BIMMS there are seven roles: two Administration roles that are in higher hierarchy and primarily do management tasks, three roles that define specific users permissions based on typically performed tasks, and two special roles used to define the guest user and the software application user (API user). The roles that can be assigned to a user are:

- the BIMMS Administrator: is the role assigned to the user that implement the BIMMS in the Organization. This role setup the BIMMS, and can create Projects and assign the Project Administrator role to a user;
- The Project Administrator: is the role assigned to the responsible of the individual projects in the Organization. This role manages the Projects and the users assigned to a project, defining also the role for every single user;
- The Guest: is the role assigned by default to any user registered to the BIMMS before being assigned to a specific role by the Account Administrator or the Project Administrator;
- The End-User is the role assigned to the inhabitants of the apartment in a Building. The End-User may be also a Owner role if he or she owns the apartment;
- The Owner is the role assigned to the owner of one or more apartments, buildings, or any other ancillary asset in the Project;
- The Professional is the role that is appointed to give consultancy and lead Architectural, Engineering and Construction activities on the apartment, building, other ancillay asset in the project;
- The Application is the role that is assigned to the user that develop the tools that access to the BIMMS data.

In a basic workflow the users are invited to sign in the BIMMS. With the sign in process the user will

register to the BIMMS with a username, an email and a password. After registration the user receive an email to confirm the registration and his or her account is assigned to the Guest role. This role cannot access to the BIMMS data and is used to put the user in a waiting list before been approved by an Administrator user to been part of a Project. This workflow enables a first access control to the BIMMS resources: only authorized users can access to the Projects and can be enabled to specific functionalities. The Administrator users can assign a project to the user. As user can be part of the project can be assigned the roles that enable the user to perform specific tasks on data with enabled functionalities.

A user can be invited and enabled to participate to more than one project and can have different roles in different projects. A user like a designer or an engineer can be invited to participate in more than one projects: in one project he or she work as appointed designer consultant in a renovation work and in another project he or she can own the apartment where is also living. In the first example he or she will be assigned the Professional role, while in the second will be assigned the Owner and End User role.

The BIMMS allows to edit the configuration of the assignments of roles and users to give more flexibility in the future implementations of the platform in the Organizations.

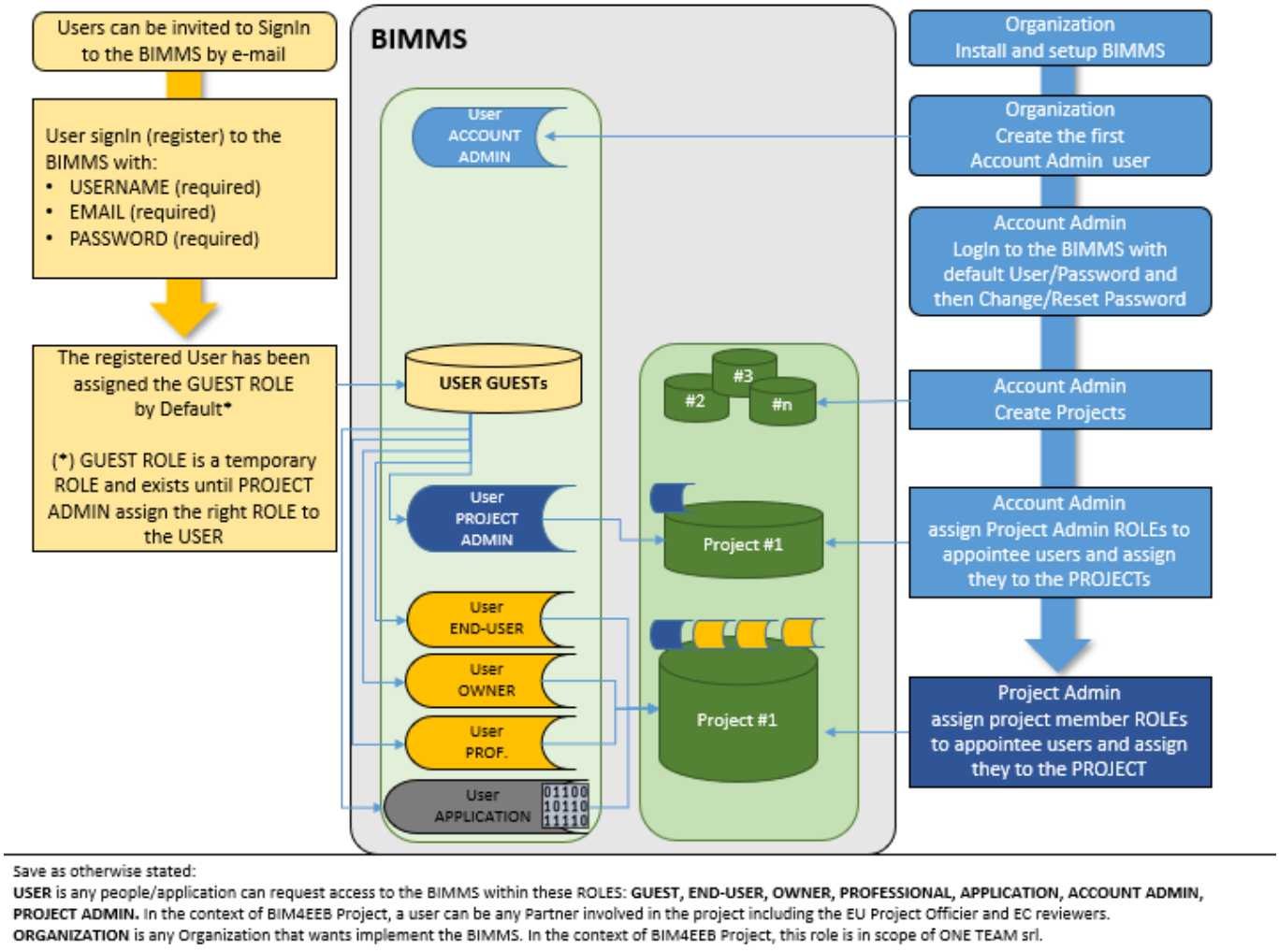
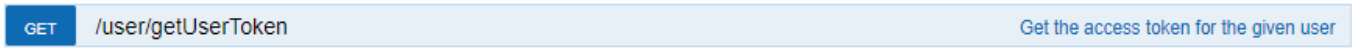


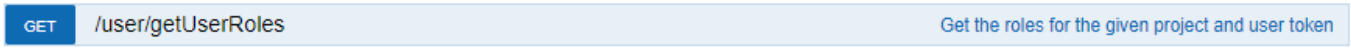
Figure 6 - On the left side the user registration workflow. On the right side the steps to setup the BIMMS, set the projects, assign the users to the project

The BIMMS API V1 allows to:

- get the access token for a given user;



- get the roles assigned to the user in a given project



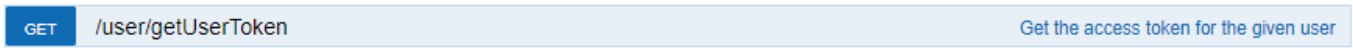
These methods are under OTMySQL API group.

5.2 Resource management

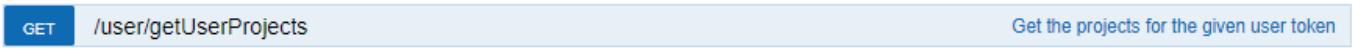
The BIMMS can have one or more Projects where will be stored the data. The project is the higher level in the data organization in the BIMMS. Usually a project contains the data related to a Building consisting in documents, logbook data, BIM Models, sensors data, etc. All the data in the projects is considered as resource: an object with a type, associated data, relationships to other resources. This concept can help also to see the resources from the point of view of the linked data paradigm.

