BIMy Project:
D3.2 Development: overview of current developments

Document metadata

Date	2019-06-17
Status	Draft
Version	0.04
Authors	Osman Kumas - NETAS Stijn Goedertier – GIM Sergio Ristagno – Aproplan İbrahim Salgın - NETAŞ
Coordinator	Osman Kumas - NETAS
Reviewed by	

Version history

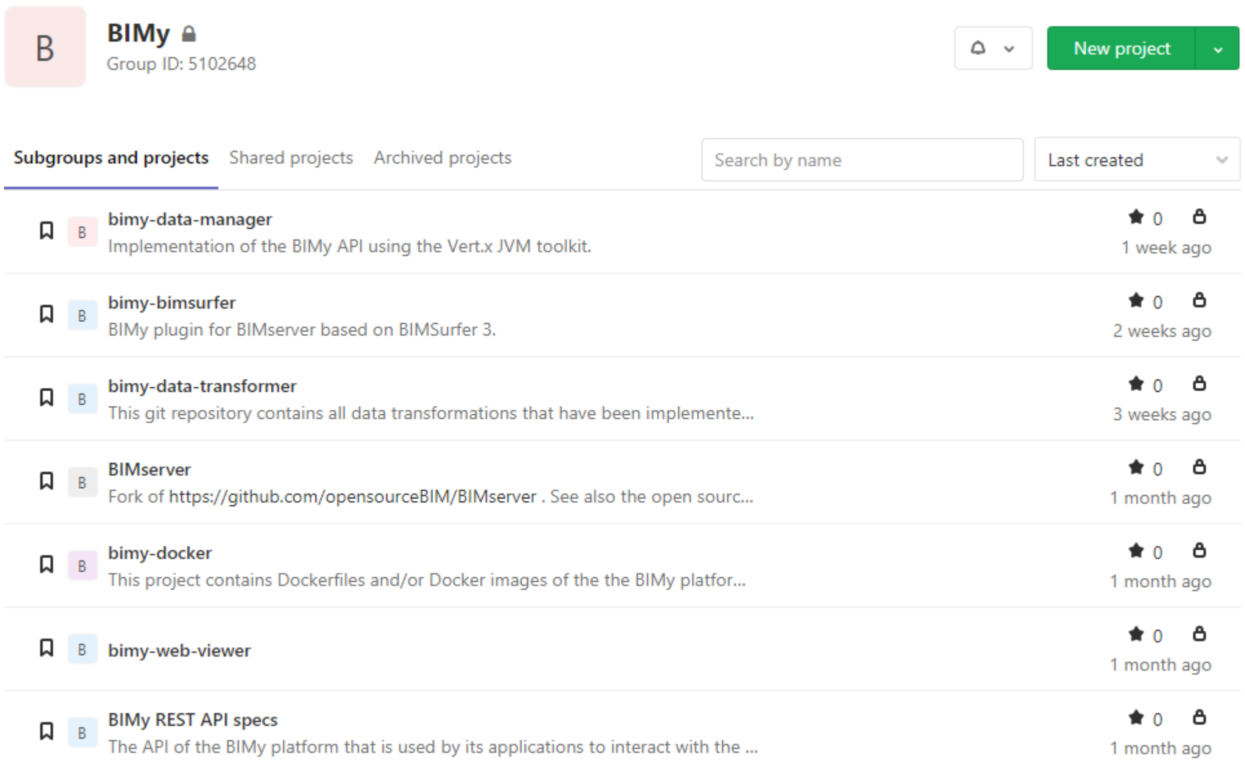
Version	Date	Author	Change
0.01	2019-04-09	OKU	Introduction and scope
0.02	2019-05-16	SGO	Document bimy-docker
0.03	2019-06-17	SGO	Document bimy-data-manager, bimy-bimsurfer, bimy-web-viewer

1 Table of Contents

1 Introduction

This document gives an overview of the BIMy development projects.

The main strategy is to implement the solution building blocks those are described in ‘D1.3 Platform architecture and API specs’ (BIMy consortium, 2019a, p. 3), therefore a multi-tiered software architecture of microservices is used. Development environment installed on BIMy cloud and it is in progress as mentioned on the project plan. Following sub projects resides on development environment.



bimy-data-manager
Implementation of the BIMy API using the Vert.x JVM toolkit.
0 — 1 week ago

bimy-bimsurfer
BIMy plugin for BIMserver based on BIMSurfer 3.
0 — 2 weeks ago

bimy-data-transformer
This git repository contains all data transformations that have been implemente...
0 — 3 weeks ago

BIMserver
Fork of https://github.com/opensourceBIM/BIMserver . See also the open sourc...
0 — 1 month ago

bimy-docker
This project contains Dockerfiles and/or Docker images of the the BIMy platfor...
0 — 1 month ago

bimy-web-viewer
0 — 1 month ago

BIMy REST API specs
The API of the BIMy platform that is used by its applications to interact with the ...
0 — 1 month ago

Figure 1Screenshot of the gitlab development environment

2 BIMserver

Repository: https://gitlab.com/bimy/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 it's 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. Furthermore the BIMserver will warn you when other users have changed something in the model while you were editing.

