



BIMy Project:

D3.2 Development: overview of current developments

Document metadata

Date 2020-11-12
Status Draft
Version 2.01
Authors Osman Kumas - NETAS
 Stijn Goedertier – GIM
 Sergio Ristagno – LetsBuild
 Gözdenur Yeşilyurt - NETAS
 Steven Smolders - GIM

Coordinator Osman Kumas - NETAS
Reviewed by

Version history

Version	Date	Author	Change
0.01	2019-04-09	OKU	Introduction and scope
0.02	2019-05-16	SGO	Document bimy-docker
0.03	2019-06-17	SGO	Document bimy-data-manager, bimy-bimsurfer, bimy-web-viewer, bimy-data-transformer
0.04	2019-12-23	SGO	Document bimy-data-model
0.05	2019-12-2020	SRI	Document BCF Manager
2.00	2020--11 23	GY	Updates for Version 2
2.01	2020-11-26	GY	Documentation MinIO Client, MinIO Client
	2021-03-16	SS	BIMy integrated BIM/GIS Web Viewer and Data transformers

2.02	2021-03-22	GY	Finalization of BIMy data manager and BIMy on Cloud section
2.03	2021-03-23	OK	Finalization of Deliverable

1 Table of Contents

1	Table of Contents.....	3
1	Introduction of BIMy Platform	4
2	BIMy Platform on Azure	5
3	BIMserver.....	10
4	Fire Prevention App	11
5	BIMy Data Model.....	15
	Rationale	15
	Conceptual model BIMy Data Model	15
	XML Schema encoding	17
	Linked Data Vocabulary.....	17
6	BIMy REST API specs	18
7	BIMy Data Manager.....	19
8	BIMy Data Transformer	22
9	BIMy Web Viewer	24
10	BIMy BCF Manager	26
10.1	Getting started	26
10.1.1	Running the application.....	26
10.2	Security: OAuth2 authentication.....	27
10.3	Functionalities	27
10.3.1	Version's service	28
10.3.2	Project service	28
10.3.3	Topic service	28
10.3.4	File service	28
10.3.5	Comment service.....	29
10.3.6	Viewpoint service	29
10.3.7	Document reference service	29
10.3.8	Document service.....	29
10.3.9	BCF Export.....	30

1 Introduction of BIMy Platform

The BIMy platform is a multi-layered system created for the realization of scenarios created within the scope of the project. In the realization of these scenarios, each partner implements different components and carries out the tasks of their work packages according to the identified user scenarios. Collaboration and teamwork between partners have been the main motivation in the realization of the implementations.

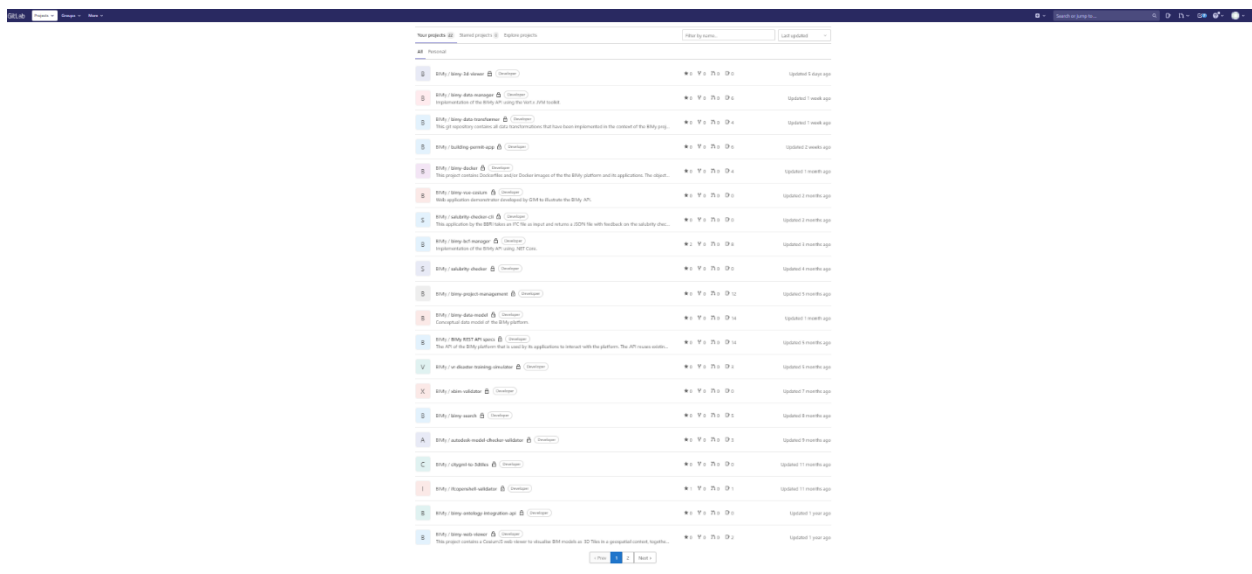


Figure 1: Gitlab Repository of BIMy Platform

Figure 1 shows the repos of all applications developed on the BIMy platform on Gitlab.

2 BIMy Platform on Azure

The project consortium has decided that it was technically feasible to deploy the BIMy platform to common cloud platforms such as Microsoft Azure in 2021. For this reason, the applications developed were built on azure as dockerized containers, and a UI was designed to enable the platform to be used by the users. In the following section, the visuals of this UI were added with their explanations.

Name	Subscription	Resource group	Location	Status	Operating system	Size	Public IP address	Disks
bimy-linux-vm-01	Microsoft Azure	bimy-rg	West Europe	Running	Linux	Standard_D2s_v4	51.137.109.43	2
bimy-test-vm	Microsoft Azure	bimy-test-rg	West Europe	Running	Windows	Standard_D2s_v3	51.124.94.210	1
bimy-windows-vm-01	Microsoft Azure	bimy-rg	West Europe	Running	Windows	Standard_D2s_v4	104.40.212.148	2
bimy-linux-vm-02	Microsoft Azure	bimy-rg	West Europe	Running	Linux	Standard_D2s_v4	52.232.120.210	1
integrationvm	Microsoft Azure	dev-stdl	West Europe	Running	Windows	Standard_D4s_v3	168.63.120.47	1
keycloak-standalone	Microsoft Azure	CRITICALCHAINS	Germany West Central	Running	Linux	Standard_D51_v2	20.52.153.77	1
stdl	Microsoft Azure	stdl_group	Germany West Central	Running	Linux	Standard_D51_v2	20.52.33.90	1

Figure 2: BIMy Platform on Azure

The BIMy platform and API have been finalized to support the final batch of integrated demonstrations. Bimy-data-manager, MinIO storage server, PostgreSQL database and BIMy Web

UI are dockerized and all run on Microsoft Azure VMs. They are placed behind a NGINX reverse proxy which directs the client requests to appropriate services.

