COGITO

CONSTRUCTION PHASE DIGITAL TWIN MODEL

cogito-project.eu

D3.7 – Visual Data Pre-processing Module v1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310

D3.7 – Visual Data Pre-processing Module v1

Dissemination Level:	Public
Deliverable Type:	Demonstrator
Lead Partner:	CERTH
Contributing Partners:	UCL, UEDIN, NT
Due date:	28-02-2022
Actual submission date:	28-02-2022

Authors

Name	Beneficiary	Email
Thanos Tsakiris	CERTH	atsakir@iti.gr
Apostolia Gounaridou	CERTH	agounaridou@iti.gr
Vasilis Karkanis	CERTH	vkarkani@iti.gr
Michalis Chatzakis	CERTH	mchatzak@iti.gr
Anastasios Sinanis	CERTH	sinanis@iti.gr
Frederic Bosche	UEDIN	f.bosche@ed.ac.uk
Enrique Valero	UEDIN	e.valero@ed.ac.uk
Kyriakos Katsigarakis	UCL	k.katsigarakis@ucl.ac.uk
Georgios N. Lilis	UCL	g.lilis@ucl.ac.uk
Dimitrios Rovas	UCL	d.rovas@ucl.ac.uk
Martin Straka	NT	straka@novitechgroup.sk

Reviewers

Name	Beneficiary	Email
Giorgos Giannakis	Hypertech	g.giannakis@hypertech.gr
Tobias Hanel	Ferrovial	thanel@ferrovial.com

Version History

Version	Editors	Date	Comment
0.1	CERTH	28.01.2022	Table of Contents
0.2	CERTH	31.01.2022	Draft version of 1.2, 1.3, 2.1, 2.2, 2.3, 2.6, 2.8
0.3	CERTH	03.02.2022	Draft version of 2.7, 2.10, 2.11
0.4	CERTH	15.02.2022	Draft version of 1.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9
0.5	UEDIN	17.02.2022	2.1.8, 2.3.1, 2.3.2
0.6	CERTH	22.02.2022	2.1 changes, 2.4 changes, overall update
1.0	CERTH, Hypertech, Ferrovial	28.02.2022	Submission to the EC

Disclaimer

Executive Summary

The COGITO Deliverable D3.7 “Visual Data Pre-processing Module v1” documents the first version of the COGITO Visual Data Pre-processing Module and presents the development activities concerning the T3.5 “Visual Data Pre-processing Module”. Overall, the Visual Data Pre-processing Module aims to receive, prepare and enhance the raw visual data (images and point clouds) acquired onsite, in order to deliver it to the relevant COGITO components for further exploitation. The raw data is first stored locally, enhanced and linked with the relevant Industry Foundation Classes (IFC) components and specific pre-processing jobs. It is then sent to the Digital Twin (DT) Platform for digesting by the involved components. Therefore, the Visual Data Pre-processing Module considers being one of the main modules of the COGITO solution.

As aforementioned, the Visual Data Pre-processing Module manages the visual data acquired onsite. It is in charge of receiving and handling the point clouds and the 2D images sent by laser scanners and static cameras, mobile phones, Augmented Reality (AR) goggles respectively. It consists of a Graphical User Interface (GUI) where the user can interact with the application, add fields and new information, as well as link the IFC components with a capturing device and specific visual data. The Visual Data Pre-processing Module is composed of three sub-modules, the main one being the Main Pre-processing sub-module, the second one the Pre-processing sub-module for 2D Visual Data and the third one the Pre-processing sub-module for Geometric Data. The Main Pre-processing sub-module contains the Device Manager, the Job Generator, the IFC Element Connector sub-modules and the IFC 3D Viewer. The Device Manager sub-module is in charge of directing the capturing devices connected with the Visual Data Pre-processing tool. The Job Generator and the IFC Element Connector sub-modules are responsible for creating and handling individual jobs or jobs related to a specific device and for linking the IFC elements with a created job respectively. Finally, the IFC 3D Viewer helps to visualize the BIM model obtained from the DT Platform. The Pre-processing sub-module for 2D Visual Data consists of two sub-modules concerning the visual data processing; the Filter Implementation sub-module and the 2D Visual Data Viewer. The former module applies visual filters to the raw data (2D image), processes it and returns it in an enhanced form (cropped, rescaled etc.). In addition, with the latter module, the user can preview the raw and processed 2D images, in order to discard them or select and send them further for Quality Control (QC) detection. Regarding the Pre-processing sub-module for Geometric Data, it is in charge of uploading and enhancing the point clouds in order to be sent for Geometric quality control.

The present documentation of the COGITO Visual Data Pre-processing Module, along with its sub-components, is oriented towards the functionalities they broadly deliver, the technology stacks they build upon, the inputs, outputs and APIs they expose, the installation instructions, the assumptions and restrictions, the applications examples, the development and integration status, and the requirements coverage. In this first release, several basic functionalities are implemented focusing on the Main Pre-processing sub-module and the Pre-processing sub-module for 2D visual data (i.e., data storage, additional data, and basic visual filters’ implementation). Furthermore, in the second release more functionalities will be implemented with regards to 2D image pre-processing refinement as well as the Pre-processing sub-module for Geometric Data (point clouds uploading).

Table of contents

Executive Summary 3
Table of contents 4
List of Figures 5
List of Tables 6
List of Acronyms 7
1 Introduction 8
1.1 Scope and Objectives of the Deliverable 8
1.2 Relation to other Tasks and Deliverables 8
1.3 Structure of the Deliverable 9
2 Visual Data Pre-processing Module 10
2.1 Overall Architecture of the Visual Data Pre-processing Module 10
2.1.1 Main Pre-processing sub-module 11
2.1.2 Local Database 12
2.1.3 Sub-module for 2D Visual Data 13
2.1.4 Sub-module for Geometric Data 14
2.2 Technology Stack and Implementation Tools 14
2.3 Input, Output and API Documentation 16
2.3.1 Input Data 16
2.3.2 Output Data 16
2.3.3 API Documentation 16
2.4 Usage Walkthrough 23
2.5 Licensing 29
2.6 Installation Instructions 30
2.7 Development and Integration Status 30
2.8 Requirements Coverage 30
2.9 Assumptions and Restrictions 31
3 Conclusions 32
References 33

List of Figures

Figure 1 - Overall architecture of the Visual Data Pre-processing Module 10
Figure 2 – Adding a device and IFC mapping 11
Figure 3 - Adding new job's properties and components' attribution 12
Figure 4 - Entity Relationship Diagram - MySQL database 13
Figure 5 - Add a new job: filter implementation and finalization 14
Figure 6 - API endpoint for adding a device- request and response 18
Figure 7 - API endpoint for getting all the devices- request and response 19
Figure 8 - API endpoint for adding a job – request and response 21
Figure 9 - API endpoint for uploading the raw photo of a job 22
Figure 10 - API endpoint for uploading the processed photo of a job 22
Figure 11 - API endpoint for getting all the jobs- request and response 23
Figure 12 - Visual Data Pre-processing Module user's login page 24
Figure 13 - Project Selection 24
Figure 14 - Account options 25
Figure 15 - Bulk actions button for grouping the devices 25
Figure 16 - Applying search filters 26
Figure 17 - Add a new capturing device 26
Figure 18 - IFC Mapping of the new device 27
Figure 19 - Jobs tab for a specific device 27
Figure 20 - Add a new capturing job 28
Figure 21 - Components attribution to a new capturing job 28
Figure 22 - Image processing and filter implementation 29
Figure 23 - Submitting a job to the DT Platform 29

List of Tables

Table 1 – Libraries and Technologies used in Visual Data Pre-processing Module 14
Table 2 - Device requests 17
Table 3 - Device properties 18
Table 4 - Sub-properties of position 18
Table 5 - Sub-properties of orientation 19
Table 6 - Job requests 20
Table 7 - Job properties 21
Table 8 - Visual Data Pre-processing tool: Stakeholders' requirements coverage from D2.1 30
Table 9 - Visual Data Pre-processing tool: Functional and Non-Functional Requirements coverage from D2.4 . 30

List of Acronyms

Term	Description
API	Application Programming Interface
BIM	Building Information Modeling
COGITO	Construction Phase Digital Twin mOdel
DigiTAR	Digital Twin visualisation with Augmented Reality
DT	Digital Twin
GUI	Graphical User Interface
IFC	Industry Foundation Classes
PM	Project Manager
QC	Quality Control
SM	Site Manager
UAV	Unmanned aerial vehicle
UID	Unique Identifier
URL	Uniform Resource Locator
WODM	Work Order Definition and Monitoring

1 Introduction

1.1 Scope and Objectives of the Deliverable

This deliverable reports on the work conducted from M7 to M16 on the Visual Data Pre-processing Module that is developed as part of T3.5. The scope of this module is to enrich and enhance the visual data acquired onsite in order to prepare and deliver it, through the DT Platform, to the relevant COGITO components (such as the GeometricQC and the VisualQC tool) for performing automatic QC compliance.

More specifically, this deliverable reports on the development of the first release of the Visual Data Pre-processing Module focusing on its main sub-modules listed below:

- **Main Pre-processing sub-module:** hosts the main general functionalities and actions during the pre-processing step. This sub-module takes as input the as-designed data (IFC 3D model) and the visual data acquired on site (2D images and point clouds). It then embeds all the relevant information (location, orientation of capturing devices, addition of new devices and new jobs), as well as the link between the relevant IFC elements and the new pre-processing jobs.
- **Pre-processing sub-module for 2D Visual Data:** is in charge of handling/processing the raw 2D visual data (images) acquired onsite. . The user can apply visual filters on this data and finally the enhanced processed data is delivered to the DT Platform for further exploitation.
- **Pre-processing sub-module for Geometric Data:** is in charge of handling the geometric visual data acquired onsite. This sub-module takes as input the, already registered, as-built geometric data, taking into account the BIM model coordinate frame. Finally, the data is delivered to the DT Platform for further exploitation.