3 BIMy Aproplan Fire Prevention App

Aproplan will provide a platform

- to visualise parts of the BIM model during inspection (views (starting in 2D at first) and meta-data of BIM objects),
- to perform the inspection in a digital format
- to capture and track issues identified during the inspection
- to update the BIM model with proof of the audit done and the state of the building (or part of) audited

Pre-requisites:

- The BIM model is available in the BIMy platform
- The BIM model includes views with the relevant information for the field, which can be shared in a PDF format at first
- The BIM model has a structure of asset/BIM Object leveraging an agreed classification
-

The flow can be as depicted in Figure 2:

1. The BIM model is sent to /extracted by APROPLAN using a format such as Revit (.rvt) or IFC
2. The Asset/Bim object tree is extracted
3. For scheduled inspections
    a. Via a simple interface the inspection tasks can be created leveraging the asset tree
        i. Leveraging the list of assets inspection tasks can be created and assigned to a specific user and linked to the relevant drawing
        ii. Leveraging the classification of the asset the right team can be assigned to the issue identified
        iii. Leveraging the classification of the asset the right checklist can be assigned to the tasks to ensure inspection follows the required regulation
    b. Based on the above an auditor (member of the fire brigade prevention team) would go on site and open his tablet on the particular floor plan provided and can go through all the checks to be done on that floor.
        i. If an issue arise (e.g. an answer to a form is not as expected. E.g. fire extinguisher B1234 has its expiration date overdue) he can create an issue that can be linked to that particular extinguisher in the BIM model.
            1. In doing so he can take a picture, assign the right company to solve this based on the category/classification of the object having the issue
            2. Then the issue follows its regular workflow: TODO by maintenance company, done and to be checked by property owner, to be verified by Fire brigade inspection.
            3. Then at the next inspection by the fire brigade this checklist can be re-open and updated to remove the issue and confirm all is in order
    c. Following the audit on site a report can be automatically created with

        i. Status of the audits done
        ii. Issues raised and to be fixed before the next inspection in order to confirm the building is conformed to the Fire prevention rules
    d. Upon request this information can be pushed back to the model for other systems relying on the system to be aware of the status of the bim objects and have proof of the audits done.
4. For unscheduled inspection
    a. When on site a fire prevention officer can use his tablet and do ad-hoc check.
    b. Using a drawing he can add a check to a specific object desired to audit following a checklist in the tool.
    c. If issues arise the same flow as above can be followed
    d. Once done his audit can also be added to the BIM model
5. Tracking of audits
    a. Once the audit information is uploaded to the BIM model it can be used during the operation of the building
        i. Objects with issues can be easily extracted from the BIM model for external audits, summary briefing of a fire brigade before going on site, etc.

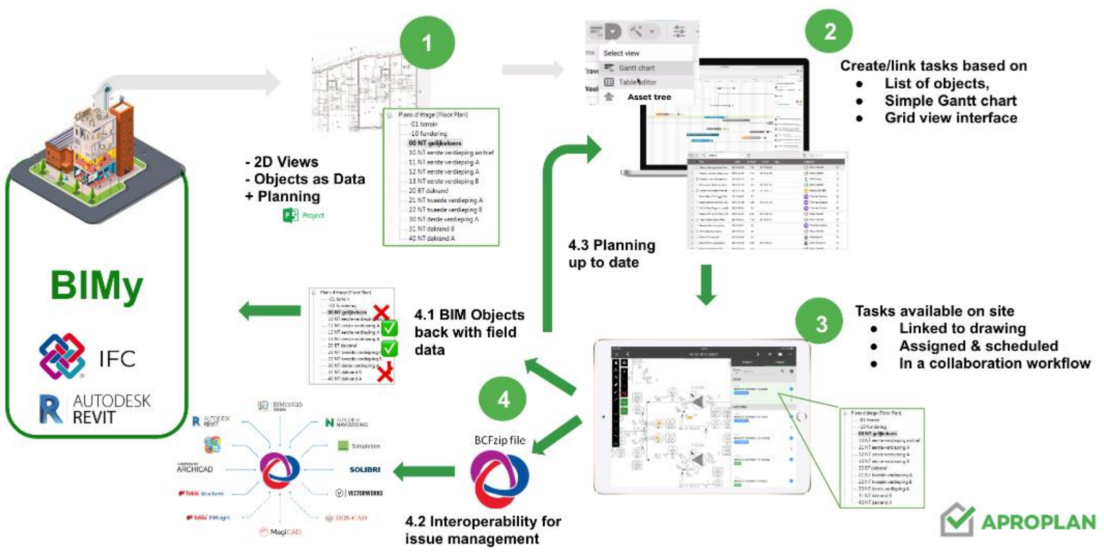


Figure 2 Flow of the Fire Prevention App

4 BIMy REST API specs

Repository: https://gitlab.com/bimy/bimy-rest-api-specs

The BIMy REST API is used by applications to interact with the BIM platform. The API reuses as where possible existing standards, but also defines specific JSON schemas and operations. The platform API specifications is defined using the OpenAPI 3.0.2 specification.