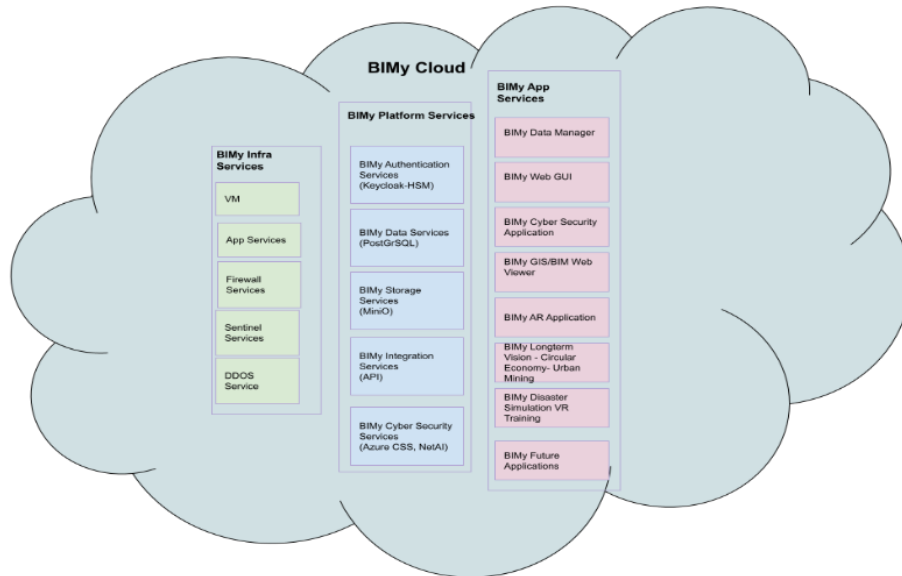


Figure 3: BIMy platform topology

The BIMy platform topology given in the image above consists of 3 main groups and these headings:

Infra Services: Cloud components for platform to work

Platform services: Platform applications necessary for 3rd party applications to run and these applications provide the necessary platform services for the development of BIMy applications and 3rd party applications.

App Services: These are use case and demonstrations applications developed within the BIMy project.

A User Interface (UI) has been developed to present the applications included in the BIMy platform to the user. Within this UI, azure-based applications are connected to the relevant units and users can access the selected application after logging in with Keycloak and use the applications described in the following sections.

When a user clicks on the login button, he/she is redirected to the login page of Keycloak. A two-factor authentication mechanism is used. In a first step, user credentials should be submitted, and as second step, a valid OTP should be submitted. Keycloak is integrated with HSM, and HSM is used for OTP generation. At the end of this operation, an access token is obtained. To consume the BIMy API endpoints, this access token should be given in the header of the http requests. If the BIMy API verifies the validity of this access token and the user has been given the necessary roles to access this endpoint, it returns the BIMy data or performs some operation on BIMy data.

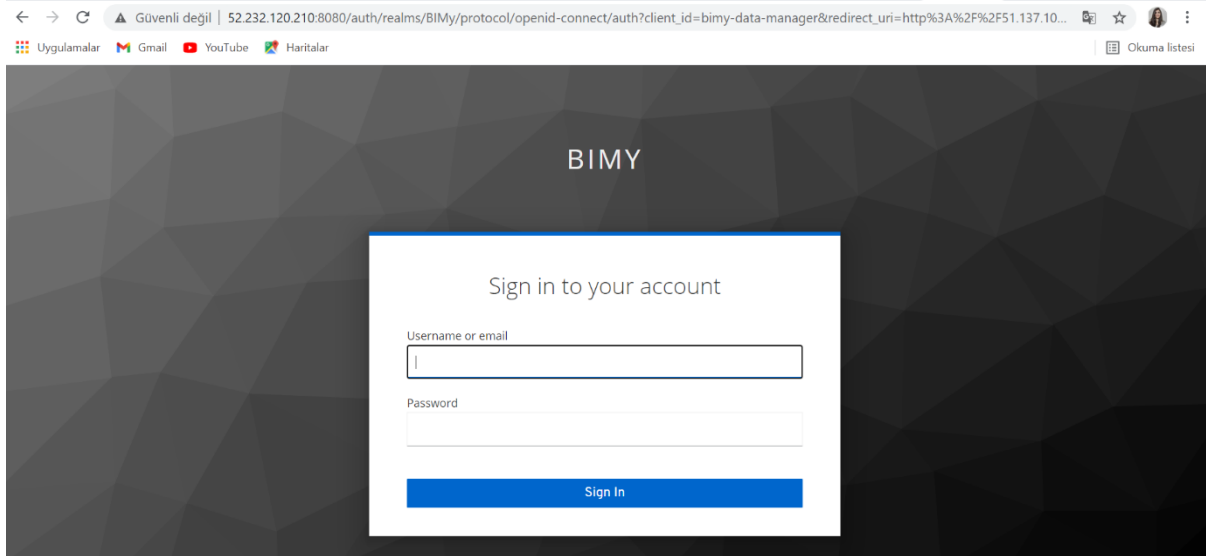


Figure 4: Platform Login Screen

After successfully logging in, the user is directed to the application page, where he needs to enter his OTP code.

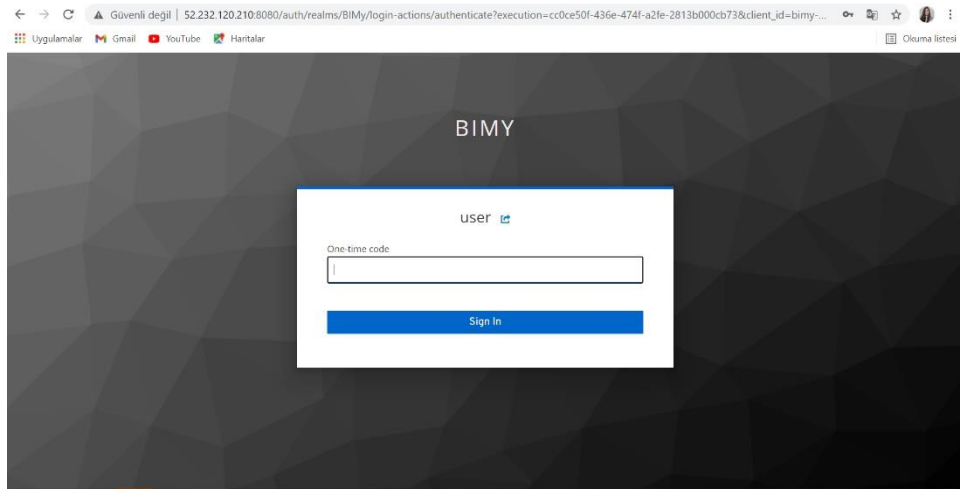


Figure 5: OTP Page

Here, all applications developed on the platform are made available to users on the frontend, as can be seen in the figure.