The resources are available through different methods. These methods are different by the kind of request. For dealing with resource management, Project methods are the main entry point and represent a space where resources are hosted.

To proceed before all request is necessary to obtain an access token using

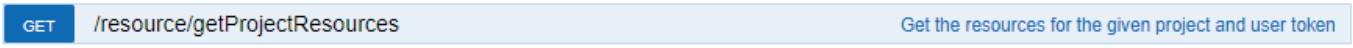


The access token is given for a user, then all subsequent requests give all the responses related to the requesting user. The access token obtained in the previous response can be used as request in the following method that will return a collection of projects for the given user

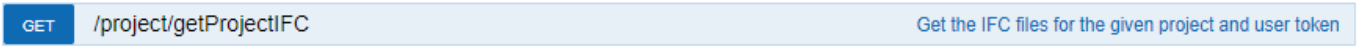


The previous two methods can be used as request to get the list of

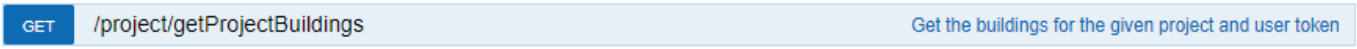
All Project Resources created or shared with the given user



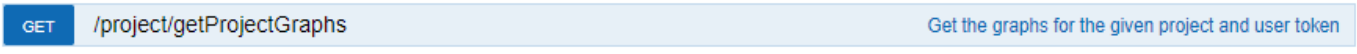
All IFC files created or shared with the given user



All the building where the given user is involved



All Linked Data Graphs created or shared with the given user



5.3 Access to 3D BIM Model IFC data

GetProjectIFC method allow to retrieve the collection of all IFC files in the Project

Every IFC file will be listed with a set of properties that are set by the user when he or she upload the IFC file in the BIMMS Resource Creation Wizard.

The most important property is the Internal File Name that is a string that join the IFC file name and the unique ID assigned to the file when it was uploaded and inserted in the BIMMS. With Internal File Name can be done most of the API request to retrieve data from the IFCs.

A set of methods are developed to work with IFC files:

getIFCFile: to retrieve IFC in binary Base64 encoding;

GET	/ifc/getIFCFile	Get the IFC file for the given file name, project and user token

5.4 How to get IFC Hierarchy and Spatial data

getIFCHierarchy: to retrieve the IFC spatial structure from the file. The spatial structure of the IFC is the nested structure of the hierarchy representing project, file, building, story and spaces. The hierarchy representation is defined and documented in the IFC documentation by building SMART (buildingSMART, 2020)

GET	/ifc/getIFCHierarchy	Get the IFC hierarchy for the given file name, project and user token

The response contains the collection of IFC items that represent the building hierarchy of the requested file. The properties of the items in the collection can be used to get further details and go through the building elements

Figure 7 - The spatial structure of the IFC file with the related Ifc Classes as defined in the building SMART documentation. On the right side, the tree view of the BIMMS BIM Viewer that shown how can be represented the IFC Hierarchy.

getIFCZones: getIFCZones is used to retrieve the IFC object relationship structure from the file. This method consider the IfcZones in the project used to group spaces, partial spaces and other zones. Zones may not be hierarchical, in contrary to the spatial structure of a project. Technically an IfcZone is grouped by the objectified relationship IfcRelAssignsToGroup where only objects of type IfcSpace, IfcZone and IfcSpatialZone are allowed as RelatedObjects (buildingSMART, 2020)

GET	/ifc/getIFCZones	Get the IFC zones for the given file name, project and user token

The response contains the collection of IFC items that represent the grouping of Zone elements from the file. The properties of the items in the collection can be used to get further details and go through the building elements. A typical response will contain:

GlobalId: the ID of the item. Can be a IfcProject, IfcZone or IfcSpace item

parentGlobalId: the ID of parent object where the item is contained or grouped. If the object have no parent is a root object and its ID will defined by the special character “#”. Usually the IfcProject is the root and it not have parents.

Name: the name of the item as defined in the IFC file.

Type: the IfcClass of the item.


```
{
  "GlobalId": "2bd35e07-b3db-43d8-81f5-6b8a283c9e56",
  "parentGlobalId": "#",
  "Name": "0001",
  "Type": "IfcProject"
},
{
  "GlobalId": "d758e715-9115-11ea-b922-005056b48219",
  "parentGlobalId": "2bd35e07-b3db-43d8-81f5-6b8a283c9e56",
  "Name": "Living room",
  "Type": "IfcZone"
},
{
  "GlobalId": "6ad8bc78-95f0-11ea-88c2-00505695f911",
  "parentGlobalId": "2bd35e07-b3db-43d8-81f5-6b8a283c9e56",
  "Name": "apartment",
  "Type": "IfcZone"
},
{
  "GlobalId": "33ef1112-fcc1-4f9a-9cbc-b7f3c3fdd3f2",
  "parentGlobalId": "33ef1112-fcc1-4f9a-9cbc-b7f3c3fdd00c",
  "Name": "3",
  "Type": "IfcSpace"
},
```


The two methods, getIFCHierarchy and getIFCZones are similar in requests but the retrieved collections in response are different:

getIFCHierarchy response shows the hierarchy saved in the IFC Files by the authoring software. It include all IFC Hierarchy from the Project Class (the root), through the Building, Storeys, Site, to the Spaces where individual construction elements can related. The relationships between the objects are based on Hierarchy.

getIFCZones response shows the relationships between Project, Zones and Spaces and represent the all Zones and Spaces defined in a Project even if they are saved in the IFC file or they were defined by other tools outside the IFC file. This data is more often used in simulation analysis and project management software’s to define the properties of work tasks and activities being done.