Swagger editor

To view or edit the API specification, paste the file 'bimy-rest-api.yaml' into the online editor at https://bimy.gim.be/swaggereditor/?url=https://bimy.gim.be/bimyapi/bimy-rest-api.yaml.

5 BIMy Data Manager

Repository: https://gitlab.com/bimy/bimy-data-manager

This project holds an implementation of the BIMy openAPI spec (simplified version) using the Eclipse Vert.x toolkit for building reactive applications on the Java Virtual Machine.

5.1 Getting started

5.1.1 Running the application

First, we need to compile and let vertx generate the necessary code for our application. Running the Maven goal "install" in this directory will do the trick:

```
mvn clean install
```

This command will generate a `target` folder containing all our generated and compiled code. The generated code can be found in `target/generated-sources` folder

The `target` folder will also contain `data-manager-0.1.0-SNAPSHOT-fat.jar` file. This jar file groups the application and all its dependencies which mean it can be executed as standalone. To run it, use the following command:

```
java -jar target/data-manager-0.1.0-SNAPSHOT-fat.jar
```

To test the application, you can try the following endpoint that take an UUID in path parameter and return a static JSON data structure containing the UUID sent: http://localhost:8081/model/6bfb284a-7624-11e9-95e1-2a86e4085a59

5.1.2 How it works

This implementation uses the Vertx toolkit for building *reactive* applications on the JVM. The application contains 2 verticles:

- HTTP verticle
- BIMserver client verticle

5.1.2.1 HTTP verticle

This first verticle has for role to listen incoming HTTP requests. Those requests will be validated against the OpenAPI spec 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.

5.1.2.2 BIMserver client verticle

The second verticle will listen to the event bus waiting for incoming request from the HTTP verticle. On event, it will fetch the requested data on the BIMy's API and return them via the event bus.

5.1.3 Creating a new service

For clean code reason, a service should be created for each "super path" defined in the specs:

- Project
- Model
- Building
- User

Each services will implement their respective endpoints defined in the OpenAPI specs.
To create a new service, follow the steps with the Model service as example:

5.1.3.1 Step 1

First, we will create a data access object as a Proxy service that will be used to query BIM rest API (or use BimServerClient java lib).
For that, we need to create an interface in the package `org.bimy.gim.bimserverclient.services`. This interface must annotated by `@ProxyGen` and at least implement 2 methods:

- *create()* : This method will return an instance of the implementation class of this interface
- *createProxy(Vertx vertx, String address)* : This method will return an instance of the service proxy that will automatically be generated on the first compilation (Don't pay attention to the errors now). The address will be the identifier of this service.

This gives us this interface:

```
package org.bimy.gim.bimserverclient.services;
import /*...*/;

@ProxyGen
public interface ModelDaoService {

  static ModelDaoService create() {
    return new ModelDaoServiceImpl();
  }

  static ModelDaoService createProxy(Vertx vertx, String address) {
    return new ModelDaoServiceVertxEBProxy(vertx, address);
  }

  //Define methods to access data here
}
```

5.1.3.2 Step 2

Create the class that will implement this interface under the package `org.bimy.gim.bimserverclient.services.impl`.

```
package org.bimy.gim.bimserverclient.services.impl;
import /*...*/;


public class ModelDaoServiceImpl implements ModelDaoService {
  //Implement methods to access data here
}
```

5.1.3.3 Step 3

Once the service is created, it need to be linked to a verticle. For DAO's object, it must be linked to the BimServerClient verticle. This is done by using a service binder.
In `org.bimy.gim.bimserverclient.BimServerClientVerticle` class:

```
new ServiceBinder(this.vertx)
  .setAddress(AddressConstants.BIMY_SERVICE_MODEL)  /*  An  address  must  be
assigned to the service to be able to find it from anywhere in the application.
*/
  .register(ModelDaoService.class, new ModelDaoServiceImpl());
```

You can already compile the code to let Vert.x generate some code (and remove the errors). The generate code can be found in `target/generated-sources/annotations/org/bimy/gim`

5.1.3.4 Step 4

Now that we have our data access object, we can create a new service interface in the package `org.bimy.gim.http.services` called "ModelService". This interface will be used by Vert.x Web API Service extension to generate the code that will add all the routes to the router for us. This service must be annotated by `@WebApiServiceGen` and optionally extend the `Service` interface:

```
package org.bimy.gim.http.services;
import /*...*/;


@WebApiServiceGen
public interface ModelService extends Service {
  //Declare all the endpoints for Model.
}
```