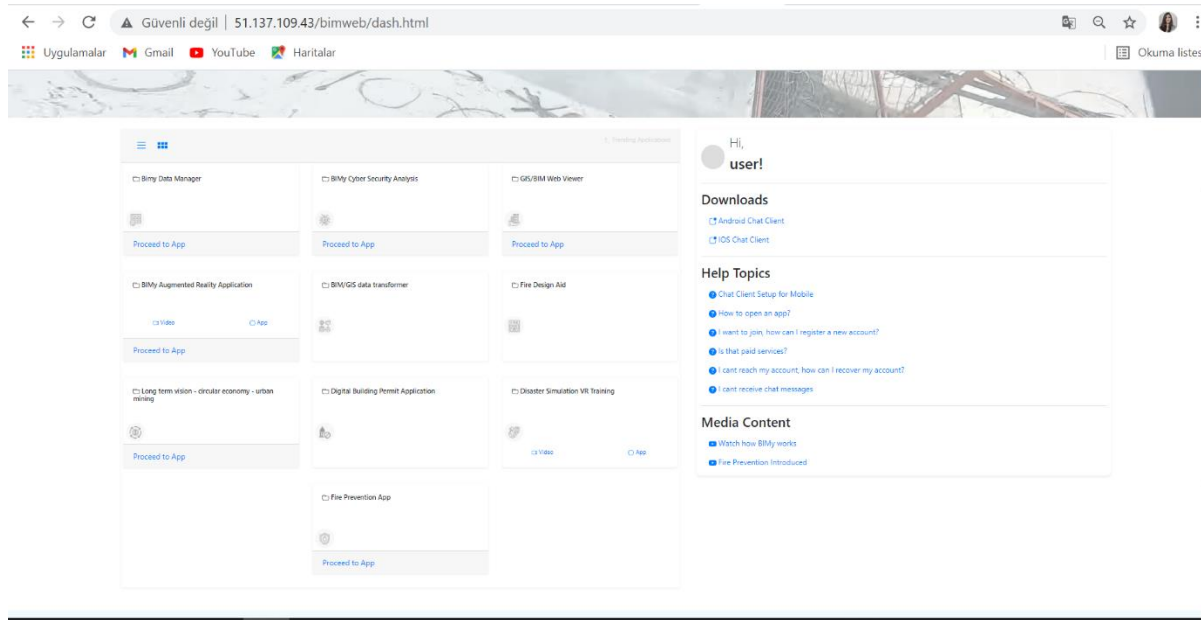


Figure 6: Applications on UI

At the same time, a chatbox has been developed that allows authenticated users to communicate and share information. This chat module provides instant messaging support between users. Transmission and read receipts of messages can be received bidirectionally.

Infrastructure Information:

XMPP (Extensible Messaging and Presence Protocol, Turkish ambiguity: Extensible Messaging and Presence Protocol), formerly Jabber [1], is an open XML protocol that allows two ends of the Internet to transfer any structural information between each other mutually and almost simultaneously.

Features:

While XMPP can be used for much more due to the infrastructure it offers, the first application area that has emerged is as an instant communication network that can be realized with services such as ICQ, AIM, Yahoo. It offers however many advantages than the messaging services currently used. XMPP protocols are open and free (public), and easily understandable; Multiple applications are available for clients, servers, components, and code libraries.

It is standard: IETF (internet engineering task force; Turkish word for word translation: General network engineering task force) core XML has formalized continuous broadcast protocols as the accepted messaging technology. It is used in XMPP and is developed by the XMPP Standards Foundation (formerly Jabber Software Foundation) in accordance with the standardization process of the IETF.

It has proven itself: Jabber technology was first developed in 1998 and is now almost completely stable. Currently, XMPP technologies are being developed by hundreds of developers, tens of thousands of servers are running on the Internet, and millions of people use it for messaging.

Decentralized: The architecture of the XMPP network is very similar to the email architecture, anyone can run an XMPP server.

Safe: Any XMPP server can be completely isolated from the XMPP network, for example it can only be used within a local network, and offers strong and robust security with the SASL (authentication security layer) and TLS (forwarding layer security protocol) included in the core XMPP specifications.

It is extensible: Anyone can build their own functions on the core protocol using XML.

Flexible: XMPP applications can be used to manage networks, share files and content, monitor the status of remote systems, play games, beyond messaging, thanks to the infrastructure provided by XMPP.

It is cosmopolitan: Many companies and open source projects use the XMPP protocol, developing real-time applications and services.

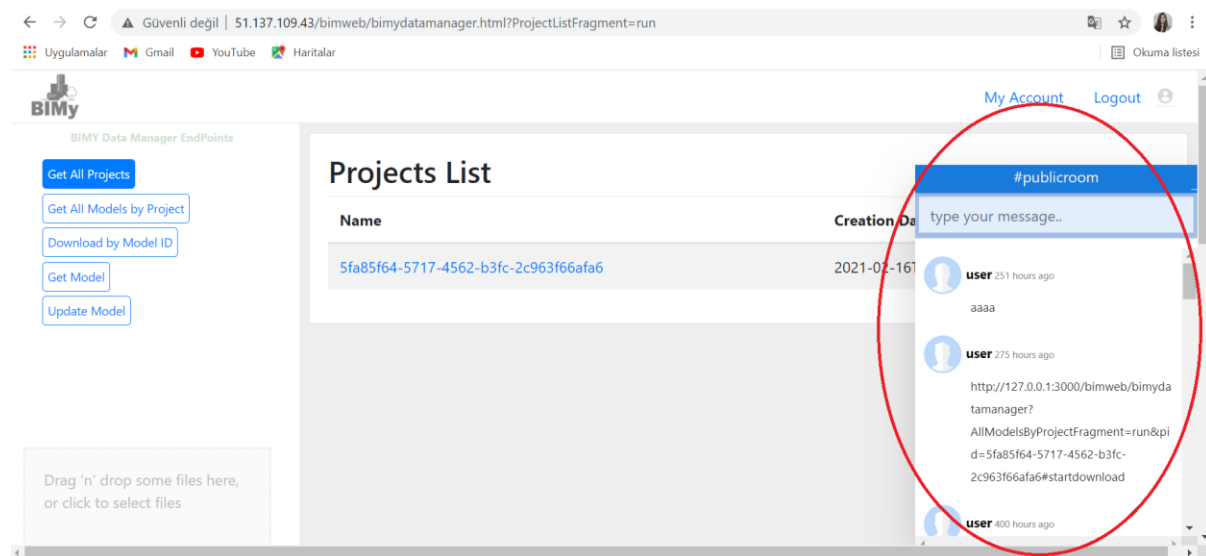


Figure 7: Chatbox for users to communicate via platform

3 BIMserver

The Building Information Model server (short: BIMserver) enables you to store and manage the information of a construction (or other building related) project. Data is stored in the open standard IFC. The BIMserver is not a fileserver, but it uses a model-driven architecture approach. This means that IFC data is stored in an underlying database. The main advantage of this approach is the ability to query, merge and filter the BIM-model and generate IFC files on the fly.

Thanks to its multi-user support, multiple people can work on their own part of the model, while the complete model is updated on the fly. Other users can get notifications when the model (or a part of it) is updated.

4 Fire Prevention App

When working on a project, architects need to keep in mind regulations and guidelines concerning fire prevention. To be able to communicate certain measures taken to comply with those regulations, certain information should be added to the model so it can be represented on floorplans, sections, 3D views, ... etc. This overview of required information is based on Belgian regulations and contains the information that's legally required to be provided. (e.g. when applying for a building permit.)

After the specifications and requirements for Fire Prevention have been established during the design phase, project stakeholder must ensure their compliance during the execution phase and further during the maintenance phase. To do so, inspectors carry out site inspections to verify information related to fire resistance of materials, evacuation routes, presence of equipment for fire-related emergencies, protection measures, etc. As such information should normally be included in BIM models, it would be ideal that inspectors have a reliable mechanism to access the relevant information supplied in the models. Further, to ensure that relevant project stakeholders are aware and can act about site inspection results, inspectors should have an effective mechanism to provide inspection data back into BIM models, as they serve as a common shared source of truth. However, reality nowadays is that inspectors normally do not have the means to access the multiple relevant data sets neither have a reliable method to provide site inspection feedback into BIM models.