These sub-modules provide the core functionalities of the Visual Data Pre-processing Module.

1.2 Relation to other Tasks and Deliverables

T3.5 “Visual Data Pre-processing Module” and consequently D3.7 “Visual Data Pre-processing Module v1” are related to the following COGITO tasks and deliverables:

- The first version of the COGITO architecture in the corresponding deliverable “D2.4 COGITO System Architecture v1” provided an overview on the Visual Data Pre-processing module, its requirements and the communication to other components.
- The Visual Data Pre-processing Module, similarly to all components, relies on a shared ontology and a common data model developed within T3.2 “COGITO Data Model, Ontology Definition and Interoperability Design”; the first version of COGITO ontologies and data models have been documented in D3.2 “COGITO Data Model, Ontology Definition and Interoperability Design v1”.
- The Visual Data Pre-processing Module provides through the DT Platform the acquired data (point clouds) for Geometric Quality Control to the relevant module (D5.1 “Scan-vs-BIM Geometric Quality Control v1”).
- The Visual Data Pre-processing Module provides, through the DT Platform, the acquired data (2D images) for Visual Quality Control to the relevant module (D5.3 “Deep Learning Image Processing for Visual Quality Control v1”).
- The DigiTAR module (D5.7 “User interface for Construction Quality Control v1”) communicates with the Visual Data Pre-processing Module to apply filters in 2D images captured by Microsoft Hololens[1] onsite, preview and finalize the processed data that is finally sent for Visual Quality Control.
- The DT Platform accommodates structured data (e.g. scanned data, images) provided by the Visual Data Pre-processing Module in its database and further forwards it to other modules (D7.3 “Extraction, Transformation & Loading Tools and Model Checking v1”).

[1] Mixed reality smartglasses - head-mounted display

1.3 Structure of the Deliverable

The rest of the deliverable focuses on the Visual Data Pre-processing Module.

- Sub-section 2.1 presents the overall architecture of the Visual Data Pre-processing Module, introducing its sub-components and its workflow diagrams.
- Sub-section 2.2 describes the technologies, libraries and tools exploited for the implementation of this specific module.
- Sub-section 2.3 notes the inputs and outputs of the component as well as the APIs documentation.
- Sub-section 2.4 provides a brief manual (guidelines) on how to use this specific component.
- Sub-section 2.5 refers to information about the source code repository, the delivery form and the license of the component.
- Sub-section 2.6 describes how the Visual Data Pre-processing Module is accessible by the user.
- Sub-section 2.7 presents the current status of the component and provides a plan regarding its second release.
- Sub-section 2.8 notes the functional, non-functional and stakeholder requirements that are covered by the component.
- Sub-section 2.9 describes the assumptions already made for this component, as well as restrictions and measures that will be taken into consideration for the final release.
- The document concludes with Section 3, where the progress, the next steps and the contribution to the overall COGITO objectives are being reported.

2 Visual Data Pre-processing Module

In this section, the Visual Data Pre-processing Module is presented in detail. The overall architecture of the tool is presented, as well as the different sub-components of the tool. In addition, relevant information about the implementation tools, the current status etc. are described in the next sub-sections.

2.1 Overall Architecture of the Visual Data Pre-processing Module

The Visual Data Pre-processing component is in charge of performing pre-processing over visual and geometric inputs, delivering them in an enhanced form to related components (the Geometric and Visual QC tools) through DT Platform. First, as-built data is captured onsite from all available sources (AR goggles, multimodal UAV-mounted cameras, laser scanners). Then raw data is registered with structural and geometric details and different filters are afterwards implemented (contrast, brightness, cropping, rescaling etc.) to enhance the data quality. The new processed data is finally provided to the DT Platform for further exploitation by the relevant tools.

The overall architecture of the Visual Data Pre-processing tool is presented in Figure 1. The tool communicates with the DT Platform for data exchanging and with several Data Acquisition tools (capturing devices) that send relevant data for pre-processing. DigiTAR tool also operates as a capturing device, as long as it sends the captured data in the Visual Data Pre-processing tool. More specifically, DigiTAR is also considered as a GUI application of the Visual Data Pre-processing tool onsite. Therefore, a two-way communication is established between these two tools, while both of them receive and send data to each other. All the other capturing devices will just send the acquired data to the module; hence, a one-way communication is established.

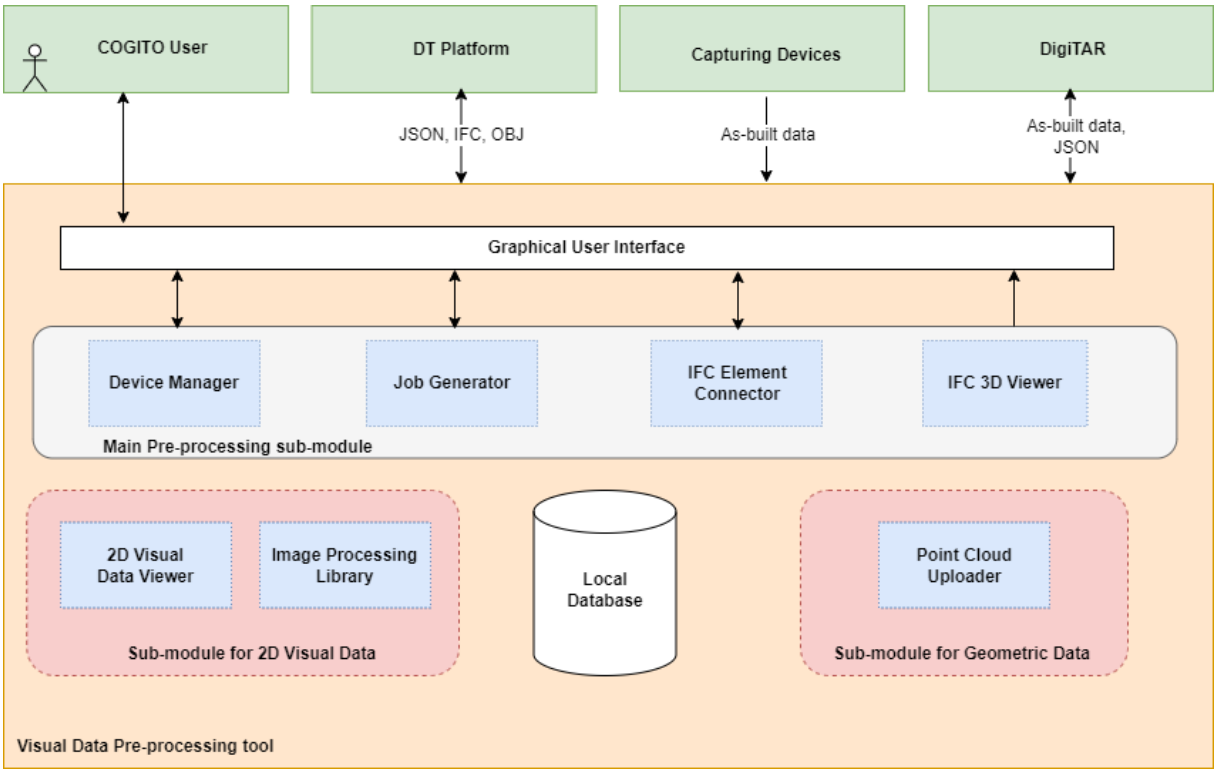


Figure 1 - Overall architecture of the Visual Data Pre-processing Module

The user interacts with the Visual Data Pre-processing tool through a GUI. Based on the related functionalities, the Visual Data Pre-processing Module consists of (1) a Main sub-module, (2) a Local Database to store the data temporarily, and (3) two separate sub-modules that are in charge of handling the 2D images (2D Visual Data) and the point clouds (Geometric Data). All the aforementioned sub-modules will be further presented in the next sub-sections.

2.1.1 Main Pre-processing sub-module

2.1.1.1 Device Manager sub-module

The first step is to define the device and its relevance to a new task. The user should first ensure that the desired capturing device is registered. If not, the new capturing device should be added, as well as its properties (Name, Type, URL). The capturing devices can be managed by selecting the specific tab in the GUI.

In addition, when a new device is added, the next step is to locate it within the 3D space. By selecting the relevant project, the related IFC 3D BIM model is loaded from the DT Platform. The user can then visualize it online, to declare the position and the orientation of this specific device. This way, the as-built data acquired by different devices, is combined with specific location data. All the produced information concerning the device properties is stored in the local database of the Visual Data Pre-processing tool in a JSON format file. Figure 2 presents the addition of a new capture device and its metadata.