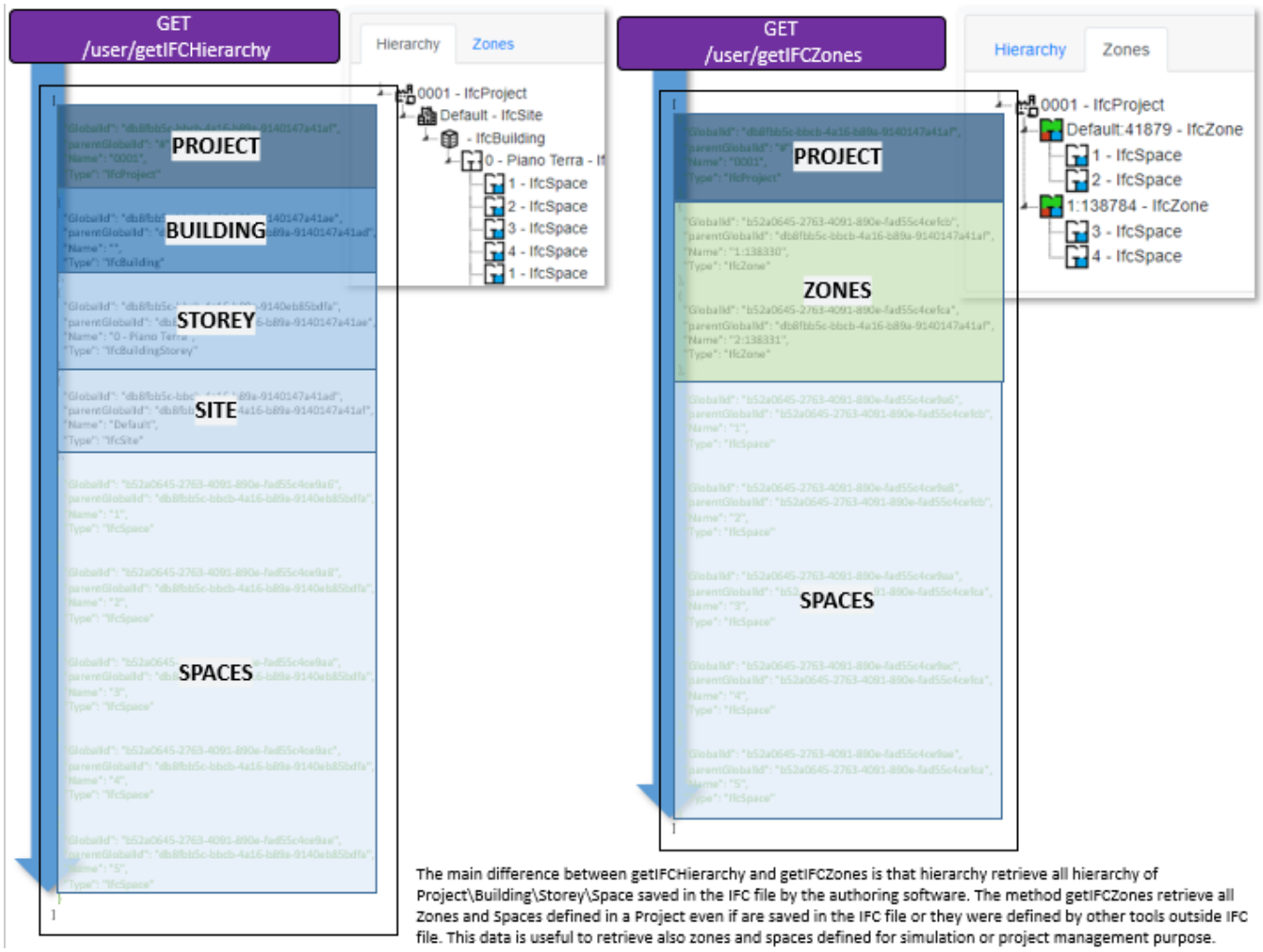


Figure 8 - Differences between getIFCHierarchy and getIFCZones methods

The typical workflow to get the Projects, IFC files, IFC Hierarchy and IFC Zones is summarized in the figure below. The application will access to the REST API and starts with requesting tokens and retrieving responses with a data collection. Zone and Hierarchy are the starting point to work with the other IFC items, and with sensors installed in the apartments.

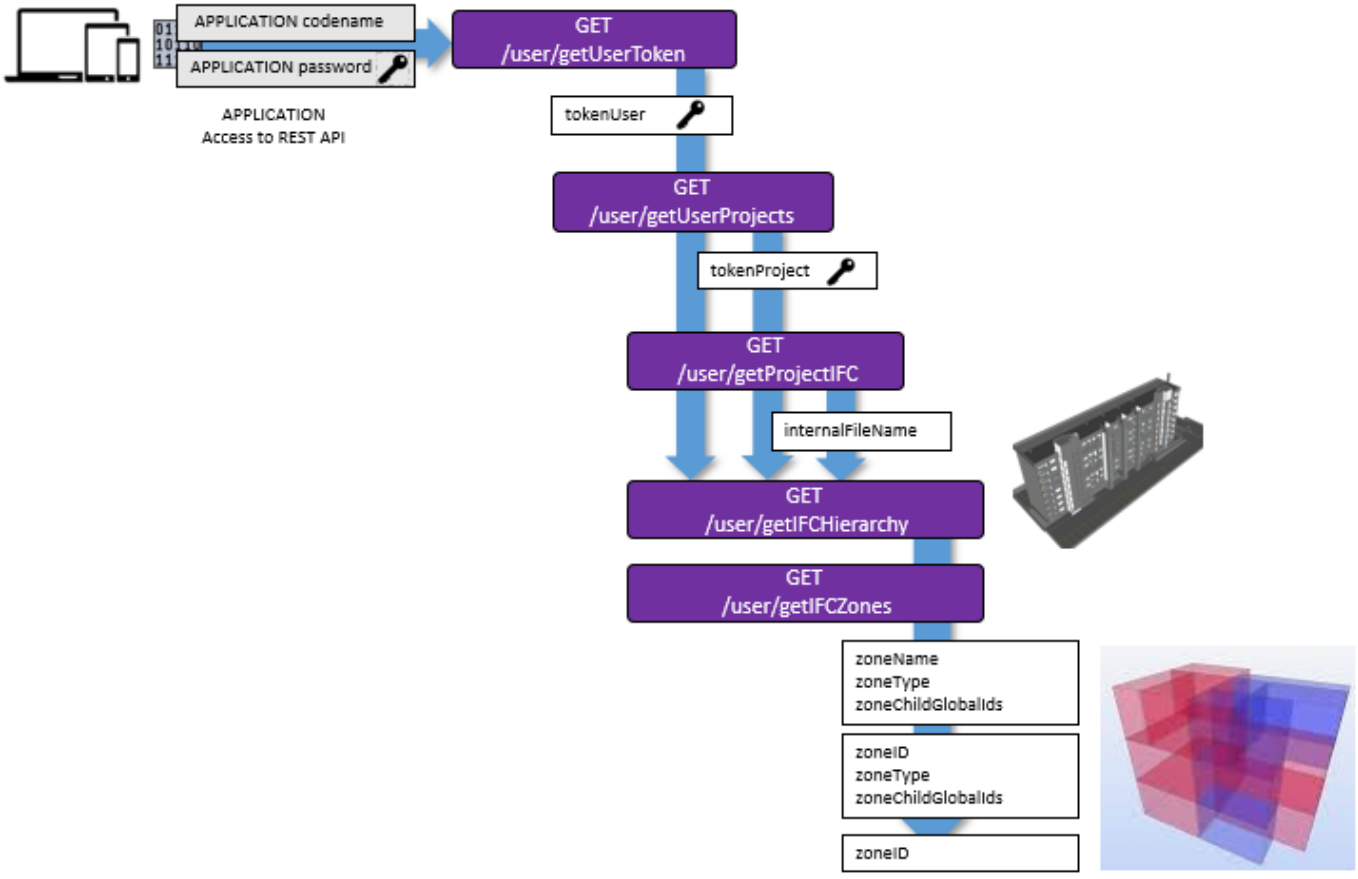


Figure 9 - Project data retrieve workflow

5.5 How to get objects from IFC Files

The IFC files can contain objects and items by various AEC disciplines starting from more common Walls, Windows, Slabs, going to electrical devices, mechanical equipment, plumbing systems.

The method getIFCSpaceObj retrieve the IFC objects in a specific location (space or zone) in a project IFC file. The method can be used to complete various tasks, just to list a few common examples can be used to get a collection of electrical devices in an apartment or in a specific room, to get a collection of the equipment in a zone of the building, to get a collection of sensors devices in a apartment, etc.

GET	/ifc/getIFCSpaceObj	Get the IFC objects in the locationID space for the given file name, project and user token

The method needs to specify the internal file name of the IFC file and the location ID where we want to retrieve objects. The location ID can be a item of IFC Hierarchy or IFC Zones in the form of a globalID as described before in the response for these methods.