In this regard, BIMy leverages its interoperability capabilities by connecting with an on-site inspections platform (LetsBuild) and empowers project stakeholders by creating an effective bridge that facilitates data exchange for the fire inspection process. The bridge makes BIM metadata available in the on-site mobile application and allows inspection results to be sent accessible through a BIM model. This information exchange is done based on the IDs of BIM objects, native IDs in the case of models in native formats and GUIDs in the case of IFC models. For this reason, it is crucial to consider that the changes in object IDs would have an impact in such workflow.

The relevant BIM models are sent to the LetsBuild platform using a native plugin for Revit models or by means of direct upload for IFC models. In this platform, the user has access to the following functionalities to facilitate the fire inspection workflow:

- Different BIM Models of a project
- BIM objects metadata
- Library of customisable site form templates according to fire inspection standards
- Sets of project participants that can carry out inspection tasks
- Customisable work breakdown structure
- Library of project documents like 2D plans from the BIM model, reports, specifications, etc.

The user can then leverage these data sources to create site inspections that include BIM objects information, standards specifications in the form of checklist, technical documentation, assignees and more.

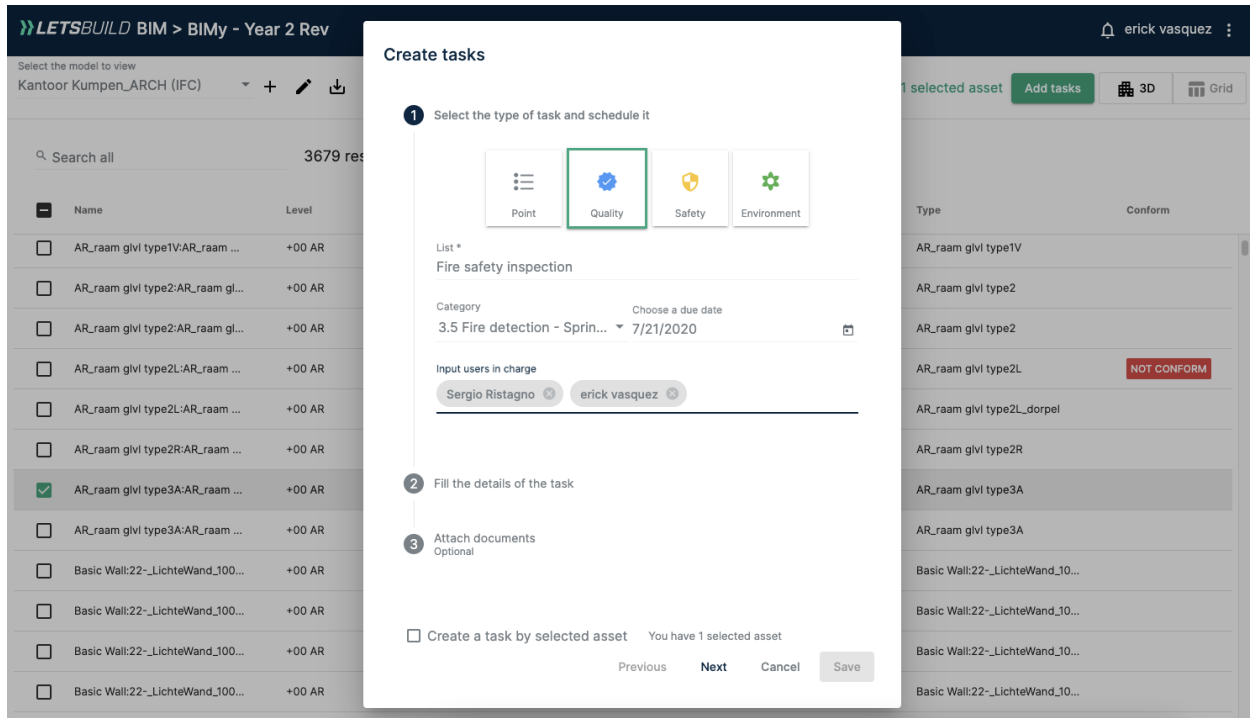


Figure 8: . Using LetsBuild BIM module to create inspection tasks linking BIM models, checklists, assignees and more

The inspection task is then transferred to the assigned stakeholder, who can use the mobile application to carry out the site inspection. The site inspector then has access to the relevant BIM metadata, the specified checklist and the technical documentation, significantly facilitating the inspection activity. In addition, such information is made available offline, in case the site inspector needs to work in a location where internet coverage is not guaranteed. For this task, the inspector uses the checklist to verify compliance with the specified standards, but also has the possibility to complement the activity with photos, comments or extra documents.

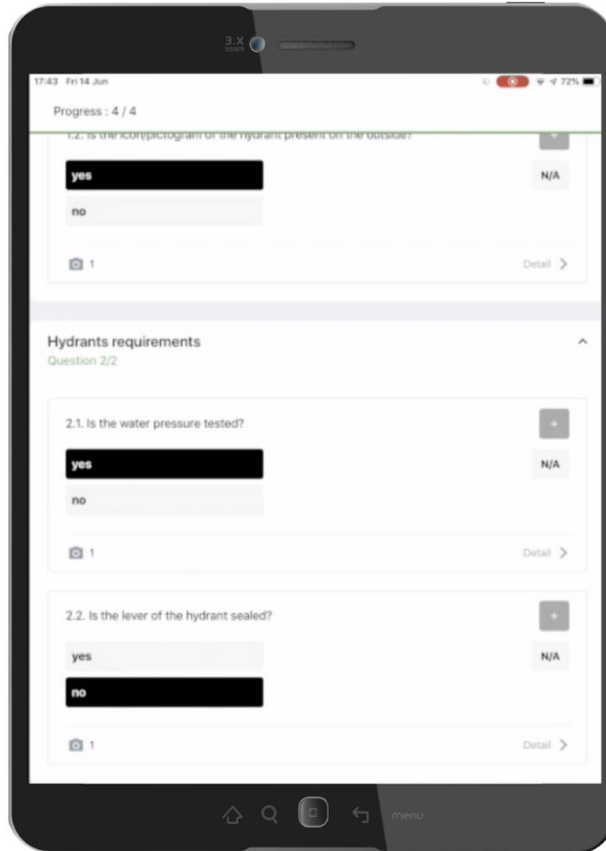




Figure 9: . Inspection task linked to BIM object carried out in LetsBuild mobile

After the inspection activity has been finalised, an inspection report is generated and the inspection details are prepared to be made accessible through the BIM model. In the case of Revit, the native plugin is used as the mechanism for information exchange, while in the case of IFC the data is directly added to the BIM objects and available in the IFC file when downloaded from the platform. To illustrate the output using an example, the Revit user sees a summary of the site inspection results by having a 'Conform' or 'Not conform' status on the object level, while also having a URL that includes all the details of the inspection if necessary (site photos, comments, relevant documents, non-conformities, etc)

Properties

 Single Window Standard

Windows (1) ▼  Edit Type

Rough Height	2700.0
Height	2700.0
Width	1500.0
Identity Data ⌵	
Image	
Comments	
Mark	95
Phasing ⌵	
Phase Created	Working Drawings
Phase Demolished	None
Other ⌵	
Head Height	2700.0
LetsBuild-Url	https://app.aproplan.com...
LetsBuild-Status	Not Conform

Figure 10: Site inspection details in Revit object

5 BIMy Data Model

The BIMy data model represents the minimal BIM and GIS objects that should be managed on the BIMy platform. This data model was developed as an Application Domain Extension (ADE) of CityGML 3.0 (Kolbe, 2020). An Application Domain Extension is a formal and systematic extension of the CityGML Conceptual Model (CM) for a specific application or domain. The ADE is expressed in the form of a UML conceptual model. The domain data is mapped to a set of additional classes, attributes, and relations. The BIMy Data Model is represented as a conceptual model, an XML Schema (GML Application Profile), and an RDF Schema.