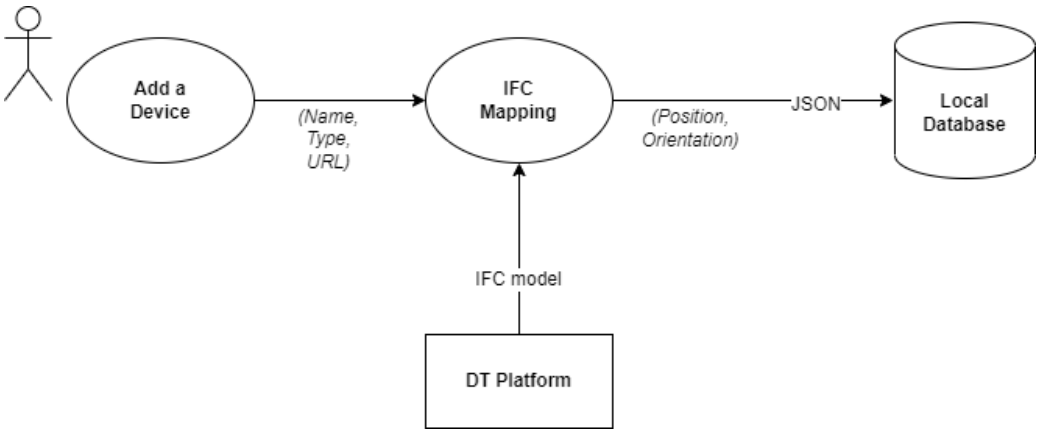


Figure 2 – Adding a device and IFC mapping

2.1.1.2 Job Generator sub-module

After ensuring that the involved capturing device is added, the user can associate the device and the captured data with a new pre-processing job. Based on the list of working orders and completed tasks provided by the WODM tool, the relevant stakeholder can create a new pre-processing job. At this step, it is necessary to define relevant properties such as Name, Format and Frequency as well as the involved capturing device (see Figure 3).

2.1.1.3 IFC Element Connector sub-module

The next step is the IFC elements registration. At this step, based on the information provided by the WODM component, the user is able to select from a list the IFC components that will be related with this specific job, and assign to them the as-build data. Hence, the selected components are linked semi-automatically to the new capture job (components attribution) and the produced information is stored in the local database in a JSON format. Figure 3 illustrates both the two aforementioned steps (adding a new capture job and link the involved IFC elements).

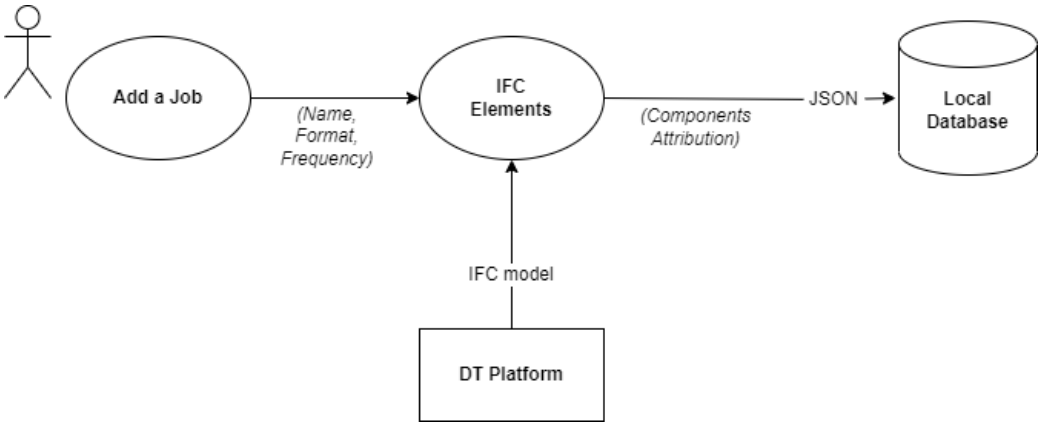


Figure 3 - Adding new job's properties and components' attribution

2.1.1.4 IFC 3D Viewer sub-module

Within the Visual Data Pre-processing Module, it is necessary to display the corresponding 3D BIM model, to declare the position and the orientation of the capturing device and its specific coordinates. Therefore, a 3D Viewer is needed in order to visualize online the as-designed BIM model. The Viewer will be integrated in the second release of the module.

2.1.2 Local Database

Within the Visual Data Pre-processing module, a local database is developed to store all the necessary information. Currently the data is stored and related to devices and their jobs. As it was mentioned before, the devices refer to the capturing devices (RGB cameras, Laser Scanners etc.) that are placed in the construction sites in order to provide as-built data. After adding a capturing device, a job can be generated and associated with a specific device, in order to define the type of data expected to be captured. When creating a new job, related to a specific capturing device, information about the capturing format (2D image or point cloud), the file type (jpg, png, etc.), the capturing frequency (one time only, hourly, daily, etc.) as well as the capturing start/end date are set and stored to the database.

The database created is a MySQL database [1] and its structure is illustrated in the Entity Relationship Diagram (Figure 4). This database considers well-structured and ideal for diverse user permissions while the data can be stored and retrieved easily [2], [3]. At this point, it is worth mentioning that a SQL database is perfect for quantitative data, automated assistance for the customer, for automation of the internal process and finally for applications where the data integrity is indispensable [3], [4].

Regarding the constructed database, each device has some properties as well as a position and orientation (one-to-one relationship) that are digital representations of the physical position and orientation of the corresponding device. Furthermore, as illustrated in Figure 4, one device can have multiple jobs (one-to-many) with different properties. In addition, each job is responsible for receiving data from the physical device and storing it in the file system. Thus, the paths of the raw image and the processed image are stored in the job table. Regarding the image processing, multiple image processors can be applied to an image related to a job. The processor types already implemented are brightness modification, contrast modification and Gaussian blurring. In the second release of the tool, more image processors will be implemented, able to accept multiple parameters, such as cropping and rescaling.

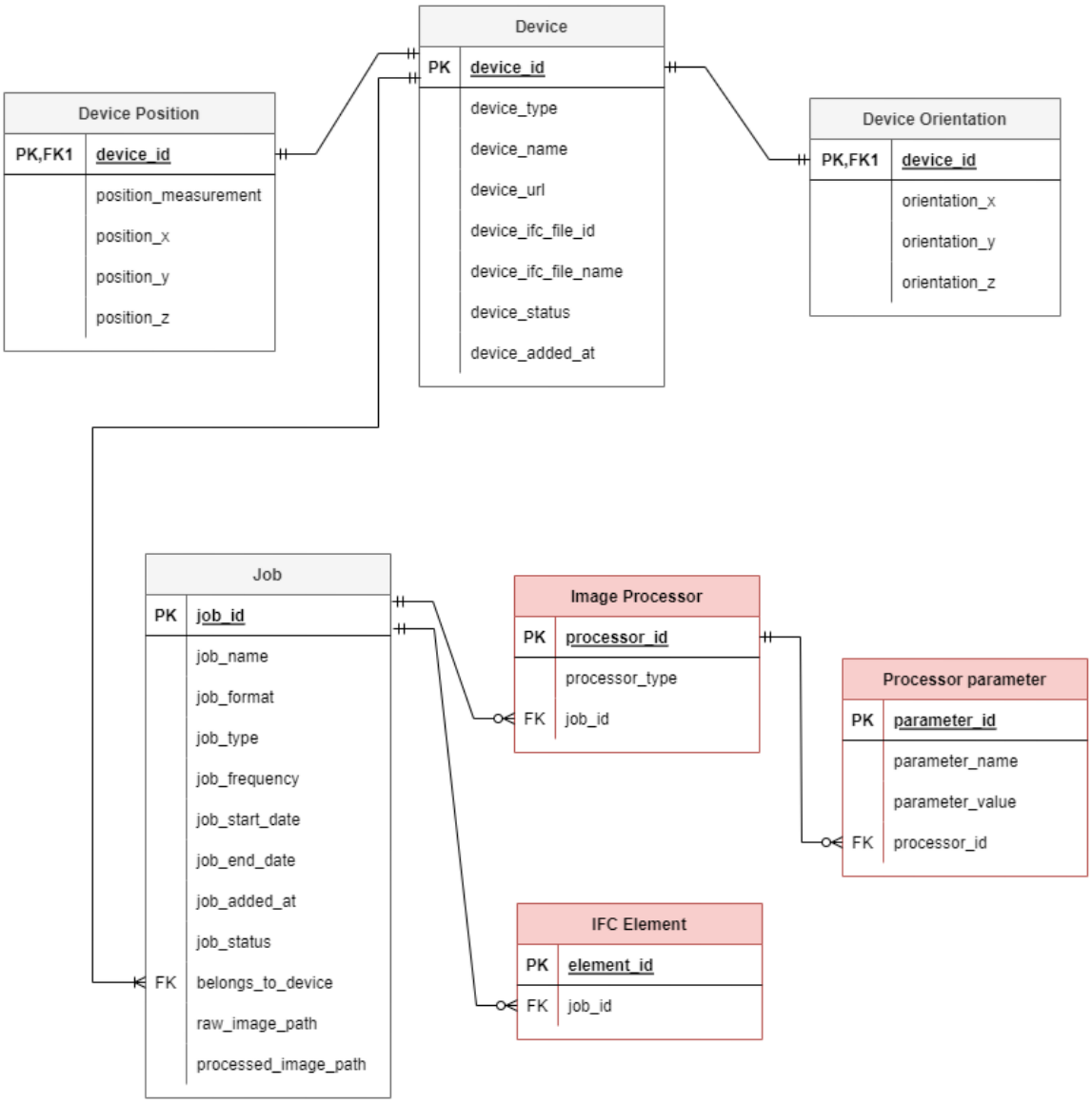


Figure 4 - Entity Relationship Diagram - MySQL database

2.1.3 Sub-module for 2D Visual Data

The Pre-processing sub-module for 2D Visual Data consists of two sub-modules concerning the visual data processing; the Filter Implementation sub-module and the 2D Visual Data Viewer. The former module applies visual filters to the raw data (2D image), processes it and returns it in an enhanced form (cropped, rescaled etc.). In addition, with the latter module, the user can preview the raw and processed 2D images, in order to discard them or select and send them further for Quality Control (QC) detection. Both the sub-modules are analysed in the next sub-sections.