The response will be a collection of items with these properties

GlobalId: the ID of the item. Can be a IfcProject, IfcZone or IfcSpace item

parentGlobalId: the ID of parent object where the item is contained or grouped. If the object have no parent

is a root object and its ID will defined by the special character “#”.

Name: the name of the item as defined in the IFC file.

Type: the IfcClass of the item.

The following is an example of response


```
"GlobalId": "b90bb6a4-eba6-4add-8779-5b9dbc956c4d",
"parentGlobalId": "f3de0c5d-e83c-48e7-880f-359673af2c3a",
"Name": "Heater:3:2122473",
"Type": "IfcBuildingElementProxy"
```


All the values of the properties listed in the response are the data compiled by the author of the IFC file in the authoring application. Should be take attention to the type value assigned to the objects during the export of the IFC files. Often items are exported as generic element and the IFC exporter assign the IfcBuildingElementProxy class instead a more specific and correct classification. In the example above The object is an Heater as can be easy recognized by its name, but the Type is misleading and should be an IfcSpaceHeaterType. IFC item classification is more important to get the right information through the BIM process and should be checked to avoid mistakes.

The method take into account that this kind of issues are more common and retrieve all items regardless the Type defined in the IFC files.

5.6 How to manage the zone assignment to the end-users

An important feature supported by the APIs is the ability to retrieve the data associated to the End users, like the inhabitants, without sharing any confidential information. End users data have to be secured and anonymized to avoid to share any personal information. The applications shall retrieve data for their users without sharing any value that can identify directly or indirectly the inhabitant. This kind of requirement has been set by the Privacy Protection policies applied in the BIM4EEB project, that must comply with GDPR regulations. To satisfy this requirement were developed the authentication workflow and the zone assignment workflow. The authentication workflow is described in the chapter 4.2 and distinguish the two kind of accesses for the users and the applications, and the zone assignment management workflow is described in this chapter, detailing how to assign, delete and retrieve zones related to an end-user.

To better understand how to use these API methods, a little introduction to the workflows will follow. The typical use case is represented by an inhabitant end user that is already registered to the BIMMS. In this scenario he or she also use an external application (a mobile app or a webapp from a third party) to get information about his or her apartment and the building where he or she lives. The BIMMS contains the mapping between this user and his or her associated building zones, like the apartment with its rooms and other ancillary rooms in use by the inhabitant. This mapping can be done knowing the inhabitant personal data and can be done only in the BIMMS by an authorized party: usually can be done by the inhabitant itself, the owner or by the project administrator depending on the settings.

The external application can access only to the information retrievable with the zones associated to the inhabitant. The inhabitant gives the consent to access to his or her building zones through an applicationToken set in the application by itself. The applicationToken is a secret passcode generated by the BIMMS and known only by the inhabitant when access to his or her BIMMS personal profile in the web portal. Copying and typing the passcode in the application confirm the access of the external application.

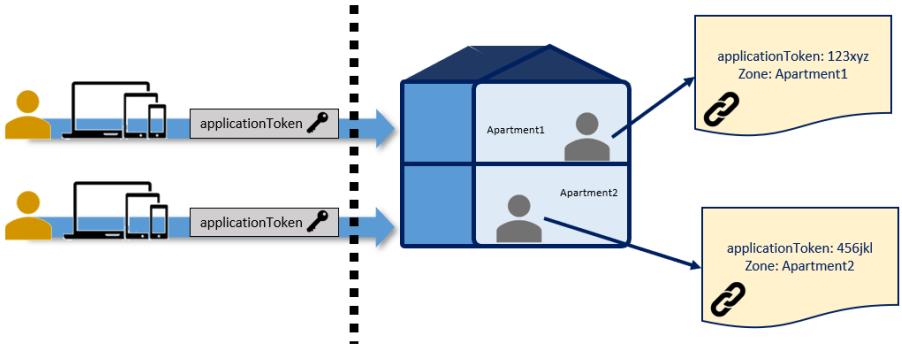


Figure 10 - IFC Zones and applicationToken

5.6.1 How to assign zones to an end user

The method setEnduserZone allow to map the end user with a zone in a project.

POST /enduser/setEnduserZone Connect end user to IFC zone for the given file name, project and user token

This method needs as request:

tokenUser: the access token of the application;

tokenProject: the token of the project;

filename: the unique file name of the IFC that contain the zones;

enduserToken: the applicationToken;

zoneGUID: the zone GlobalID.

The response of a successful request will be the name of the IFC zone set as result of the mapping.

Response Body

```
"1"
```

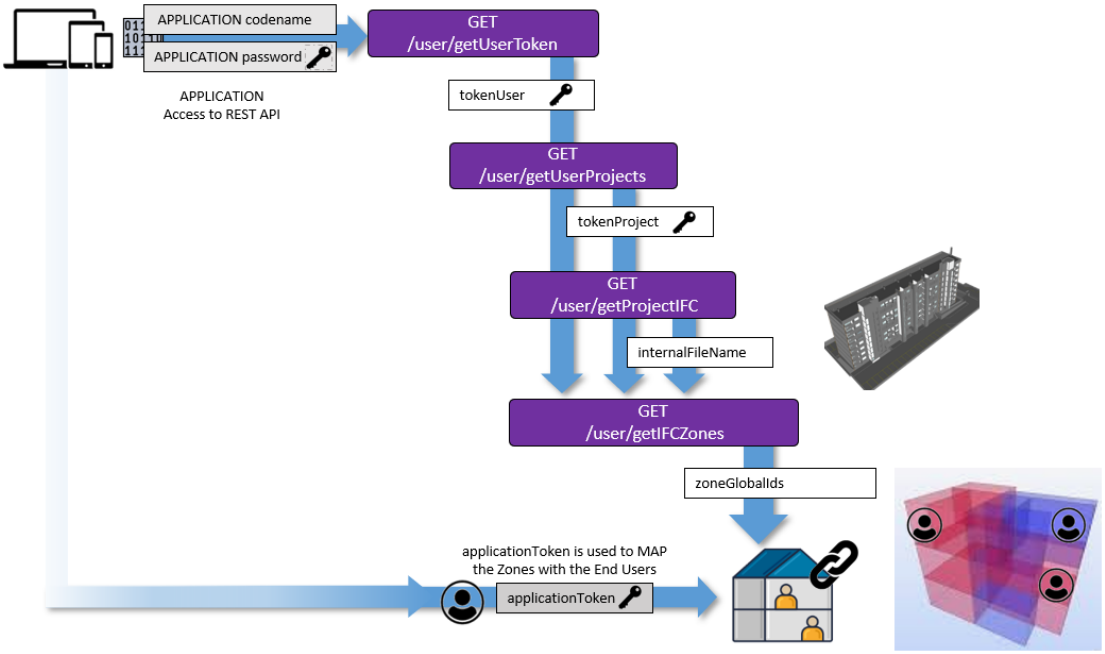


Figure 11 - Retrieving IFC Zones to set end-user assignment

5.6.2 How to delete zones to an end user

The method delEnduserZOne allow to delete the end user from a zone in a project

DELETE	/enduser/delEnduserZone	Disconnect end user from IFC zone for the given file name, project and user token

This method needs as request:

tokenUser: the access token of the application;

tokenProject: the token of the project;

filename: the unique file name of the IFC that contain the zones;

enduserToken: the applicationToken;

zoneGUID: the zone GlobalID.

The response of a successful request will be the name of the IFC zone deleted from mapping.