5.1.3.5 Step 5

We must create a class that will contain the implementation of this interface. Create the class named `ModelServiceImpl` under the package `org.bimy.gim.http.services.impl`. In the constructor, we will instantiate a ModelDaoService:

```
package org.bimy.gim.http.services.impl;
import /*...*/;

public class ModelServiceImpl implements ModelService {
  private ModelDaoService service;
  private Vertx vertx;

  public ModelServiceImpl(Vertx vertx) {
    this.vertx = vertx;
    this.service              =              ModelDaoService.createProxy(vertx,
AddressConstants.BIMY_SERVICE_MODEL);
  }

  // If extend service, implement mountExceptionalEndPoint method

  // Implement all the endpoints for Model.
}
```

5.1.3.6 Step 6

The last step will be to bind this service to the HTTP verticle and generate all the routes for the HTTP router. In `org.bimy.gim.http.HttpServerVerticle`, write this snippet:

```
new ServiceBinder(this.vertx)
  .setAddress(AddressConstants.HTTP_SERVICE_MODEL)
  .register(ModelService.class, new ModelServiceImpl(this.vertx));
routerFactory.mountServiceInterface(ModelService.class,
AddressConstants.HTTP_SERVICE_MODEL);
modelService.mountExceptionalEndPoint(routerFactory);  //  If  extend  service
interface
```

5.1.4 Adding an endpoint to an existing service

In this example, we will implement the Model.getModelById method using Vert.x Web API Service extension and declared in the OpenAPI specs like so:

```
  /model/{modelId}:
    get:
      tags:
```

```
    - model
  summary: Get model by ID
  description: Get descriptive metadata of a model
  operationId: getModelById
  parameters:
    - name: modelId
      in: path
      description: ID of model
      required: true
      schema:
        type: string
        format: uuid
    - name: versionId
      in: query
      description: version ID of model
      schema:
        type: string
        format: uuid
  responses:
    200:
      description: successful operation
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Model'
    400:
      description: Invalid ID supplied
      content: {}
    404:
      description: Model not found
      content: {}
  security:
    - OAuth2:
      - email
```

5.1.4.1 Step 1

Create a method in `org.bimy.gim.http.services.ModelService.java` interface with the name of the OpenAPI operationId value. This method doesnt have any return value and takes in arguments all the necessary request parameters namely "modelId" and "versionId".

Important: The 2 last argument of this function MUST be:

- OperationRequest: contain the context of the request (e.g. user information)
- Handler<AsyncResult<OperationResponse>>: result handler

For the operationId "getModelById", the declaration in the interface will be:

```
void getModelById(String modelId, String versionId, OperationRequest context,
Handler<AsyncResult<OperationResponse>> resultHandler);
```

5.1.4.2 Step 2

Implement this method in ModelServiceImpl.java.

This method will delegate the job to bimyclient verticle by calling the ModelDaoService proxy and provide a result handler that will resolve with the response of the ModelDaoService:

```
@Override

public void getModelById(String modelId, String versionId, OperationRequest
context, Handler<AsyncResult<OperationResponse>> resultHandler) {

  //Call service from bimyClient veticle

  this.service.getOneById(modelId, versionId, result -> {

    if(result.succeeded()) {

resultHandler.handle(Future.succeededFuture(OperationResponse.completedWithJs
on(result.result())));

    } else {

resultHandler.handle(Future.succeededFuture(OperationResponse.completedWithPl
ainText(Buffer.buffer(result.cause().getMessage()))));

    }

  });

}
```

5.1.4.3 Step 3

In the bimServerClient verticle, create a proxy service function by declaring `getOneById` method in ModelDaoService.java interface:

```
void getOneById(String modelId, String versionId,
Handler<AsyncResult<JsonObject>> result);
```

5.1.4.4 Step 4

Implement this method in ModelDaoServiceImpl.java.

This method will interact with BIMy's api to fetch or modify data.

5.1.4.5 Special cases

Unfortunately, the service function we can define using Vert.x Service Proxy extension and Vert.x Web API Service extension can only take arguments of types within this list: https://vertx.io/docs/vertx-service-proxy/java/#_parameter_types. A work-around still need to be found for files download and upload.

Check `org.bimy.gim.http.services.impl.ModelServiceImpl#mountExceptionalEndPoint` method to see what was already tried as work-around. This method will add endpoints without using the 2 Services proxy extensions but using only Vert.x Web API Contract extension.

5.2 Security: OAuth2 authentication

The BIMy 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. Internally, the API then authenticates with the backend services (e.g. BIMserver, PostgreSQL, FME Server, Autodesk BIM A360). In the future, other OAuth2 authentication providers could be added.