2.1.3.1 2D Visual Data Viewer sub-module

Within the Visual Data Pre-processing module, it is necessary to display the 2D images in order to select and apply the desired filters to enhance the raw image data. Hence, an image viewer was developed to provide to the user the ability to apply filters and preview the processed data.

2.1.3.2 *Image Processing Library sub-module*

A basic step in the case of 2D visual data is the image processing. The user selects the desired raw data (2D captured image) for pre-processing, applies the relevant filters and then evaluates the result (processed image). Once the result is acceptable and the job is finalized , the raw and processed datasets are sent to the DT Platform (through the DT Platform API) to be used as input to the VisualQC data model. The DT Platform stores internally the image data and returns to the Visual Data Pre-processing Module a corresponding path. The metadata, including the information produced in the last two steps (job properties etc.) as well as the above path provided by the DT Platform, is also sent to the DT Platform for further exploitation in a JSON format. Figure 5 illustrates the basic workflow for applying filters and produce processed data in the Visual Data Pre-processing tool.

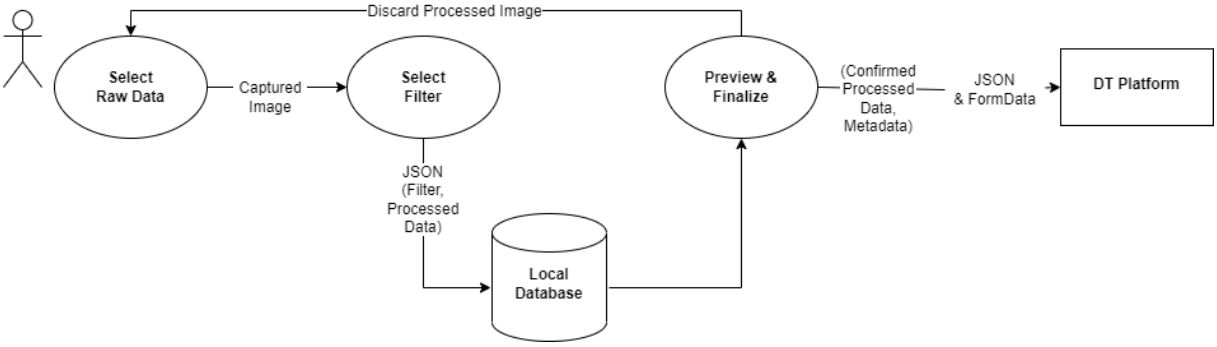


Figure 5 - Add a new job: filter implementation and finalization

2.1.4 Sub-module for Geometric Data

The Pre-processing sub-module for Geometric Data is in charge of uploading and enhancing the point clouds in order to be sent for Geometric quality control. Its sub-module is further analysed in the next sub-section.

2.1.4.1 *Point Cloud Uploader sub-module*

The Visual Data Pre-processing Module also includes the option to allow the user to provide the as-build geometric data. The user receives a working order from the WODM tool, containing a list with the IFC elements that need to be checked by the GeometricQC tool. The relevant stakeholder is in charge of capturing the relevant point clouds covering the current geometry of the structural components of interest and uploading them to the Visual Data Pre-Processing Module. The Point Cloud Uploader sub-module allows the user to upload each of the point cloud files and associate them to the project and the working order. The point cloud pre-process involves the registration of the multiple captured point clouds (in case of multiple scans), cleaning of undesired clutter data, and placing of it in the same coordinate frame as the BIM model. Once the pre-processing step is finalized, the as-built geometric data (in point cloud format) is ready to be uploaded into the COGITO system. Once all the files have been received, the DTP is ready to distribute all the necessary information to the GeometricQC tool to start the dimensional and geometric quality control tasks.

2.2 Technology Stack and Implementation Tools

The Visual Data Pre-processing Module is developed from scratch and programmed utilizing: Node.js for the backend development; Angular for the User Interface development; and MySQL as a relational database for storing the necessary data. Furthermore, several libraries and packages are used for the development of the application. These are briefly described here:

Table 1 – Libraries and Technologies used in Visual Data Pre-processing Module

Library/Technology Name	Version	License
Node.js	12.22.7	MIT License
Angular	12.1.0	MIT License
MySQL	8.0.27	GPLv2
npm	6.14.15	Artistic License 2.0
express	4.17.2	MIT License

CORS	2.8.5	MIT License
multer	1.4.4	MIT License
opencv4nodejs	5.6.0	MIT License
socket.io	4.4.1	MIT License
ngx-admin	8.0.0	MIT License
nebular	8.0.0	MIT License

Node.js is an open-source, cross-platform, back-end Javascript runtime environment that runs on the V8 engine and executes Javascript code outside a web browser. Node.js allows the creation of Web servers using a collection of modules that handle various core functionalities. Node.js's modules use an API designed to reduce the complexity of writing server applications [5].

Angular is a development platform for building mobile and desktop web applications using Typescript/Javascript and other languages. It is and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is used as the frontend [6].

MySQL is an open-source relational database management system that is widely used. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data [7].

npm is a package manager for the Javascript programming language and the default package manager for Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website [8].

express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications and it is designed for building APIs. It is a Node.js module available through the npm registry [8].

CORS (Cross-Origin Resource Sharing) is an HTTP-header base mechanism that allows a server to indicate any origins (domain, scheme or port) other than its own from which a browser should permit loading resources [8].

Multer is a node.js middleware for handling multipart/form-data[2], which is primarily used for uploading files. Multer will not process any form which is not multipart (multipart/form-data) [8].

opencv4nodejs allows you to use the native OpenCV library in nodejs. Besides a synchronous API the package provides an asynchronous API, which allows you to build non-blocking and multithreaded computer vision tasks. opencv4nodejs supports OpenCV 3 and OpenCV 4. OpenCV is a library of programming functions mainly aimed at a real-time computer-vision [8].

Socket.IO is a JavaScript library for real time web applications. It enables real time, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser and a server-side library for Node.js. Both components have a nearly identical API. Like Node.js, it is event-driven. It can be installed with the npm package manager. Socket.io library is used for the real time filtering of the images [8].

ngx-admin is a front-end admin dashboard template based on Angular 9+, Bootstrap 4+ and Nebular. This template comes with lots of popular UI components with a unified color scheme, plus it is based on a modern Angular framework and has a flexible component-based structure. Ngx-admin was used for a faster kick off for the frontend development [9].

Nebular is a customizable Angular UI library that contains 40+ UI components, four visual themes, and Auth and Security modules. Recognized at the prestigious AngularConnect 2018, this Angular framework allows focusing on beautiful designs to adapt them to your brand. Nebular is free of charge and open-source [9].

[2] https://developer.mozilla.org/en-US/docs/Web/API/FormData

2.3 Input, Output and API Documentation

The Visual Data Pre-processing Module will basically interact with the DT Platform for the data exchange. In addition, it will communicate with several data acquisition tools such as laser scanners and static cameras, which will send to it as-build data captured on site. Finally, the DigiTAR component will act as the Visual Data Pre-processing Module's GUI onsite, in order to generate new jobs and send visual data captured onsite for pre-processing.

2.3.1 Input Data

The Visual Data Pre-processing Module in case of the geometric data would require as input the following files:

- **Project ID:** the project ID that the as-built data belongs to.
- **Work Order ID:** the working order ID to associate the point clouds and extract the list of components that are ready for quality control.
- **Captured date:** date that the survey is performed to associate the point clouds to a current project snapshot. Along the entire project lifespan there might be several point clouds associated to the structural components (captured at different stages) so it is necessary to identify them at processing time.
- **As-built point cloud data:** point cloud file(s) (e.g. E57/PLY format) of the as-built data already registered and is/are in the same coordinate frame as the BIM model that needs to be processed.

The Visual Data Pre-processing Module in case of the 2D image data would require as input the following files:

- **Project ID:** the project ID that the as-built data belongs to.
- **Work Order ID:** the working order ID to associate the image data and extract the list of components that are ready for quality control.
- **IFC file:** BIM model exported in IFC file format. The Visual Data Pre-processing Module will visualize the IFC model in a 3D Viewer in order to connect data acquisition tools with the as-designed data and the as-build data (IFC components' attribution).
- **As-built 2D image data:** images (i.e. .png, .jpeg) of the raw data that needs to be processed.

2.3.2 Output Data

The Visual Data Pre-processing Module in case of the geometric data would provide as an output the following files:

- **Transformed as-built point cloud data**: the uploaded point cloud file(s) (e.g. E57/PLY format) of the as-built data that will be exploited for Geometric Quality Control.
- **Metadata:** text file containing all the relevant information accompanying the processed point cloud (e.g. JSON format).

The Visual Data Pre-processing Module in case of the 2D image data would provide as an output the following files:

- **Processed as-built 2D image data:** processed images (i.e. .png, .jpeg) of the as-built data that will be exploited by the Visual Quality Control tool.
- **Metadata:** text file containing all the relevant information accompanying the processed image (e.g. JSON format).

2.3.3 API Documentation

Regarding the API, a first version is implemented in this first release of the Visual Data Pre-processing Module. The API is used for storing, retrieving, updating and deleting data regarding devices and the related jobs [10].

Several requests for both the **Device** and **Job** cases are built and tested in Postman API Platform[3]. A few examples are presented in the next sub-sections.