Response Body

```
"1"
```

5.6.3 How to retrieve the zones associated to an end user

The external application can retrieve the user data associated to the building zones using the method getEndusersZones

GET	/enduser/getEndusersZones	Get end users connected to IFC zones for the given file name, project and user token

This method needs as request the access token of the application, the token of the project and the unique file name related to the user. All parameters in the request do not ask any end user personal data. The response is a collection of locations with ID, name end APPTOKEN:

APPTOKEN: the applicationToken of an user;

locationID: ID of a Zone or a Space in the project

Name: The name of the Zone or the Space in the project

The following is an example of response:

```
{
  "APPTOKEN": "jc1234!",
  "locationID": "d758e715-9115-11ea-b922-005056b48219",
  "Name": "Living room"
}
```

The APPTOKEN parameter value of the response will be used in the application to filter the collection and then show only the end user zones associated to the user

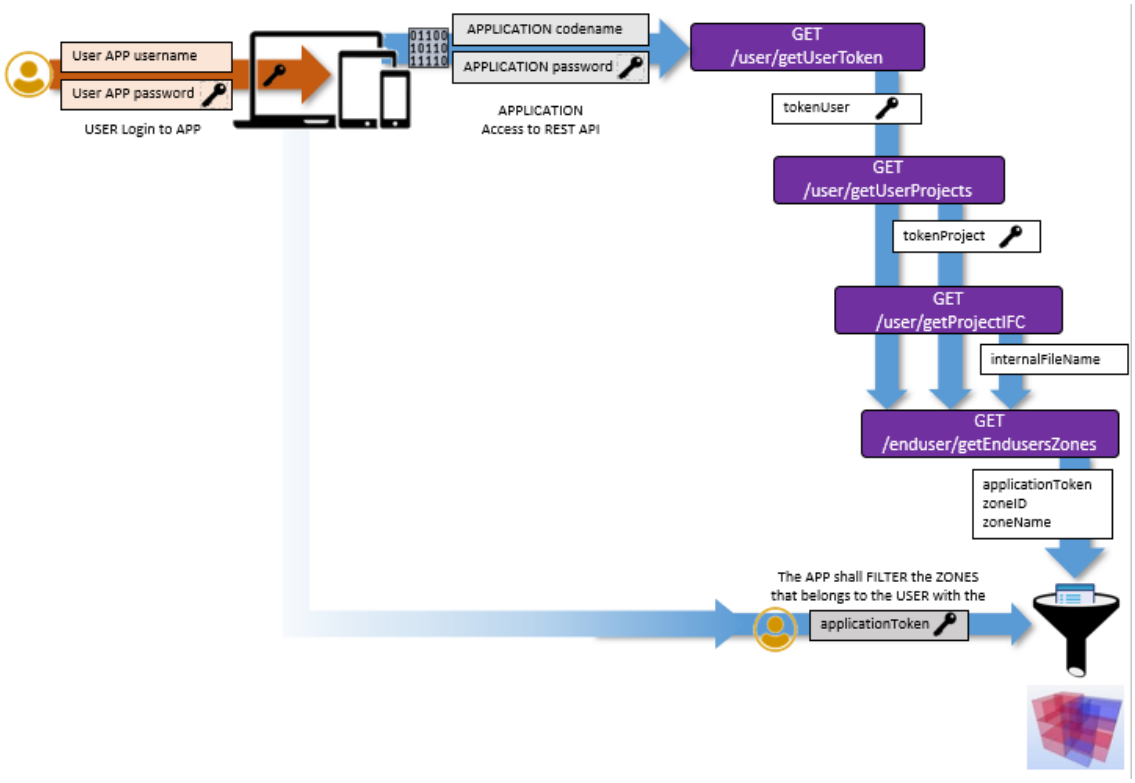


Figure 12 - getEndusersZones workflow

5.7 How to retrieve the projects associated to an end user

Using getEnduserProjects an application is able to retrieve the collection of the projects in the BIMMS

where an end user have access to.

The request needs just the accessToken of the application and the applicationToken set by the end user in the application.

The response gets a collection of one or more projects with parameters that give some details about them. Two Booleans parameters, inhabitant and owner give the status of the user in the project. If the value is set to 0 (zero) the end user have not associated the role, otherwise if the value is set to 1 (one) the end user have associated role. In the following example, the two parameters are set to 1 and this means that the end user is the inhabitant and also the owner of the building in the project “test”.


```
{
  "TOKEN": "4d9911cf-102e-11ea-9c7f-005056920005",
  "NAME": "test",
  "COMPANY": "One Team srl",
  "CITY": "Milano",
  "STATUS": 1,
  "INHABITANT": 1,
  "OWNER": 1,
  "ORIGINAL_NAME": "test"
},
```

5.8 How to manage IoT data, Devices and Measurements

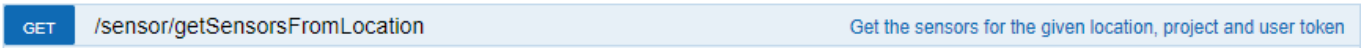
The sensors usually are physically installed inside the building rooms to measure the internal conditions and immediately outside to measure the external conditions. As described before the hierarchy of the building can be decomposed in a relationship that ends with Zones and Spaces. These two items, defined in the IFC Project files can contain devices like sensors. Finally the sensor device have as child relationship its measurements. All of these items are collections of data: a building more often have more than one Zones, Zones can have more than one sensor, and a sensor usually have more than one measure. Sensor can also be defined by type of measurement like temperature, CO2, and other air component mixture or type of sensing. For every type of measurement, will be a collection of records of streaming data in a range of time.

The BIMMS APIs have a set of methods to manage sensors devices and their related measurements. These methods can add data to the BIMMS as result of setup of the sensor in a location and the streaming of data, and then retrieve this data related to a particular location, range of time, or type of measurement.

The starting point to work with sensors is to get the access token of the application, the project token of the project, and then retrieve IFC Hierarchy and IFC Zones where the sensor is located or will be located.

5.8.1 How to get the list of sensors in a location

getSensorsFromLocation is the method used to retrieve the sensor in given location of a project.

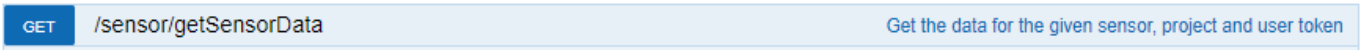


The request needs an access token, a token project and a location ID. The response is a collection of sensors defined by and ID code and a type that can describe the sensor.


```
{
  "code": "cf0ac67e-8483-4a6e-b5d7-1feaf361449b",
  "type": "Aeon Multisensor 6"
},
{
  "code": "fe324291-ff2e-49c4-b507-b0e782a90412",
  "type": "Z-Wave Metering Switch"
}
```


5.8.2 How to get the measurements from a sensor

The getSensorData method retrieve the measurements from a sensors in a range of time



Using the ID sensor code retrieved before and specifying a range date to limit the recorded streaming, the response is a collection of measurements with:

Date: the time record of the measure;

Value: the value of the measurement;

UM: the unit of measure of the measurement;

Type: the sensed measure.

```
{
  "date": "2012-10-24T12:09:00",
  "value": 23.2,
  "UM": "C",
  "type": "Temperature"
},
{
  "date": "2012-10-24T12:08:00",
  "value": 23.12,
  "UM": "C",
  "type": "Temperature"
},
{
  "date": "2012-10-24T12:07:00",
  "value": 23.09090909,
  "UM": "C",
  "type": "Temperature"
},
```


5.8.3 How to set sensors devices in a project

There are two methods to set sensors devices in a project. The setProjectSensors method allow to set a collection of sensors of different type and different location in an only once import.