The authentication was implemented using the Vertx OAuth2 authentication provider. See also the Gitlab OAuth2 Authorization code flow. The autorization flow works as follows:

1. **Request authorization code**. To do that, you should redirect the user to the `/oauth2/gitlab/authorize` endpoint: http://localhost:8081/oauth2/gitlab/authorize This will redirect you to gitlab authentication web page. Enter your Gitlab credentials and click "Authorize". See the Gitlab OAuth2 Authorization code flow for further documentation on the request parameters to use.
2. **request an access_token**: Once you have the authorization code you can request an access token using the obtained authorization code on the `/oauth2/gitlab/token` endpoint. See the Gitlab OAuth2 Authorization code flow for further documentation on the form data parameters to use. Once you are authenticated, you will be redirected back (use the redirect_uri parameter) to request an access token (session id):

```
{
  "sessionId": "33f5338ffdd05cabfb14ce4872636efa",
  "access_token":"33f5338ffdd05cabfb14ce4872636efa",
  "expires_in":1800000,
  "token_type":"bearer"}
}
```

You can already do protected requests from your web browser. However, if you want to use an other http client, you need to put this sessionId in a cookie for all your next requests:
`vertx-web.session=<sessionId>; path=/; domain=localhost;`

Logout URL: http://localhost:8081/oauth2/gitlab/logout

Protected endpoint are defined directly in the openAPI spec with the following snippet:

```
security:
  - OAuth2:
    - email
```

It is possible to implement multiple security schemas other than "OAuth2" with other behavior.

5.3 Downloading a model by transformation

To download a model, you need to put its revision oid (e.g. 327683) and a serializer oid (e.g. 1441830) in the path. You also need a query (e.g. IfcWall which gets all walls).

It works by first getting an authorization token from the server using the username and password in the config file. It then asks for a download id. Finally, the download id gets added to the correct url, and the request is redirected to the correct download on the BIMserver.

The requests are made using Vertx's web client.

localhost:8081/model/327683/transform/1441830?queryname=IfcWall

5.4 Documentation

- https://vertx.io/docs/vertx-web/java/
- https://vertx.io/docs/vertx-service-proxy/java/
- https://vertx.io/docs/vertx-web-api-contract/java/
- https://vertx.io/docs/vertx-web-api-service/java/
- https://vertx.io/docs/vertx-web-client/java/
- https://vertx.io/docs/vertx-auth-oauth2/java/
- Gitlab as an OAuth2 provider

6 BIMy Data Transformer

Repository: https://gitlab.com/bimy/bimy-data-transformer

All data transformations that have been implemented in the context of the BIMy project, and that are documented in deliverable D2.2 (BIMy consortium, 2019b, p. 2). These include:

6.1 IFC to floorplan:

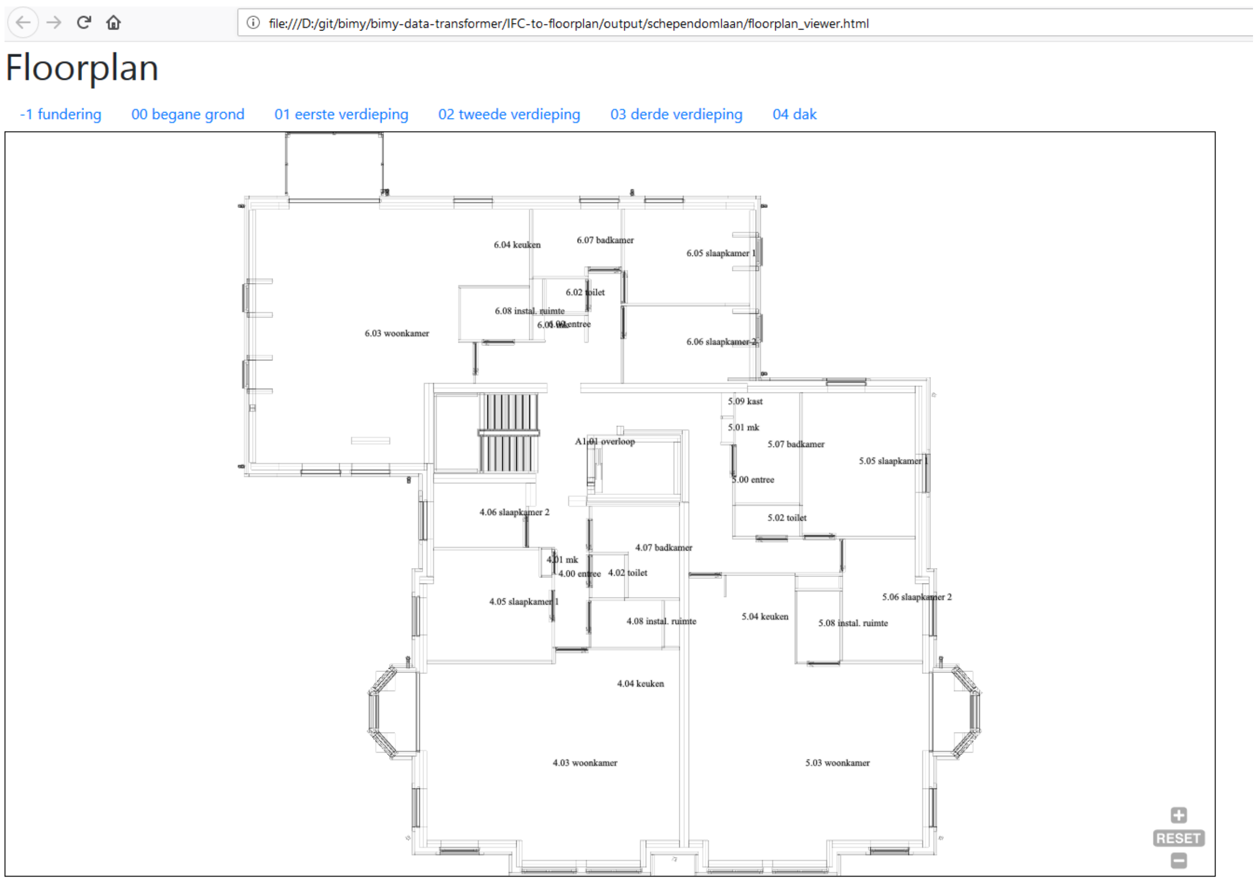
The folder contains an implementation of a transformation that takes an IFC and input and converts that into a geospatial PDF and an SVG on the output.