2.3.3.1 Device Requests

The Device requests that can be made are the following and summarized in Table 2:

- Add a device
- Get data of all devices
- Get data of all devices from a specific IFC file
- Get data of a specific device using its ID
- Update a specific device
- Delete a specific device

Table 2 - Device requests

Request	Description	Method	Endpoints	Response
Add a device	API endpoint to add a device	POST	**POST** localhost:3000/devices	JSON
Get data of all devices	API endpoint to get all the data from the devices.	GET	**GET** localhost:3000/devices	JSON
Get data of all devices from a specific IFC file	API endpoint to get all the data of the devices from a specific IFC file. *(Needs the ifcFileId to be passed as a parameter in the JSON request body.)*	GET	**GET** localhost:3000/devices	JSON
Get data of a specific device using its ID	API endpoint to get the data from a specific device.	GET	**GET** localhost:3000/devices/{{:id}}	JSON
Update a specific device	API endpoint to update the data of a specific device.	PUT	**PUT** localhost:3000/devices/{{:id}}	JSON
Delete a specific device	API endpoint to delete a specific device	DELETE	**DELETE** localhost:3000/devices/{{:id}}	JSON

Example 1: Add a device

The API endpoint for adding a device is presented in Figure 6. A successful registration returns HTTP 200 Status and the id created by the database for the newly registered device is displayed on the screen.

[3] https://www.postman.com/

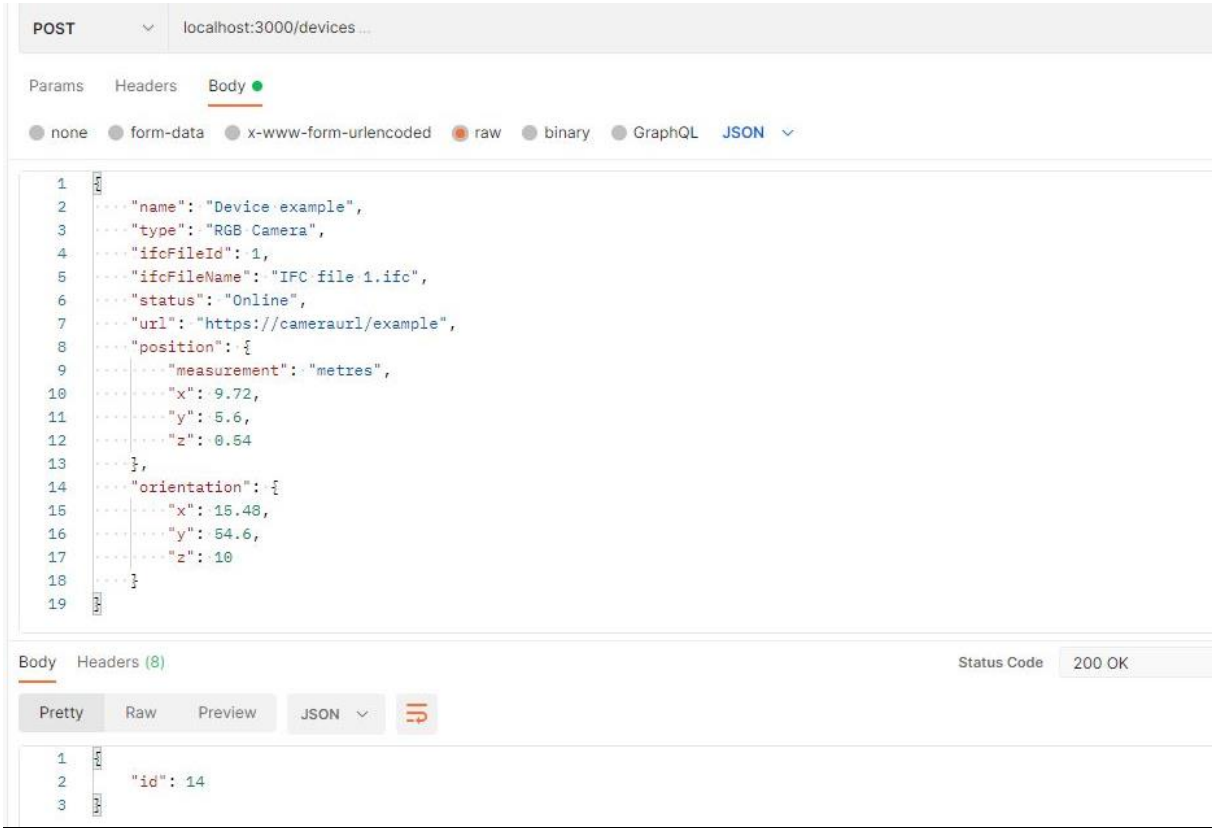


Figure 6 - API endpoint for adding a device- request and response

The properties that should be passed in the JSON body request are presented in Table 3.

Table 3 - Device properties

Property	Description
name	The name of the device
type	The type of the device (RGB Camera, Laser Scanner...)
ifcFileId	The unique identifier for the IFC file to which the device belongs
ifcFileName	The name of the IFC file
status	*Online* or *Offline*
url	The IP address of the device if it is able to stream data
position	The position of the device
orientation	The orientation of the device

The position property has the sub-properties illustrated in Table 4.

Table 4 - Sub-properties of position

Position property	Description
measurement	The measurement that will be used for calculating the position of the device. *Metres* or *Centimetres*
x	The x coordinate of the device's position
y	The y coordinate of the device's position
z	The z coordinate of the device's position

The orientation property has the sub-properties illustrated in Table 5.

Table 5 - Sub-properties of orientation

Orientation property	Description
x	The x axis rotation of the device
y	The y axis rotation of the device
z	The z axis rotation of the device

It is worth mentioning that all the properties of Tables 3, 4 and 5 are returned but with different names tin order to be compatible with the database naming conventions.

Example 2: Get all devices

The API endpoint for getting all devices is presented in Figure 7. A successful request returns HTTP 200 Status and the properties of all devices are displayed on the screen.

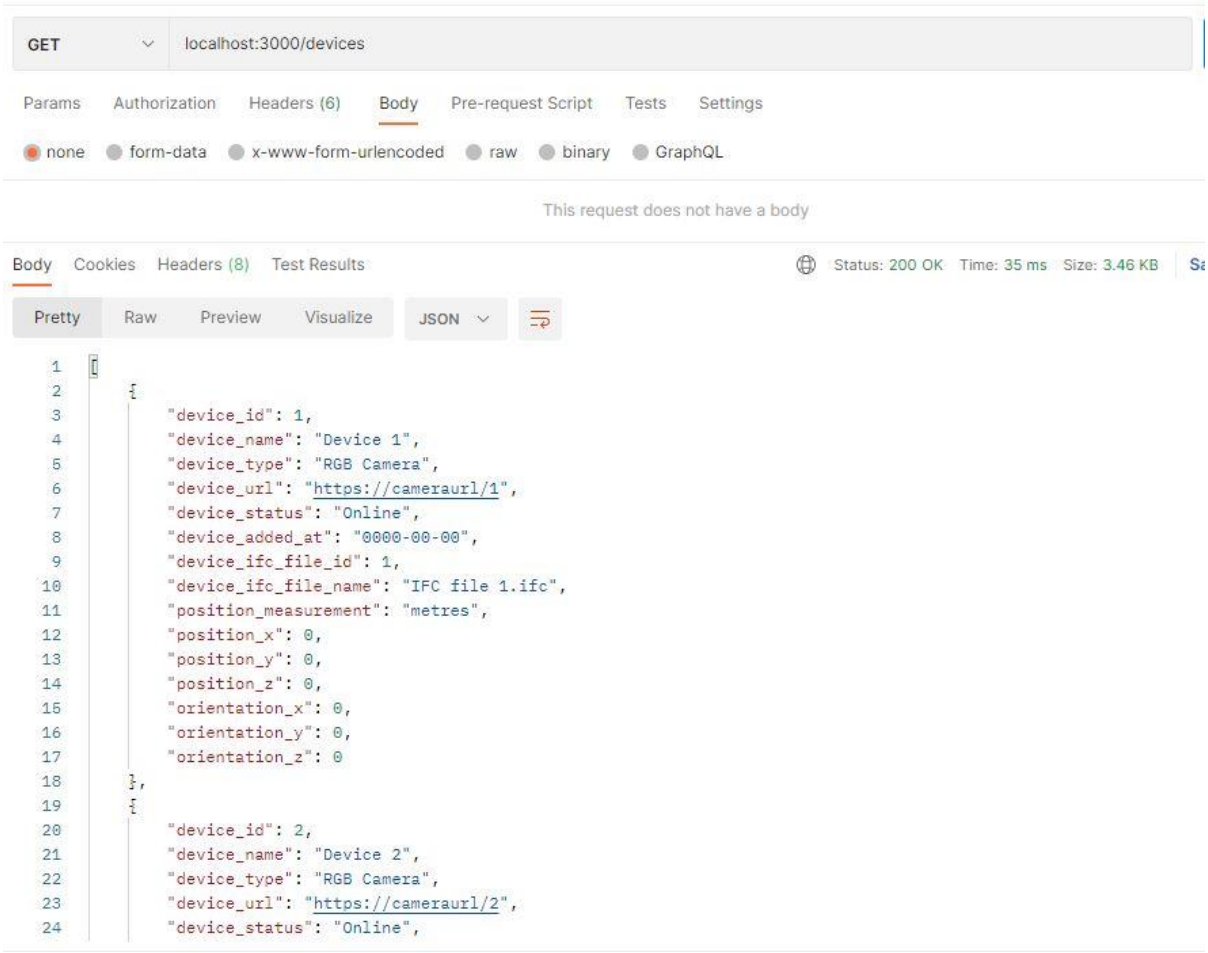


Figure 7 - API endpoint for getting all the devices- request and response

2.3.3.2 Job Requests

The Job requests that can be made are the following and summarized in Table 6.