Rationale

Via the BIMy API users can upload BIM models on the BIMy platform in one of the supported upload formats, such as IFC2x3, IFC4, or Revit. The BIMy platform stores the BIM model as a file, but additionally it will read a filtered set of relevant objects from the BIM model, convert these objects into a simplified, yet fully georeferenced, representation - this is the BIMy data model - and store this data in a datastore on the BIMy platform. The data store will then allow users to search, visualise, annotate, raise issues on, etc. these elements. Because the platform knows for each element in which BIM model(s) it occurs, the user can also download the corresponding BIM model (either as it was uploaded or converted into another format) and use a dedicated BIM tool for further analysis.

Conceptual model BIMy Data Model

The core of the BIMy Data Model supports the following key requirements:

BIMModel: A BIMModel is defined as a snapshot of a BIM model represented in a native format (e.g. Revit, IFC2x3, etc.). The BIMy data model makes it possible to find out in which BIMmodel a VersionedObject occurs. The BIMy Data Model therefore does not need to have the same level of detail (level of geometry, level of information need) as the linked BIMModel.

Identifier: The data type Identifier consists of a namespace (a URI), a localId, and an optional versionId. The concatenation of the {namespace}{localId} attributes makes up a (preferably HTTP) URI. This enables a "Linked Construction Data" approach where each Object has a Web identifier (HTTP URI). The BIMy ReST API will resolve this URI to a CityGML or JSON document describing this object. The combination of {namespace}{localId} (without versionId) is also expected to be a stable object identifier that can be used to refer to the object during its lifecycle.

Object-level versioning: In CityGML 3.0 every FeatureType is a subclass of the abstract class AbstractFeatureWithLifespan. The validFrom and validTo dates indicate the validity of the object in the real world. The creationDate and terminationDate are the dates the object was created.

Basic object properties: each object has very basic properties such as name, description, etc. that are inherited from GML.

Project, Building, BuildingStorey, Space: the model allows representing key BIM/CityGML entities. Note that Building has a building footprint property (multipolygon), which is a typical GIS concept.

Element: all key IFC building elements are modeled as Element on the BIMy platform. An Element may be classified by one or more Classifiers.

Support multiple levels of geometry: An element must be modeled as a (simplified) GML MultiSurface geometry. For the moment, only LoD 1 and LoD 2 are considered. Also, only simple surface geometries are in scope of the BIMy platform, as the main requirement is visualisation of these geometries.

ElementType: An element may also be linked to an ElementType. The IFC standards have a similar concept.

Support shared representations: Rather than instantiating each geometry, the BIMy data model can support common shape representations with the RepresentationMap concept, greatly reducing the amount of storage required to store frequently occurring geometries (e.g. a water hydrant shape). The IFC standard has a similar concept (IfcRepresentationMap).

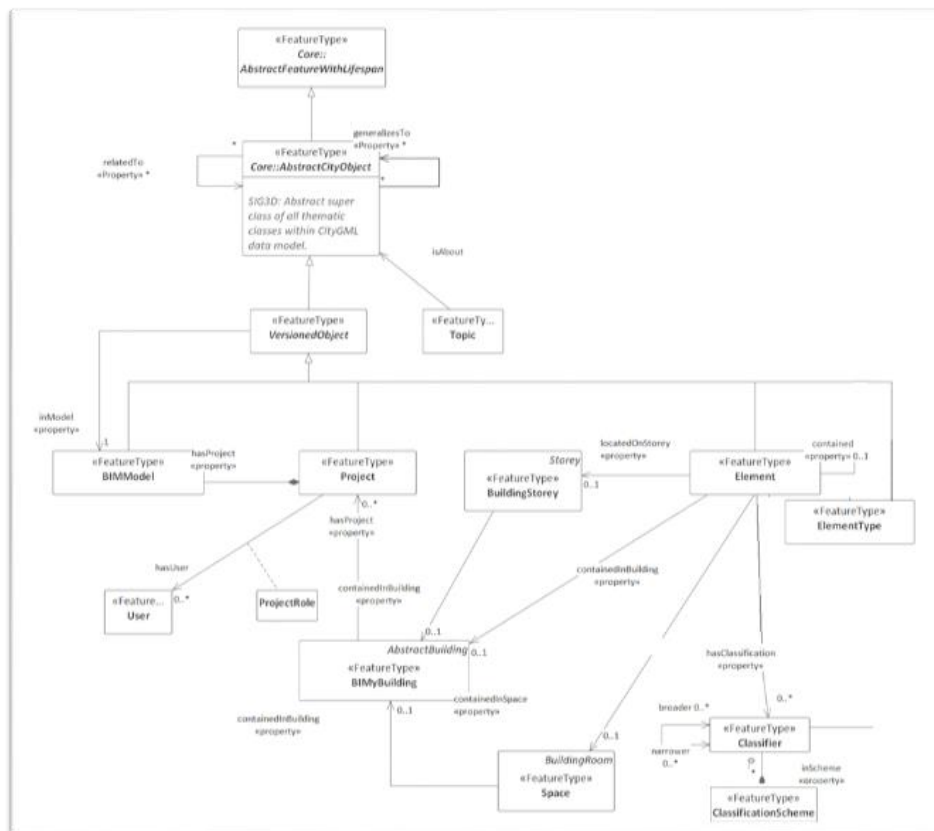


Figure 11: BIMy conceptual data model (without properties)

XML Schema encoding

From the above conceptual model developed in the Unified Modelling Language, an encoding in XML (bimy.xsd) has been derived in the form of a CityGML 3.0 application schema using a model-driven approach. To note is that the CityGML 3.0 XML encoding is -at the time of writing- only available as an informative resource that will be further standardised at the OGC and hence future changes may be required.

Linked Data Vocabulary

The bimy.ttl vocabulary is an encoding of the BIMy Conceptual Data model based on the Building Topology Ontology (BOT) and the Simple Knowledge Organization System (SKOS). [BOT](#) is a minimal ontology for describing the core topological concepts of a building. It defines the relationships between the sub-components of a building. It was prepared by the Linked Building Data Community Group of W3C and a draft for public comments is available.