POST	/sensor/setProjectSensors	Import CSV file of sensors data for the given project and user token

This method read the sensors settings form a CSV file provided in the request. The CSV file must be an encoded Base64 string formatted without headers and with value fields as follow:

Code: the sensor ID code;

Location: the location ID where the sensor is positioned, the location must exists before import;

Type: the sensor type.

In another scenario the sensors can to be set by an external application capable to send a request by its APIs and set the parameters in the setProjectSensor method. Usually sensors works with a collecting Hub that have functionalities to read, show and send sensors data to other tools and applications.

POST	/sensor/setProjectSensor	Import sensor data for the given project and user token

The request needs a ID code of the sensor, a location name of the sensor and the sensor type. Location name is the name of the Zone or Room in the building and must be exist before setting the sensor. Sensor type can be a description or the type of measurement done by the sensor.

5.8.4 How to delete sensors devices in a project

Sensors can be deleted from a project using two methods. As described in the set sensor devices in chapter 5.8.3 the methods consider If the request come from another external application as collecting tool or a bulk editing of multiple sensors in multiple locations. The requests for the two methods are the

same used for create and set sensors

POST	/sensor/delProjectSensors	Delete sensors data for the given CSV file, project and user token
POST	/sensor/delProjectSensor	Delete sensor data for the given sensor, project and user token

5.8.5 How to import measurements from a sensor

There are two methods to import measurements data for the given sensor depending if it is a import data from a CSV file or a direct streaming from a sensor.

The setSensorMeasures allow to import a collection of measurements from a CSV file. This method is useful for bulk import of data and to import a collection of measurements from different sensors and from different locations. As request is necessary to provide an encoded Base64 file with the content.

POST	/sensor/setSensorsMeasures	Import CSV file of sensors measurements data for the given project and user token

The CSV file must be formatted without headers and with value field as follow:

Code: The sensor ID code. Sensor must exists before import;

Date: Date of measurement in yyyy-mm-dd hh-mm format;

Value: the value of measurement without UM;

Type: the measured quantity.

The setSensorMeasure allow to get the stream from a sensor. In this scenario the stream usually is coming from a sensors hub that collect the data and will be sent an endpoint. The sensor hub will be set to post the data as needed.

POST	/sensor/setSensorMeasure	Import sensor measurement data for the given sensor, project and user token

The request needs the ID of the sensor, the date of measurement, the value of measurement, and the type of measurement.

5.9 How to manage occupancy data, activities, alerts

Occupancy, activities status and alerts data can be used by the users to inform about the building's usage to manage the renovation and the operational phase of the building. Owner and inhabitants can exchange information about building operational characteristics, to monitor the system performance of HVAC and lighting systems. Moreover activities can be set and retrieved to get all users informed about the status of the renovation works with also the possibility to handle events messages and manage safety and security alerts.

5.9.1 Occupancy

The Occupancy parameter get a valuable information about the building's usage. The parameter is used in operating and occupancy schedules to provide information about occupancy assumptions for different building types used during energy analysis.

The method getOccupancyFromLocation allows to retrieve the occupancy data in a specific location.

GET	/occupancy/getOccupancyFromLocation	Get the occupancy data for the given location, project and user token

As this information will be retrieved it can be used to be compared with the assumptions in design phase or can be triggered an alert or an action to configure the equipment or improve the resource usage in the

location. The response is a collection of parameters that describe:

zoneMaxOccupancy: the max number of occupants in a zone;

scheduleID: unique ID of the schedule;

scheduleName: the name for the schedule (e.g. week, weekday, etc);

annotationEntity: the user that created the schedule

dataID: the unique ID of schedule data

date: when the location is occupied

value: is the percentage of occupation in a time (refers to date parameter)


```
{
  "occupancyID": 5,
  "zoneMaxOccupancy": 5,
  "scheduleID": 10,
  "scheduleName": "my second schedule",
  "annotationEntity": "annotation entity test value",
  "scheduleDate": "2020-06-18T22:10:44",
  "dataID": 22,
  "date": "2020-06-20T00:01:00",
  "value": 100
}
```


The following methods can be used to define the occupancy schedules.

POST	/occupancy/setOccupancy	Create occupancy data for the given location, project and user token
POST	/occupancy/setOccupancySchedule	Create occupancy schedule for the given occupacy data, project and user token
POST	/occupancy/setOccupancyScheduleData	Create occupancy schedule data for the given schedule, project and user token

5.9.2 Activities

Activities can be used to set and retrieve activities planned, ongoing and finished in a specific location. These methods can be used by project management tools to manage the tasks performed in the building and as starting point to set alerts and event messages

GetActivityFromLocation can retrieve all activities set in a location in a range of time

GET	/activity/getActivityFromLocation	Get the activity data for the given location, project and user token

SetActivity can set an activity in a location with a range of duration.

POST	/activity/setActivity	Create activity data for the given location, project and user token

The request parameters include details about the activity:

activityName: the name of the new activity being created;

activityStartTime: the activity start date in yyyy-mm-dd hh-mm format;

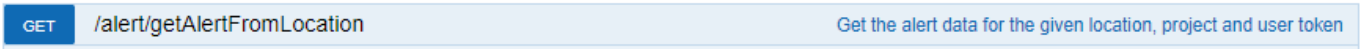
activityEndTime: the activity end date in yyyy-mm-dd hh-mm format;

activityDuration: the activity duration in days or fraction of day.

5.9.3 Alerts

Alerts can be used to handle events messages and manage safety and security alerts.

The method getAlertFromLocation allow to retrieve the alert data set for a given location and in a range of time



The method setAlert allow to create an alert in the given location.

The request parameters include details about the alert:

alertType: type of alert being created;

alertTopic: alert topic;

alertSubject: alert subject;

alertDescription: alert description;

alertTime: alert date in yyyy-mm-dd hh-mm format;

alertOwner: alert owner;

importance : alert priority.

5.10 How to manage GIS data

The BIMMS manages the GIS data used to geo locate resources and connect them to the GIS data management tools.
The OTGeoLinkedData method retrieves a collection of GIS data nodes with latitude and longitude of the site where the resource is located. This method retrieves data from linked data charts enabled in the project.
The data are stored in the "linked data" graph of the BIMMS project (example
http://bim4eeb.oneteam.it:8890/bim4eeb-**PROJECTNAME**-linkeddata#
http://bim4eeb.oneteam.it:8890/bim4eeb-**test**-linkeddata#
)
The ontology used for cartographic information is the Geo ontology - WGS84
https://www.w3.org/2003/01/geo/

5.10.1 How to get Geo Linked data

The query in the graph is performed by the method to obtain all the geo points: Point (https://www.w3.org/2003/01/geo/wgs84_pos#Point)
The results are returned by the method in J SON (http://vos.openlinksw.com/owiki/wiki/VOS/VOSSparqlProtocol#Additional%20HTTP%20Response%20Formats%20--%20SELECT).