- Add a job
- Get data of all jobs
- Get data of all jobs from a specific device
- Get data of a specific job using its ID

- Update a specific job
- Upload the raw photo of a job
- Get the raw photo of a job
- Upload the processed photo of a job
- Get the processed photo of a job
- Delete a specific job and its photos

Table 6 - Job requests

Request	Description	Method	Endpoints	Response
Add a job	The API endpoint to add a job	POST	**POST** localhost:3000/jobs	JSON
Get data of all jobs	API endpoint to get all the data from the jobs.	GET	**GET** localhost:3000/jobs	JSON
Get data of all jobs from a specific device	API endpoint to get the data of all jobs from a specific device.	GET	**GET** localhost:3000/jobs/device/{{:deviceId}}	JSON
Get data of a specific job using its ID	API endpoint to get the data of a specific job.	GET	**GET** localhost:3000/jobs/{{:id}}	JSON
Update a specific job	API endpoint to update the data of a specific job.	PUT	**PUT** localhost:3000/jobs/{{:id}}	JSON
Upload the raw photo of a job	API endpoint to upload the raw photo of a job	PUT	**PUT** localhost:3000/jobs/{{:id}}/raw	JSON
Get the raw photo of a job	API endpoint to get the raw photo of a specific job	GET	**GET** localhost:3000/jobs/{{:id}}/raw	File of the raw photo
Upload the processed photo of a job	API endpoint to upload the processed photo of a job	PUT	**PUT** localhost:3000/jobs/{{:id}}/processed	JSON
Get the processed photo of a job	API endpoint to get the processed photo of a specific job	GET	**GET** localhost:3000/jobs/{{:id}}/processed	File of the processed photo
Delete a specific job and its photos	API endpoint to delete a specific job	DELETE	**DELETE** localhost:3000/jobs/{{:id}}	JSON

Example 1: Add a job

The API endpoint for adding a job is presented in Figure 8. A successful registration returns HTTP 200 Status and the id created by the database for the newly registered job is displayed on the screen.

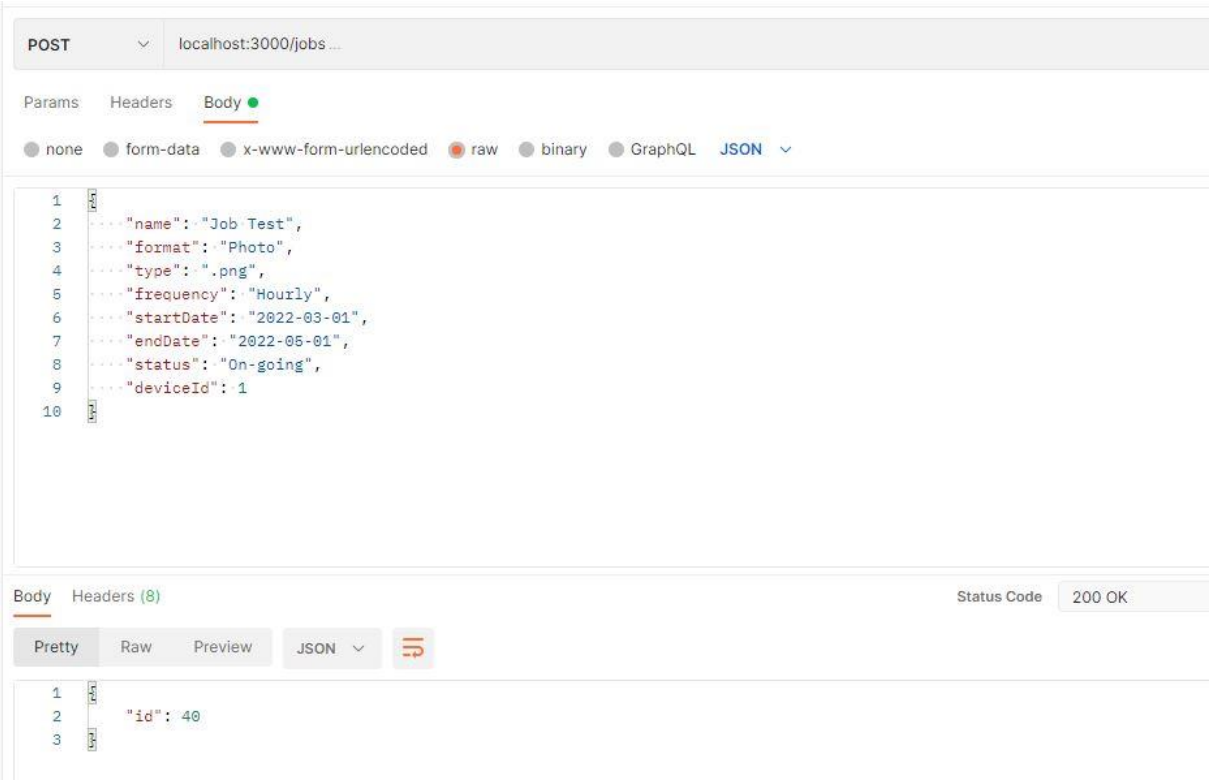


Figure 8 - API endpoint for adding a job – request and response

The properties that should be passed in the JSON body request are presented in Table 7.

Table 7 - Job properties

Property	Description
name	The name of the job
format	The format of the job (Photo, Point Cloud Data)
type	The type in which the file will be stored (jpeg, png, etc)
frequency	The frequency in which the device will perform this job
startDate	The starting date of the job
endDate	The ending date of the job
status	Status of the job. *On-going* or *Finished*
deviceId	The unique identifier of the device to which the job belongs

All the properties of Table 7 are returned with different names to be compatible with the database naming conventions.

Example 2: Upload the raw photo of a job

The API endpoint for uploading the raw photo of a job is illustrated in Figure 9. The body of the request is not a JSON but multipart/form-data. A key-value pair is generated with the “key” being set to “image” and its “value” being the file of the photo.

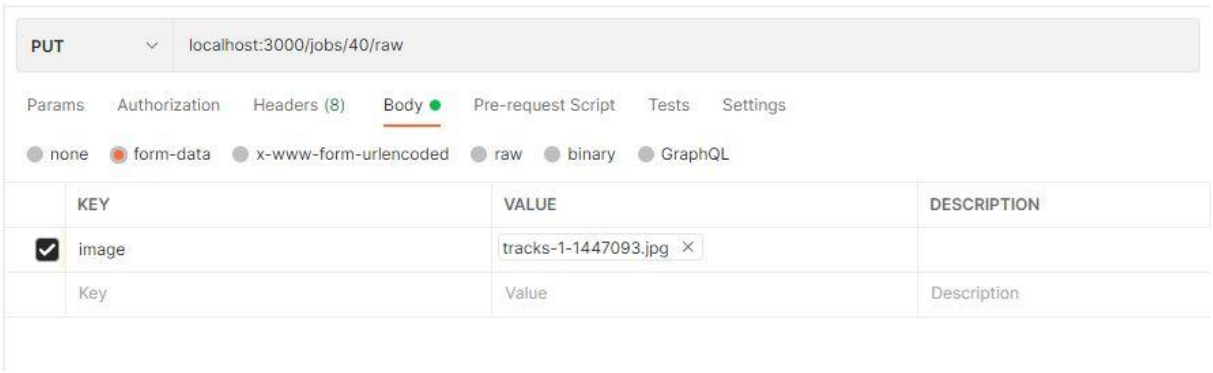


Figure 9 - API endpoint for uploading the raw photo of a job

Example 3: Upload the processed photo of a job

The API endpoint for uploading the processed photo of a job is depicted in Figure 10. As before, the body of the request is not a JSON but multipart/form-data. A key=value pair is generated with the "key" being set to "image" and its "value" being the file of the photo.

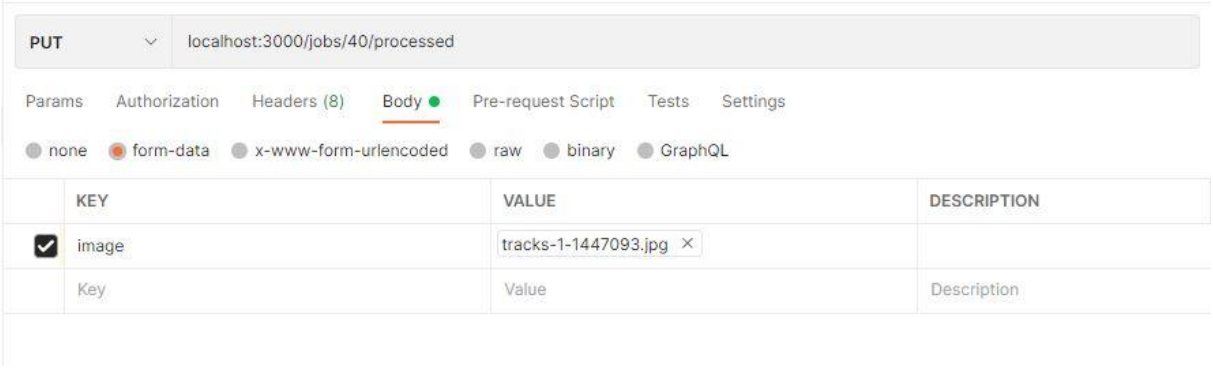


Figure 10 - API endpoint for uploading the processed photo of a job

Example 4: Get all jobs

The API endpoint for getting all the jobs is depicted in Figure 11. A successful request returns HTTP 200 Status and the properties of all jobs are displayed on the screen.