6 BIMy REST API specs

The BIMy API consist of a custom, ReST API supplemented with standardized API services. The platform API specifications are defined using the OpenAPI 3.0.2 specification. The specification is maintained as a separate YAML document.

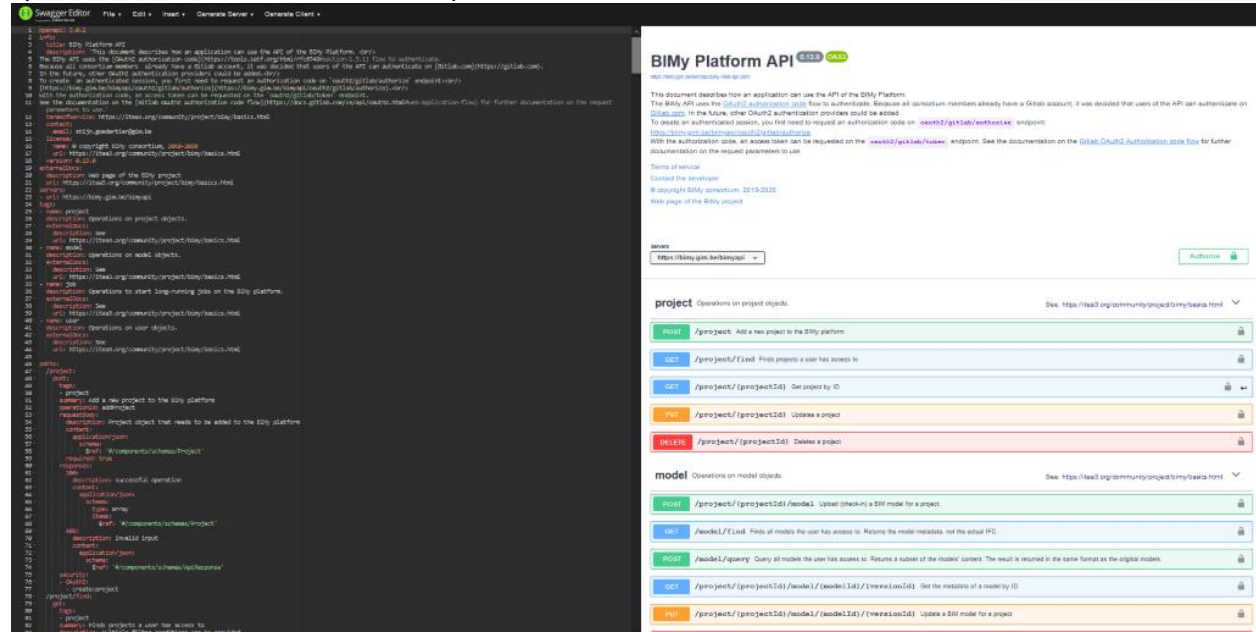


Figure 12: BIMy Platform API (openAPI specification)

7 BIMy Data Manager

The BIMy data manager provides a generic BIM model storage service which is used in the platform applications. It consists of a storage API and an object storage service which has been implemented in the MinIO client verticle and the BIMserver client verticle.

Our reference implementation uses the Eclipse [Vert.x](#) toolkit for building *reactive* applications on the JVM. Vert.x is resource-efficient, that is to say, it can handle more request with less resource compared to traditional frameworks based on blocking I/O. It is also flexible and easily scalable. On the other hand, it makes asynchronous programming easier, without sacrificing correctness and performance. A high performance asynchronous programming is important in BIMy platform since it is needed to handle long-running tasks such as the upload of a BIMy model which can be a large file.

The application contains 3 so-called *verticles* (code that can be deployed on Vertx):

- The *HTTP verticle* listens to incoming HTTP requests. Those requests will be validated against the OpenAPI specification and a mapping will be done between the request and a Java method. This method will then propagate the request on the event bus, waiting for a reply.
- The *BIMserver client verticle* listens to the event bus, waiting for incoming request from the HTTP verticle. On event, it will fetch the requested data on the BIMy API and return them via the event bus.
- The *MinIO client verticle* listens to the event bus, waiting for incoming request from the HTTP verticle. On event, it will fetch the requested data on the BIMy API and return them via the event bus.

MinIO has been chosen as object storage technology as it is a highly performant object storage server with an API that is fully compatible with the Amazon S3 cloud storage service.

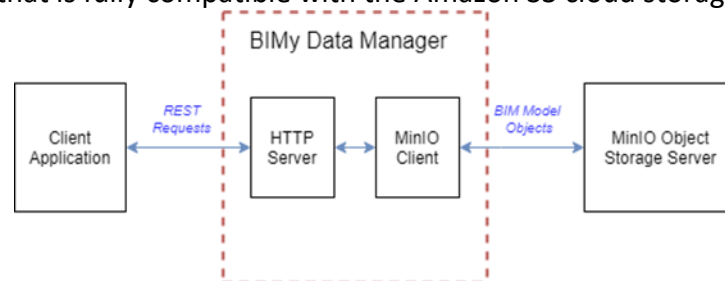


Figure 13: BIMy Data Manager Architecture

Also, many other operations are possible such as retrieving, updating, and deleting a model. MinIO Java SDK^[1] is utilized for this purpose.

The model service is implemented with 4 basic operations:

- `uploadModel`: Upload (check-in) a BIM model for a project.
- `getModelById`: Get the metadata of a model by ID.
- `updateModel`: Update a BIM model for a project.
- `deleteModel`: Delete model by ID.

Besides, metadata as sidecar files are stored under related projects as well as the model file itself.

A simple versioning is achieved by including versions in the MinIO storage path structure. Files with different formats can be uploaded into the base folder. Figure 10—5 shows the MinIO server UI after uploading a BIM model. The model is accompanied by a JSON sidecar file with metadata relating to the BIM model object.