6.1.1 How to use

Install FME 2019.0 Desktop and run the IFC-to-floorplan.fmw workspace.

6.1.2 Examples

Take a look at the output/schependomlaan and output/willemen-office folders for some sample outputs in PDF and SVG format.



6.1.3 Possible improvements

- Recognise specific objects (e.g. furniture, hydrants, sprinklers) and replace them with a specific symbology.
- Use offical color coding

6.2 IFC to 3D Tiles

6.3 IFC to CityGML

The folder contains an implementation of a transformation that takes an IFC and input and converts that into a CityGML 2.0 LOD4 dataset. The transformation rules are described in detail in deliverable D2.2 Integrated BIM/GIS model of the BIMy project. The implementation is based on the implemenation by Safe Software.

6.3.1 How to use

Install FME 2019.0 Desktop and run the IFC-to-CityGML.fmw workspace.

6.3.2 Examples

Take a look at the output/schependomlaan and output/willemen-office folders for some sample outputs in CityGML format.

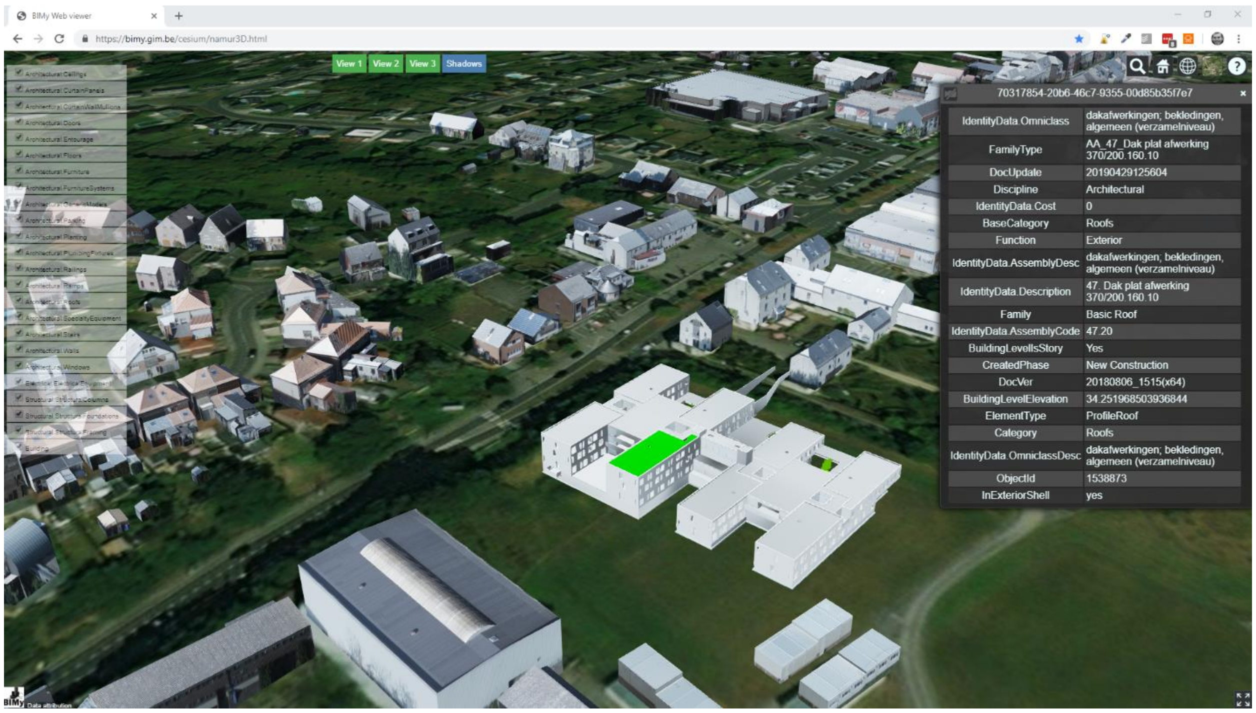
6.3.3 Possible improvements

- Include construction time information (citygml:constructionDate, citygml:demolitionDate)
- Experiment with including geometries with different Level of Detail (LOD).