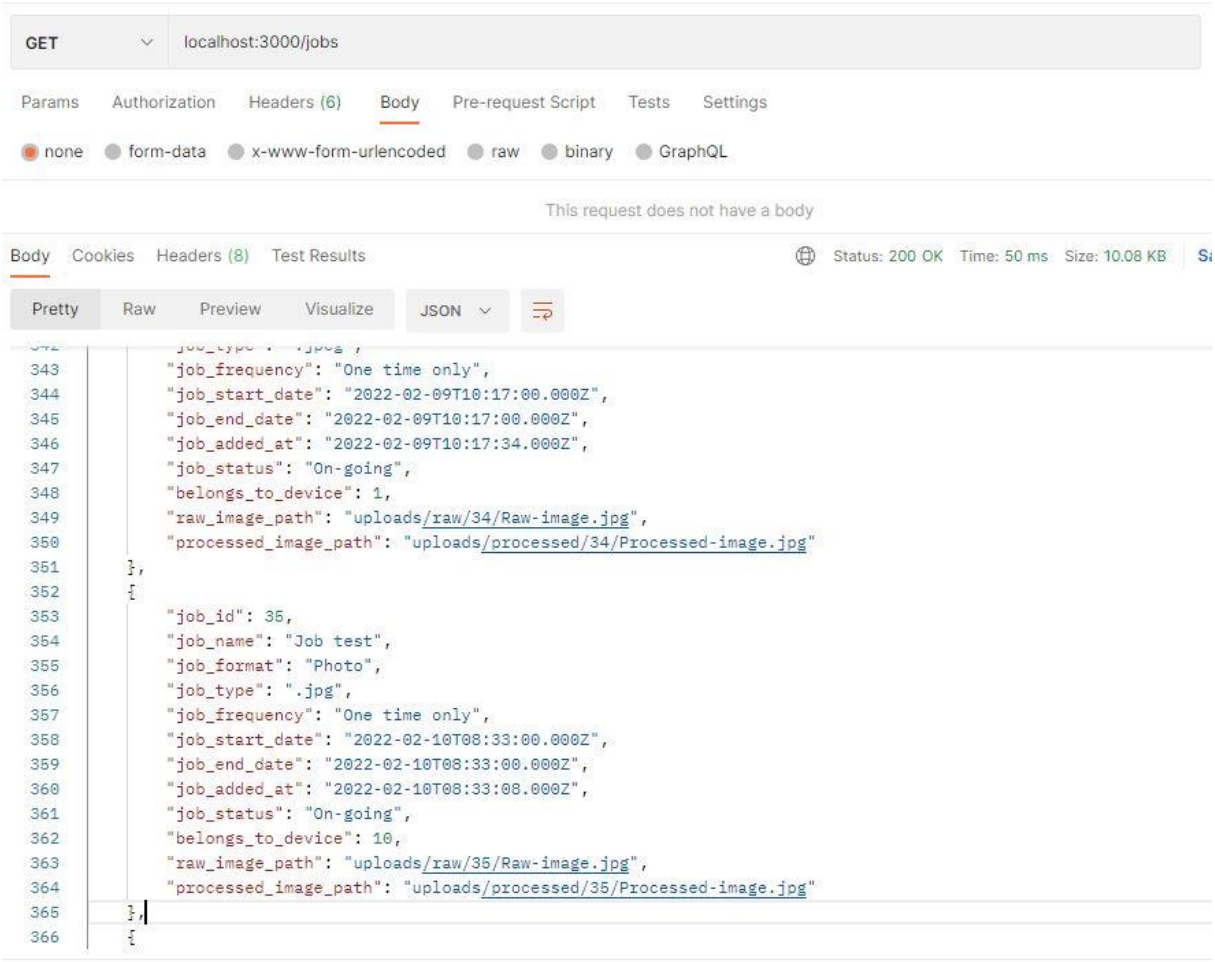


Figure 11 - API endpoint for getting all the jobs- request and response

For now, the Visual Data Pre-processing Module can run locally. However, the module will expose an execution interaction with the Digital Twin Platform in its second release, to be consistent with the rest of the COGITO solution.

2.4 Usage Walkthrough

The user can access the Visual Data Pre-processing Module's home page using any modern browser. Upon accessing the home page, the user should provide the credentials for authentication and login to the application, as illustrated in Figure 12.

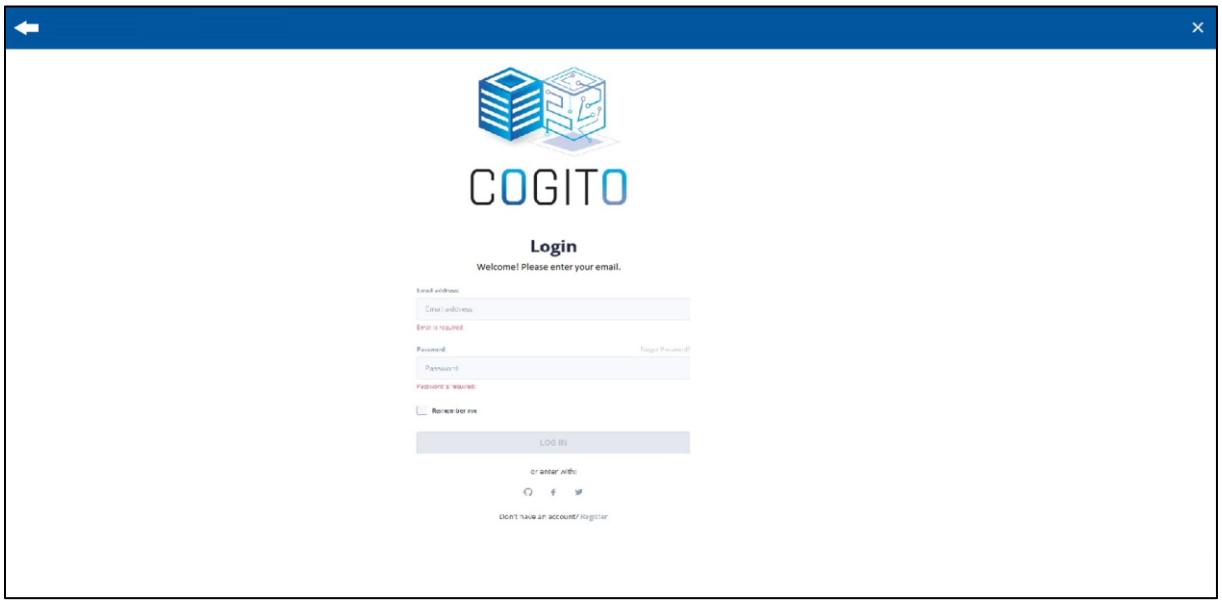


Figure 12 - Visual Data Pre-processing Module user's login page

In addition, the desired project can be selected from a list of projects provided by the DT Platform, based on the user's credentials (Figure 13).

Figure 13 - Project Selection

In Figure 14 the accounts' options are illustrated. The user can make changes regarding on their account profile and configure the general settings according to preference.

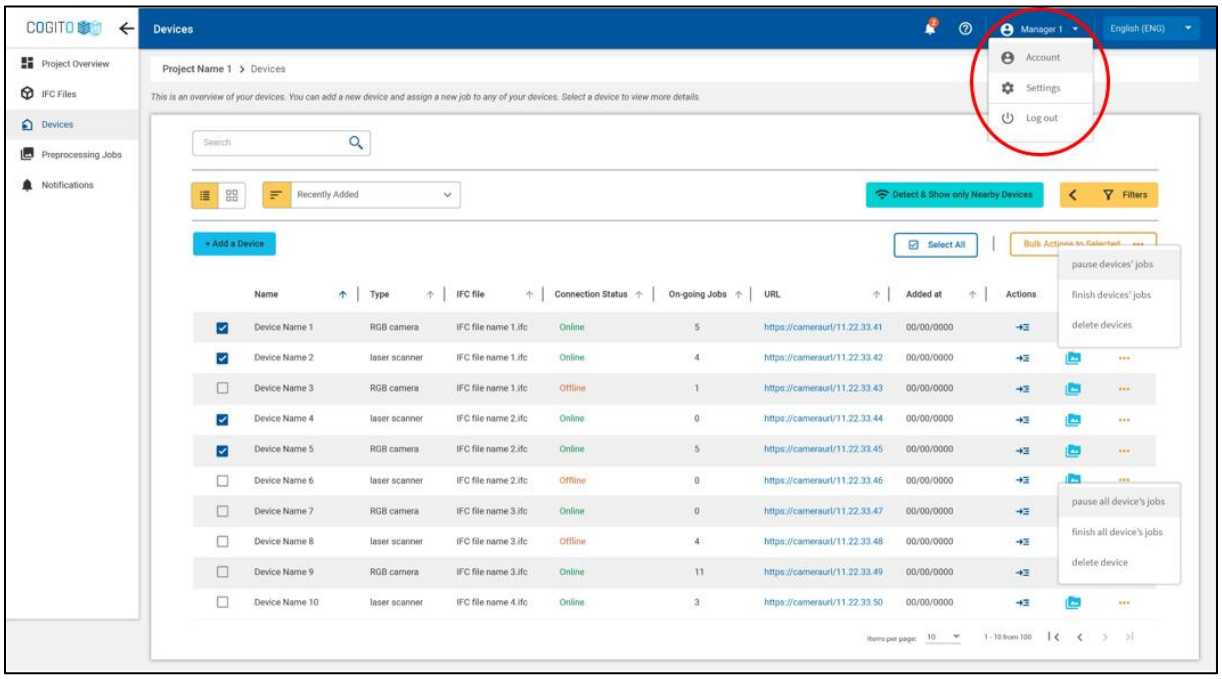


Figure 14 - Account options

By selecting the bulk actions button, the user can handle a group of devices (pause, finish and delete all devices) and the relevant jobs, as illustrated in Figure 15.