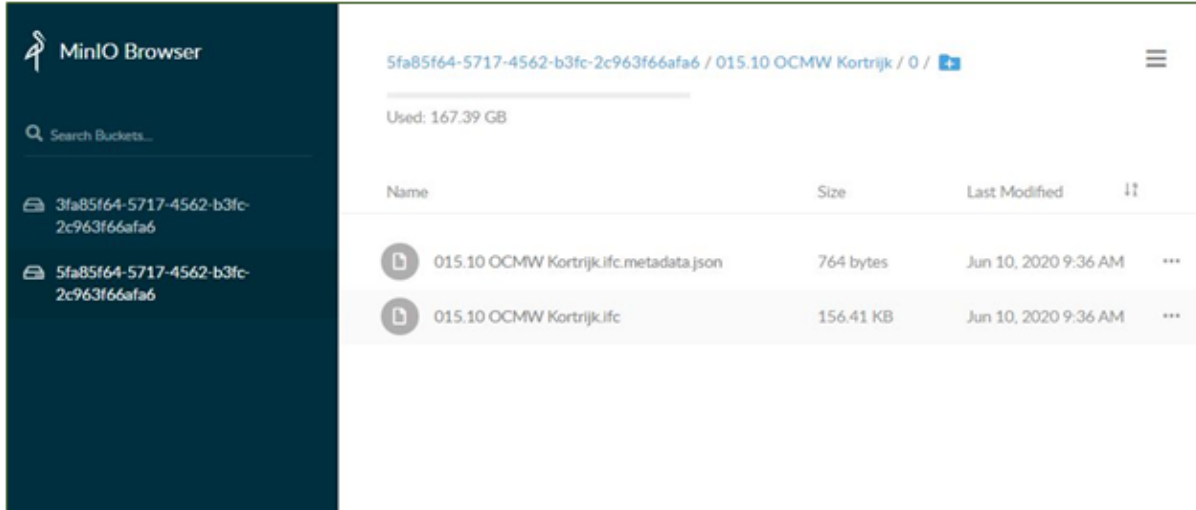
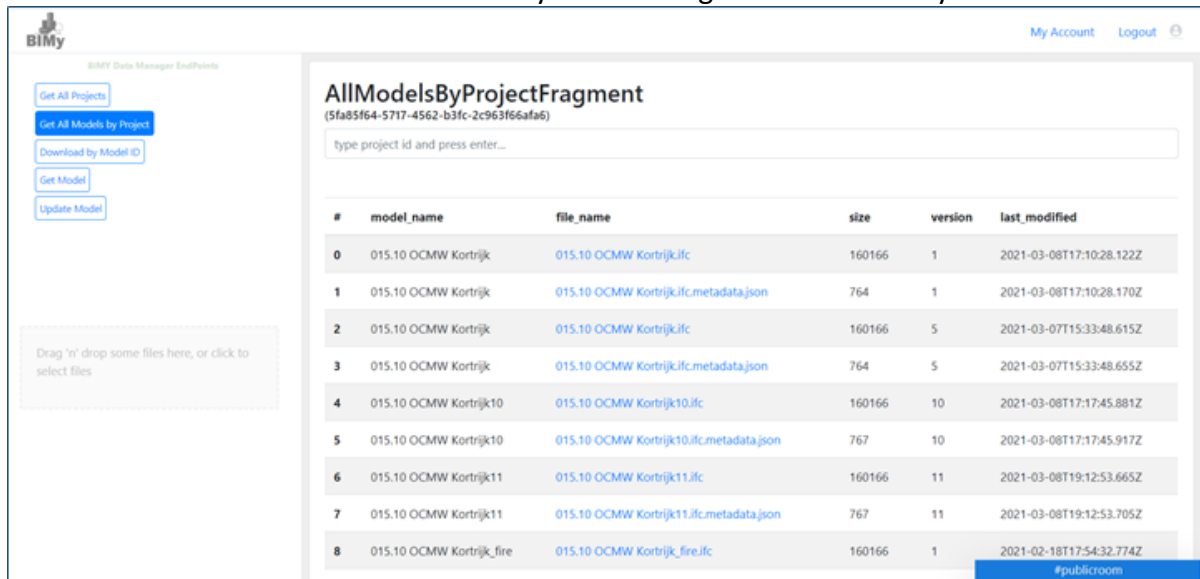


Figure 14: MinIO Browser

On the platform, BIMy Data Manager has been integrated Azure and this Cloud platform has been visualized with a WebUI. Here is the BIMy Data Manager section of BIMy UI:



Role Name	Composite	Description	Actions	
adminProject	True	Allows every operation on BIMy data.	Edit	Delete
createProject	False	Allows to upload models to BIMy platform.	Edit	Delete
readProject	False	Allows to view projects and models and download models on BIMy platform.	Edit	Delete
writeProject	False	Allows update and delete of models on BIMy platform.	Edit	Delete

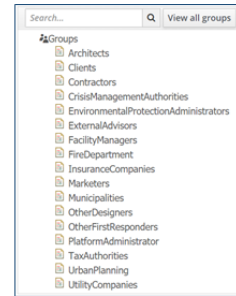


Figure 15: : BIMy Data Manager on WEB

^[1] <https://docs.min.io/docs/java-client-api-reference.html>

8 BIMy Data Transformer

This building block provides functionality to transform and subset BIM models to GIS formats and GIS formats to IFC. The supported format conversions are presented in Table 3—1.

Table 1. Supported format conversions

TO ↗ FROM	IFC	Floorplan (SVG)	Revit	Cesium 3DTiles	CityGML ADE	BOT	GeoTiff	GML	LiDAR	Shape	Film Box	Unreal Datasmith	OBJ/Material (AR)
IFC		✓		✓	✓	✓					✓	✓	✓
Floorplan (SVG)													
Revit	✓			✓							✓	✓	✓
Cesium 3DTiles													
CityGML ADE	✓												
BOT													
GeoTiff	✓		✓										
GML (IMKL)	✓		✓	✓	✓								
LiDAR (LAS)	✓		✓	✓	✓		✓				✓	✓	✓
Shape	✓		✓	✓	✓						✓	✓	✓
DEM	✓			✓							✓	✓	✓

As example, the transformation from BIM IFC to the BIMy data model City GML ADE encompasses among others the following transformations:

Reading IFC objects: all IFC objects are read and their scale converted.

Transformation of geometries: resolving the solid geometries (CSG or BRep) and convert geometries into MultiSurface representations.

Georeference geometries: convert the Cartesian coordinates of the IFC file into geospatial coordinates according to a reference geometry. CityGML uses a 3D geodetic coordinate reference system. Because IFC files may use local coordinate systems, this may require the following adjustments:

- **Scale:** adjust the unit of measurement, e.g. convert from millimetre into metre.
- **Rotate:** it may be needed to rotate the objects. This is particularly needed if the y-axis is not directed at the true north.
- **Offset:** It may be needed to offset the (x, y) coordinates in the IFC file according to the required translation defined in the IFC

file's `IfcGeometricRepresentationContext WorldCoordinateSystem` attribute. It may also be needed to adjust elevation, for instance, if a different vertical reference system is used (e.g. Belgian TAW versus Dutch NAP).

- **Reproject:** after applying the above-mentioned scaling, rotation, and offset, the coordinates of the IFC file may be in a geodetic coordinate reference system.

Readjust object hierarchies: in IFC, many objects have an `IfcBuildingStorey` as direct parent. Other objects have a more complex hierarchy.

Construct aggregates: In IFC, some instances are composed of other instances. For example, curtainwalls contain plates and members; stairs contain railings, stair flights and slabs; and ramps contain slabs and ramps. These parts must be added to their parent feature when converting to CityGML.

Rename object properties: for instance, the-unique id of each object can be used as `gml:id` attribute in CityGML.

Change instance classification: convert the IFC classification of instances into a corresponding CityGML feature type.

9 BIMy Integrated BIM/GIS Web Viewer

The **BIMy Integrated BIM/GIS Viewer** has been developed as a Progressive Web Application (PWA), a type of application software delivered through the web, built using common web technologies including HTML, CSS and JavaScript. It is intended to work on any platform that uses a standards-compliant browser, including both desktop and mobile devices. PWA features narrow the gap between user experience in web-based and native applications. The Viewer is developed using typescript and incorporates a modular design, both to minimize the dependency on particular Javascript frameworks. It allows for the combined visualisation of BIM models and the contextual GIS data consisting of Digital Terrain models, cadastral parcels, vegetation, underground infrastructure and the surrounding buildings in LoD2. It also interfaces with the BCF APIs for the creation and follow up of BCF topics. It is based on a modular design that allows to select individual modules in function of the functionalities that are required for a specific instance. It is integrated with Keycloak that provides the authentication and authorisation functionality for the BIMy system. It relies on the BIMy data transformation component to convert all 3D BIM/GIS information into 3D Tiles.