SPARQL example used to get geo: Point

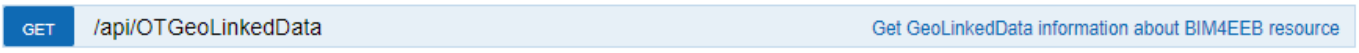
```
SELECT *   WHERE {
?s ?p ?o .
?s a geo:Point .
```

} order by ?s

The SPARQL results

subject	predicate	object
http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/7646cc0a-e293-40f1-800c-45d473b45e5c#this	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2003/01/geo/wgs84_pos#Point
http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/7646cc0a-e293-40f1-800c-45d473b45e5c#this	http://www.w3.org/2000/01/rdf-schema#label	"Milan"
http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/7646cc0a-e293-40f1-800c-45d473b45e5c#this	http://www.w3.org/2003/01/geo/wgs84_pos#lat_long	"9.213289620651928,45.45776033119512"
http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/7646cc0a-e293-40f1-800c-45d473b45e5c#this	http://www.w3.org/2003/01/geo/wgs84_pos#lat	"9.213289620651928"
http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/7646cc0a-e293-40f1-800c-45d473b45e5c#this	http://www.w3.org/2003/01/geo/wgs84_pos#long	"45.45776033119512"

Figure 13 - Example SPARQL geo:point data



The request needs an access token, a token project and the “linked data” graph of the project.

Parameters

Parameter	Value	Description	Parameter Type	Data Type
TokenUser	9400dcd7-f0f3-11e9-87b6-[redacted]	BIM4EEB token user, insert tokenUser retrieved by user/getUserToken method	query	string
TokenProject	4d9911cf-102e-11ea-9c7f-[redacted]	BIM4EEB token project,insert tokenProject retrieved by user/getUserProjects method	query	string
graph	http://bim4eeb.oneteam.it:8890/bim4eeb-test-linkec	optional sparql graph, insert destination graph example http://bim4eeb.oneteam.it:8890/bim4eeb-test-linkeddata#"	query	string

Figure 14 - Example input parameters OTGeoLinkedData


```
"\n{ \"head\": { \"link\": [], \"vars\": [\"s\", \"p\", \"o\"] },\n  \"results\": { \"distinct\": false, \"ordered\": true, \"bindings\": [\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://
bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/443fbf96-90ea-4e46-a769-4c9fb3ab1903#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/1999/02/
22-rdf-syntax-ns#type\" }\t, \"o\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#Point\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.
it:8890/bim4eeb-test-resources/v_geopoint/ID/443fbf96-90ea-4e46-a769-4c9fb3ab1903#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2000/01/rdf-schema#label\" }\t, \"o\": {
\"type\": \"literal\", \"value\": \"Mokba\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/
443fbf96-90ea-4e46-a769-4c9fb3ab1903#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#lat_long\" }\t, \"o\": { \"type\": \"literal\", \"value\": \"37.
77636718749999,55.724355519718835\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/
443fbf96-90ea-4e46-a769-4c9fb3ab1903#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#lat\" }\t, \"o\": { \"type\": \"literal\", \"value\": \"37.
77636718749999\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/443fbf96-90ea-4e46-a769-4c9fb3ab1903#this\" }\t, \"p\": {
\"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#long\" }\t, \"o\": { \"type\": \"literal\", \"value\": \"55.724335519718835\" }},\n    { \"s\": { \"type\": \"uri\",
\"value\": \"http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/6f06350e-0fce-49e0-8642-0a74dd3b9f5b#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/1999/02/
22-rdf-syntax-ns#type\" }\t, \"o\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#Point\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.
it:8890/bim4eeb-test-resources/v_geopoint/ID/6f06350e-0fce-49e0-8642-0a74dd3b9f5b#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2000/01/rdf-schema#label\" }\t, \"o\": {
\"type\": \"literal\", \"value\": \"\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/
6f06350e-0fce-49e0-8642-0a74dd3b9f5b#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#lat_long\" }\t, \"o\": { \"type\": \"literal\", \"value\": \"56.
101562499999986,43.484779332187 63\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/
6f06350e-0fce-49e0-8642-0a74dd3b9f5b#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#lat\" }\t, \"o\": { \"type\": \"literal\", \"value\": \"56.
101562499999986\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/6f06350e-0fce-49e0-8642-0a74dd3b9f5b#this\" }\t, \"p\": {
\"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#long\" }\t, \"o\": { \"type\": \"literal\", \"value\": \"43.484779332187 63\" }},\n    { \"s\": { \"type\": \"uri\", \"value\":
\"http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/7646cc0a-e293-40f1-800c-45d473b45e5c#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/1999/02/
22-rdf-syntax-ns#type\" }\t, \"o\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#Point\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.
it:8890/bim4eeb-test-resources/v_geopoint/ID/7646cc0a-e293-40f1-800c-45d473b45e5c#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2000/01/rdf-schema#label\" }\t, \"o\": {
\"type\": \"literal\", \"value\": \"Milano\" }},\n    { \"s\": { \"type\": \"uri\", \"value\": \"http://bim4eeb.oneteam.it:8890/bim4eeb-test-resources/v_geopoint/ID/
7646cc0a-e293-40f1-800c-45d473b45e5c#this\" }\t, \"p\": { \"type\": \"uri\", \"value\": \"http://www.w3.org/2003/01/geo/wgs84_pos#lat_long\" }\t, \"o\": { \"type\": \"literal\", \"value\": \"9.
```


Figure 15 - Example JSON response

It is possible to represent the results in map for example using the Open Layer libraries.

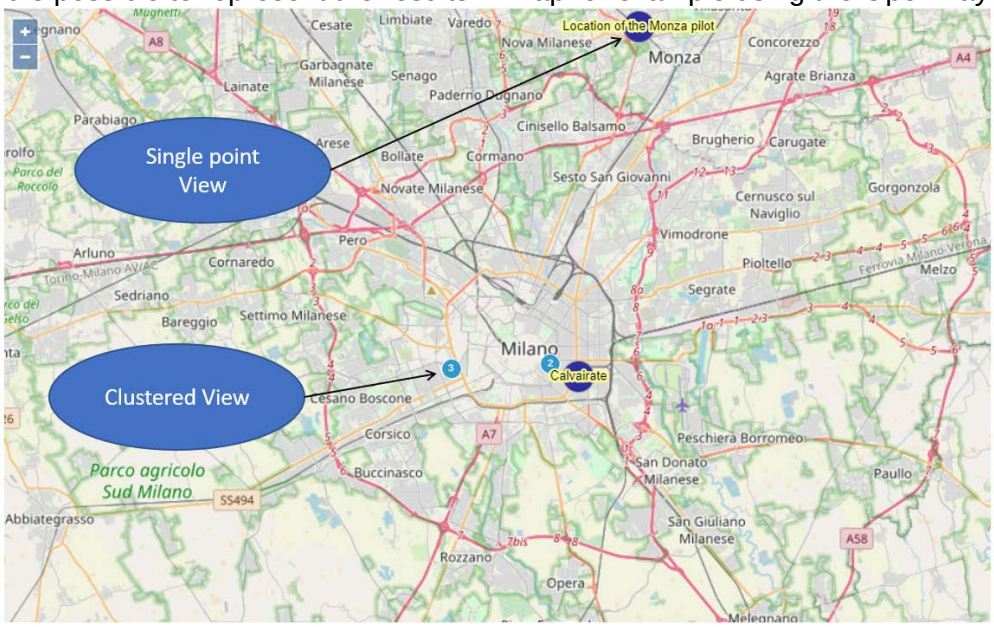


Figure 16 - Map View with Open Layer libraries

5.11 Linked Data and Graph management

This paragraph illustrates the APIs available for managing project graphs and linked data.

API exposes OTRdf to load RDF files into the graph through the sparql-graph-crud-auth Virtuoso endpoint (http://vos.openlinksw.com/owiki/wiki/VOS/VirtGraphUpdateProtocolvirtuoso).