7 BIMy Web Viewer

repository: https://gitlab.com/bimy/bimy-web-viewer

This repository contains a CesiumJS web viewer to visualise BIM models as 3D Tiles in a geospatial context, together with elevation data and map data.



70317854-20b6-46c7-9355-00d85b35f7e7

IdentityData.Omniclass	dakafwerkingen; bekledingen, algemeen (verzamelniveau)
FamilyType	AA_47_Dak plat afwerking 370/200.160.10
DocUpdate	20190429125604
Discipline	Architectural
IdentityData.Cost	0
BaseCategory	Roofs
Function	Exterior
IdentityData.AssemblyDesc	dakafwerkingen; bekledingen, algemeen (verzamelniveau)
IdentityData.Description	47. Dak plat afwerking 370/200.160.10
Family	Basic Roof
IdentityData.AssemblyCode	47.20
BuildingLevelIsStory	Yes
CreatedPhase	New Construction
DocVer	20180806_1515(x64)
BuildingLevelElevation	34.251968503936844
ElementType	ProfileRoof
Category	Roofs
IdentityData.OmniclassDesc	dakafwerkingen; bekledingen, algemeen (verzamelniveau)
ObjectId	1538873
InExteriorShell	yes

Figure 3 An visualization of the Kortrijk nursing home (ASSAR, 2019)

7.1 Get started

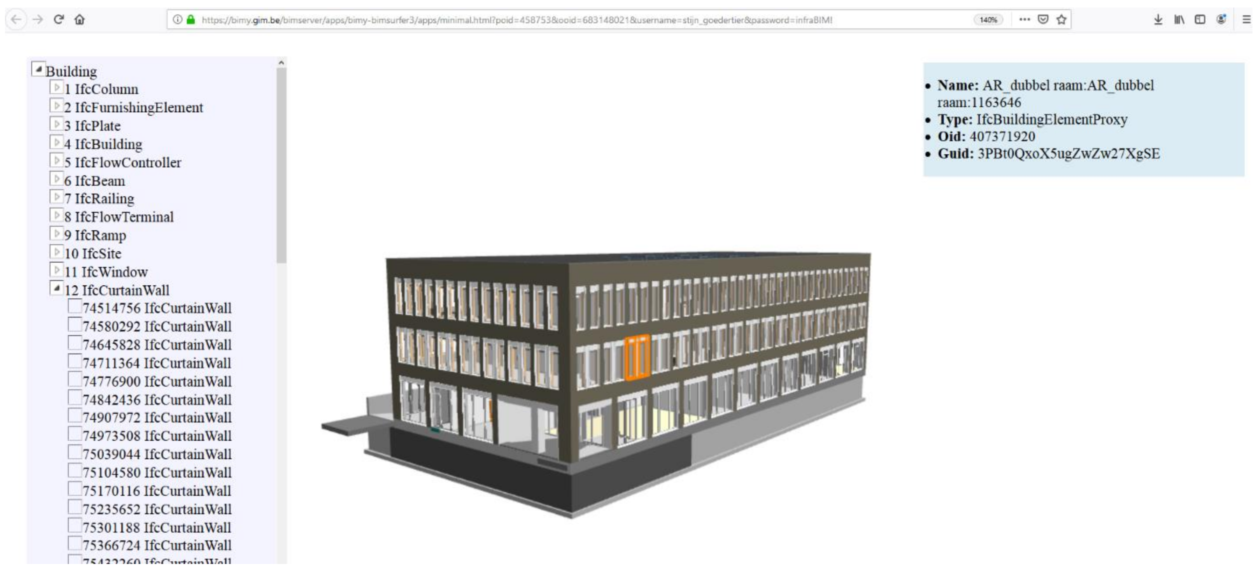
The web content is served using the default NodeJS Express web server app that is shipped with CesiumJS. Install NodeJs. Then, run the following commands to install required packages and start a NodeJs Express Web server.

```
npm install
npm start
```

8 BIMy BIMsurfer

Repository: https://gitlab.com/bimy/bimy-bimsurfer

BIMy plugin for BIMserver based on BIMSurfer 3.



https://bimy.gim.be/bimserver/apps/bimy-bimsurfer3/apps/minimal.html?poid=458753&ooid=683148021&username=stijn_goedertier&password=infraBIM!