The following functionality is implemented:

- 2D and 3D Navigation with seamless switch between the 2d and 3D views
- Layer manager
- BIM object tree
- Identification of objects
- 2D and 3D measurement functionality
- Attribute table that is kept in sync with the map view
- Editing functionality for GIS objects
- BCF client whereby an authenticated user can create and manage BCF topics hereby interfacing with the BCF manager.



Figure 16: BIMy Web Viewer snapshot



Figure 17: Snapshot from GIS/BIM Web Viewer

10 BIMy BCF Manager

Repository: <https://gitlab.com/bimy/bimy-bcf-manager> (private, not publicly accessible)

This project holds an implementation of the [BCF API](#) defined by BuildingSMART.

The API has been developed in .NET core (.NET5), the data are saved in a MongoDB system and files that can be attached to a BCF topic are saved to a Minio Server.

The BCF API supports the version 2.1 of the BCF standard.

10.1 Getting started

10.1.1 Running the application

A MongoDB server to store data and a minio server storing document should be up and running. To set up a test environment we use docker with docker-compose where those two services are defined.

Set up this infrastructure,

```
docker-compose up -d
```

Compilation of the application can be done with the following dotnet commands:

```
dotnet restore "Bimy.Bcf.Manager.sln"  
dotnet publish "Bimy.Bcf.Manager.sln" -p:PublishDir=publish  
dotnet Bimy.Bcf.Manager.Api/publish/Bimy.Bcf.Manager.Api.dll
```

As the application is an API, the specification of that API is done by using OpenAPI 3 specification and it is auto-generated by the code using "[Swashbuckle.AspNetCore 5.6.3](#)" dotnet library. Locally, the application is reachable to the following address through your browser <http://localhost:5000/swagger>.

ApiVersion		▼
GET	/versions	
Auth		▼
POST	/auth	
Comment		▼
GET	/{apiVersion}/projects/{projectId}/topics/{topicId}/comments	🔒
POST	/{apiVersion}/projects/{projectId}/topics/{topicId}/comments	🔒
GET	/{apiVersion}/projects/{projectId}/topics/{topicId}/comments/{commentId}	🔒
PUT	/{apiVersion}/projects/{projectId}/topics/{topicId}/comments/{commentId}	🔒
DELETE	/{apiVersion}/projects/{projectId}/topics/{topicId}/comments/{commentId}	🔒
Document		▼
GET	/{apiVersion}/projects/{projectId}/documents	🔒
POST	/{apiVersion}/projects/{projectId}/documents	🔒
GET	/{apiVersion}/projects/{projectId}/documents/{documentId}	🔒
Project		▼
GET	/{apiVersion}/projects	🔒
POST	/{apiVersion}/projects	🔒
GET	/{apiVersion}/projects/{id}	🔒
PUT	/{apiVersion}/projects/{projectId}	🔒
GET	/{apiVersion}/projects/{projectId}/extensions	🔒
GET	/{apiVersion}/projects/export	🔒

10.2 Security: OAuth2 authentication

The BCF API uses the [OAuth2 authorization](#) code flow to authenticate. Because all consortium members already have an account on Gitlab, it was decided that users of the API can authenticate on [Gitlab.com](#).

The authentication was implemented using the [NGitLab](#) to facilitate the implementation as we should have an authentication service.

In the future, other OAuth2 authentication providers could be added.

10.3 Functionalities

In the context of BIMy, it was not necessary to include all the BCF features specified by BuildingSMART for the BCF API (<https://github.com/buildingSMART/BCF-API>). Therefore, it was decided to only implement the services that would add value to BIMy-oriented use cases. Here is a list of services and their description that are included in the BCF API:

10.3.1 Version's service

End point `/versions` returning the list of BCF versions is available through the api. One value of the list should be used as path parameter of all other endpoints.

The currently supported version is: v2.1. This is the only endpoint can be called without being authenticated.

10.3.2 Project service

A BCF project is a container where the topics are classified and organised. Project members can be set up to manage accessibility duties. It is also possible to configure projects to define values that can be used in topics and comments as topic type, priorities, status and more.

This service contains several endpoints to manage a BCF project: create, update, configure or get details.

The projects service endpoints start by: `{bcfVersion}/projects`

10.3.3 Topic service

The topic is the main entity in the BCF schema, it represents an issue or observation to raise over a BIM object; topics are organised by project. Through this service, it is possible to retrieve the list of topics in a project to which a user has access to. This service also allows to create, update or delete a topic.

The topic service endpoints start by:

`{bcfVersion}/projects/{projectId}/topics`

10.3.4 File service

The file service is used to enrich a topic by specifying the ifc file on which the topic is making references.

The file service endpoints start by:

`{bcfVersion}/projects/{projectId}/topics/{topicId}/files`

10.3.5 Comment service

Comment service is used to enrich a topic permitting the different members of the project to communicate over a topic by creating comments.

The comments service endpoints start by:

```
{bcfVersion}/projects/{projectId}/topics/{topicId}/comments
```

10.3.6 Viewpoint service

The viewpoint service is used to enrich a topic by adding view context to the topic:

- How the topic should be seen when opened over the BIM model (where the camera should be positioned in the 3D space)
- A snapshot of the view when the topics has been raised
- The model objects of the model to highlight, hide/show, transparency, etc.

Viewpoints are immutable, meaning that they should never change.

The viewpoint service endpoints start by:

```
{bcfVersion}/projects/{projectId}/topics/{topicId}/viewpoints
```

10.3.7 Document reference service

The document reference is used to enrich a topic by adding a reference to a document. The topic is either referencing a document stored in the BCF service itself using its ID or instead to a document located in another repository by using its URL. The service targets to reference any documents that provide meaningful information about the topic. Documents referenced in the BCF service itself are managed by project in the document service.

The document reference service endpoints start by:

```
{bcfVersion}/projects/{projectId}/topics/{topicId}/document_references
```

10.3.8 Document service

The document service is used to store document necessary to enrich the context of topics. They are stored inside a project and can be referenced by a topic through the document reference service.

The document service endpoints start by:

```
{bcfVersion}/projects/{projectId}/documents
```

10.3.9 BCF Export

A BCF Export endpoint has been created in order to allow exporting topics to a BCF file for use in other tools that support BCF (BCF Managers like BIM Collab or Solibri). This endpoint is not defined in the standard specification of BuildingSMART.

Our search for existing software components yielded an XBIM library (developed in .NET). Unfortunately, that library hasn't been updated since 2017 and we quickly discovered issues with the product. In addition, some key features weren't developed or implemented in XBIM either.

We decided to create a fork from the XBIM library to fix major issues and to implement the missing functionality we needed, aiming to provide a functional BCF API for the BIMy platform.

The export is done by project, so the end point is `{bcfVersion}/projects/export` and the body of the call is a json object containing the following properties:

- **ProjectId** (required): The projectId you want to export
- **TopicIds** (optional): An array containing the topic Id that should be exported. If no defined, all topics will be exported.