OTVirtuoso allows the loading / deleting of triple lists and the cancellation of the graph through sparql-auth Virtuoso endpoint.

(http://vos.openlinksw.com/owiki/wiki/VOS/VirtTipsAndTricksGuideSPARQLEndpoints)

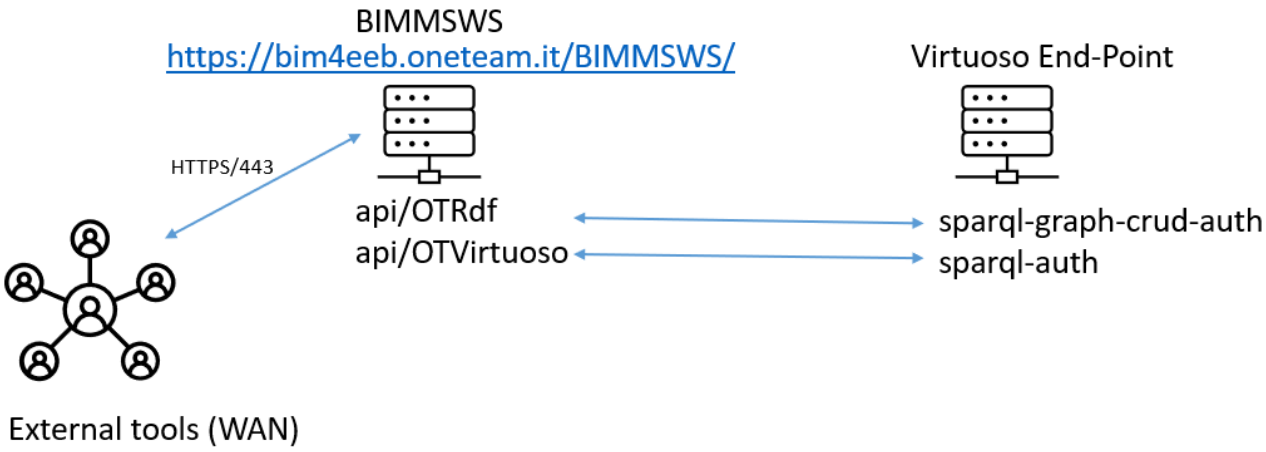


Figure 17 - Linked Data endpoint

5.11.1 How to upload RDF file

Upload RDF base64 file into Graph (Virtuoso Database)

The request needs an access token, a token project, a destination graph and a base64 rdf/ttl file.

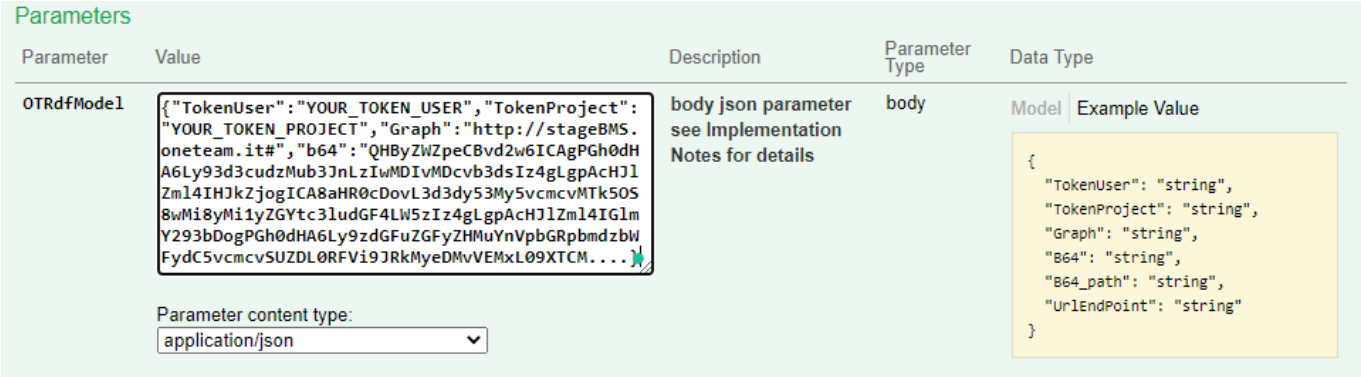


Figure 18 - Example Input OTRdf

5.11.2 How to insert triples

Insert triple list into virtuoso graph defined by Graph parameter, for the given project and user token.

POST	/api/OTVirtuoso	Insert triple into virtuoso graph

In the "TripleList" parameter, insert the triples to be loaded separated by dot.

5.11.3 How to delete triples

Delete triples in TripleList parameter from virtuoso Graph defined in Graph parameter, for the given project and user token

DELETE	/api/OTVirtuoso	Delete triple from virtuoso Graph

5.11.4 How to drop graph

Only the owner (creator) can delete the graph. To delete the graph, use the DropSilentGraph method.

The graph is also removed from the BIMMS portal

DELETE	/api/OTVirtuoso/DropSilentGraph	removes the graph from BIMMS. Only owner (creator) can delete graph

5.11.5 How to query graph

To query the graph, use the getSPARQLData method.

This method gets SPARQL data from graph name and query, with limit and result type (Table, Raw) format.

POST	/sparql/getSPARQLData	Get SPARQL data from graph name and query, with limit and result type (Table, Raw) format

Example:

```
{
"graphURI":"http://bim4eeb.oneteam.it:8890/bim4eeb-test-iotdata#",
"query":"SELECT * from <http://bim4eeb.oneteam.it:8890/bim4eeb-test-iotdata#> WHERE {?s ?p <https://w3id.org/saref#Measurement>}",
"limit":"1",
"resultType":"R"
}
```

6 Conclusions

The guidelines for the integration of new tools in the BIMMS has been delivered starting from the requirements defined in the previous tasks of the WP4 and the first outcomes of the activities of the WP2 and WP3. The guidelines are consistent with the ongoing work in the parallel tasks in WP4 and in the other WPs by the other developer partners for tools developments. The configuration tested satisfies both the tool integration requirements for the preliminary toolset of testing purposes in M18 and for the final toolset of validation purposes in M24.

The improvement of all the tools that are part of the BIM4EEB Toolkit is expected in the coming months.

Working in an integrated environment with the project partners, the development of the BIMMS system will allow to expand the possibilities of integration and interoperability with the project Tools. Furthermore, new developments will define the complete guidelines for the access of new software platforms to the CDE of project.

In this version of the system, in M18, all the basic functions for authentication and data exchange with external systems have already been developed and all integration development and testing activities have already been carried out by the partners for the functionalities already implemented in their tools.

In the coming months, the development of the integration processes will be completed according to the needs and functional requirements indicated by the project partners.

7 Bibliography

buildingSMART, 2020. *ifcZone.* [Online]
Available at:
http://standards.buildingsmart.org/MVD/RELEASE/IFC4/ADD2_TC1/RV1_2/HTML/schema/ifcproductextension/lexical/ifczone.htm
[visited on May 2020].
buildingSMART, 2020. *Industry Foundation Classes 4.0.2.1.* [Online]
Available at: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/
[visited on May 2020].
Fielding, R. T., 2000. Chapter 5: Representational State Transfer (REST). In: *Architectural Styles and the Design of Network-based Software Architectures.* Irvine: University of California.
Swagger, 2020. *Swagger UI.* [Online]
Available at: https://swagger.io/tools/swagger-ui/
[visited on May 2020].