- Building
  - 1 IfcColumn
  - 2 IfcFurnishingElement
  - 3 IfcPlate
  - 4 IfcBuilding
  - 5 IfcFlowController
  - 6 IfcBeam
  - 7 IfcRailing
  - 8 IfcFlowTerminal
  - 9 IfcRamp
  - 10 IfcSite
  - 11 IfcWindow
  - 12 IfcCurtainWall
    - 74514756 IfcCurtainWall
    - 74580292 IfcCurtainWall
    - 74645828 IfcCurtainWall
    - 74711364 IfcCurtainWall
    - 74776900 IfcCurtainWall
    - 74842436 IfcCurtainWall
    - 74907972 IfcCurtainWall
    - 74973508 IfcCurtainWall
    - 75039044 IfcCurtainWall
    - 75104580 IfcCurtainWall
    - 75170116 IfcCurtainWall
    - 75235652 IfcCurtainWall
    - 75301188 IfcCurtainWall
    - 75366724 IfcCurtainWall

- **Name:** AR_dubbel raam:AR_dubbel raam:1163646
- **Type:** IfcBuildingElementProxy
- **Oid:** 407371920
- **Guid:** 3PBt0QxoX5ugZwZw27XgSE

Figure 4 Screenshot of the Willemen Office (Willemen, 2018) visualized by BIMSurfer3

8.1 How to build

Build this by packaging it through Maven. You can then import the JAR-file through Bimviews.

8.2 How to use

Based on BIMsurfer 3 0.17

The standard url structure `https://bimy.gim.be/bimserver/apps/bimy-bimsurfer3/apps/minimal.html?poid=1179649&ooid=683148021&username=firstname_lastname&password=password`

8.3 Parameters

8.3.1 poid

This is the project's oid.

8.3.2 selectOid

This is the object's oid. It is *optional*.

8.3.3 selectGuid

This is the object's guid. It is *optional*.

8.3.4 colorGuid

This is the guid you would like to color. It is *optional*.

8.3.5 username

The part before the @ of your GIM mail address. Replace dots with underscores.

For example, john.doe@gim.be should be entered as john_doe.

8.3.6 password

Your password.

8.3.7 type

The IfcType to filter on. It will affect both the rendered model and the tree model. It is *optional*.

8.3.8 tree

Do you want to render a tree model? It is *optional* and defaults to `true`.

8.4 Useful console commands

8.4.1 viewObjectsByType

```
this.bimServerViewer.viewer.viewObjectsByType
```

Gives you a list of object, sorted be types. Useful to find another Object ooid to use.

8.5 Notes

Models from BIMserver are loaded in through the various methods in bimserverviewer.js.

9 BIMy Docker

Repository: https://gitlab.com/bimy/bimy-docker

This project contains Dockerfiles and/or Docker images of the the BIMy platform and its applications. The objective is to have a continuous integration/continuous deployment pipeline for the BIMy project.

9.1 About Docker

Docker is an open-source project that allows automating the deployment of software applications inside **containers** by providing an additional layer of abstraction and automation of **OS-level virtualization** on Linux. The key benefit of Docker is that it allows users to package an application with all of its dependencies into a standardized unit for software development. Unlike virtual machines, containers do not have the high overhead and hence enable more efficient usage of the underlying system and resources.

9.2 How it works

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

A BIMserver Dockerfile on the repository (https://gitlab.com/bimy/bimy-docker) describes how to build a BIMserver images.

```
FROM tomcat:8.5-jre8


ENV TOMCAT_HOME=/usr/local/tomcat \

    CATALINA_HOME=/usr/local/tomcat \

    JAVA_OPTS="-Djava.awt.headless=true -
Djava.security.egd=file:/dev/./urandom" \

    CATALINA_OPTS="-Xms512M -Xmx1024M -server -XX:+UseParallelGC"


ARG BIMSERVER_VERSION

ENV BIMSERVER_VERSION=${BIMSERVER_VERSION} \

    BIMSERVER_APP=${TOMCAT_HOME}/webapps/bimserver



# Add BIMserver java web archive

RUN set -x && \
curl -fSL
https://github.com/opensourceBIM/BIMserver/releases/download/v${BIMSERVER_VER
SION}/bimserverwar-${BIMSERVER_VERSION}.war -o $BIMSERVER_APP.war && \
    unzip $BIMSERVER_APP.war -d $BIMSERVER_APP && \
rm $BIMSERVER_APP.war

# Add roles and increase file size for webapps to 500Mb
ADD ./tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml
ADD ./web.xml /usr/local/tomcat/webapps/manager/WEB-INF/web.xml
ADD ./run.sh /run.sh
RUN chmod a+x /run.sh

EXPOSE 8080

COPY docker-entrypoint.sh /
RUN chmod a+x /docker-entrypoint.sh
ENTRYPOINT ["/docker-entrypoint.sh"]

CMD ["bimserver"]
```

9.3 Run with Docker Compose

Install a Docker host following the Docker CE installation tutorial. Clone this git repository and run the following command to build the required Docker images and deploy the containers:

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

Bibliography

BIMy consortium. (2019a). *BIMy - BIM in the City - D1.3 Platform and API specifications*.

BIMy consortium. (2019b). *BIMy - BIM in the City - D2.2 Integrated BIM/GIS model*.

Willemen. (2018). *BIM model of the Kumpen Office in Hasselt*.