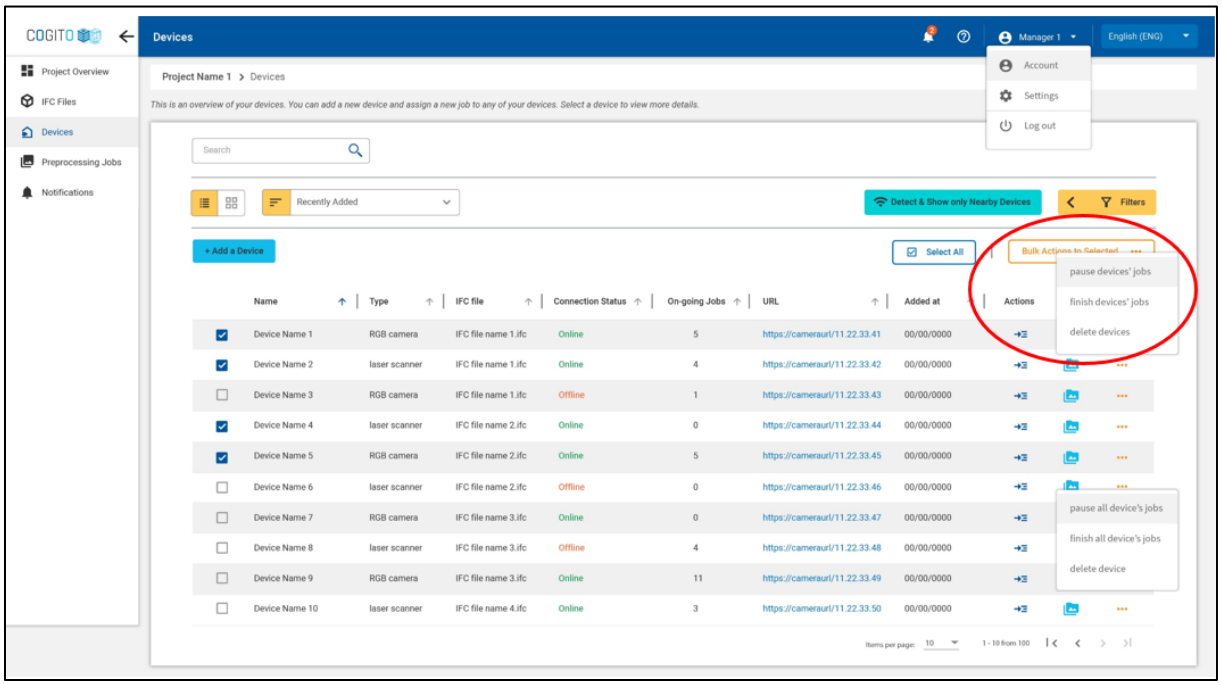


Figure 15 - Bulk actions button for grouping the devices

By clicking the Filters button, the user can apply filters to facilitate the search of specific (available) devices (e.g. Type of device, Project file, number of jobs, etc.) (see Figure 16).

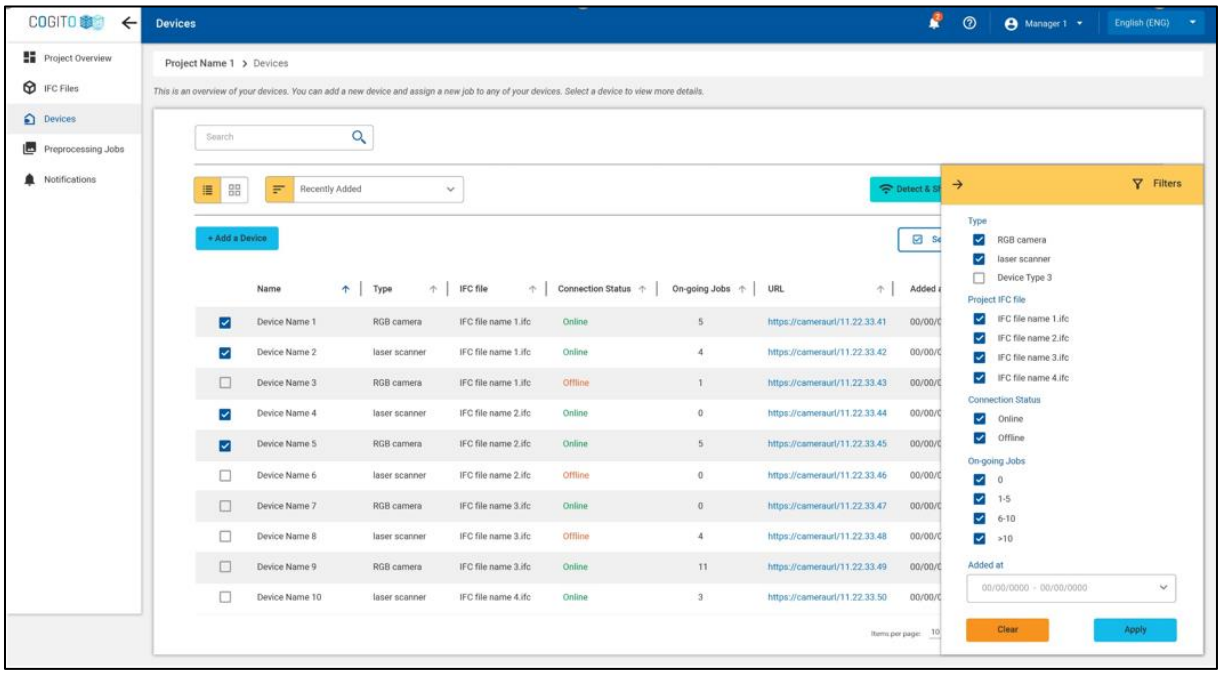


Figure 16 - Applying search filters

If the desired capturing device is not included in the above list, the user can add a new device and its properties (Name, Type and the relevant URL) as illustrated in Figure 17.

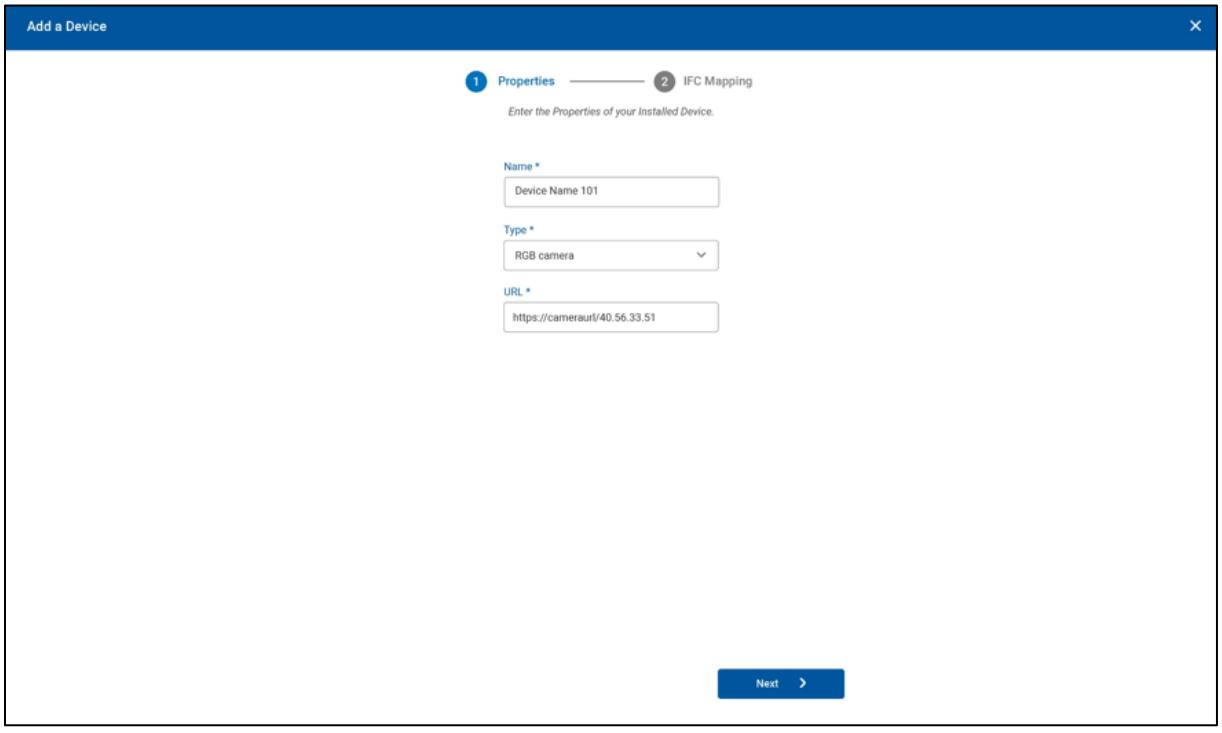


Figure 17 - Add a new capturing device

The next step is the IFC Mapping. More specifically, the user has to load the corresponding IFC file and declare the basic localization information. It is necessary to mark the new device's location (X, Y, Z) and define its view. At this point, the Device IFC View should match the actual device's view, as seen through the device's URL. The above process is depicted in Figure 18.

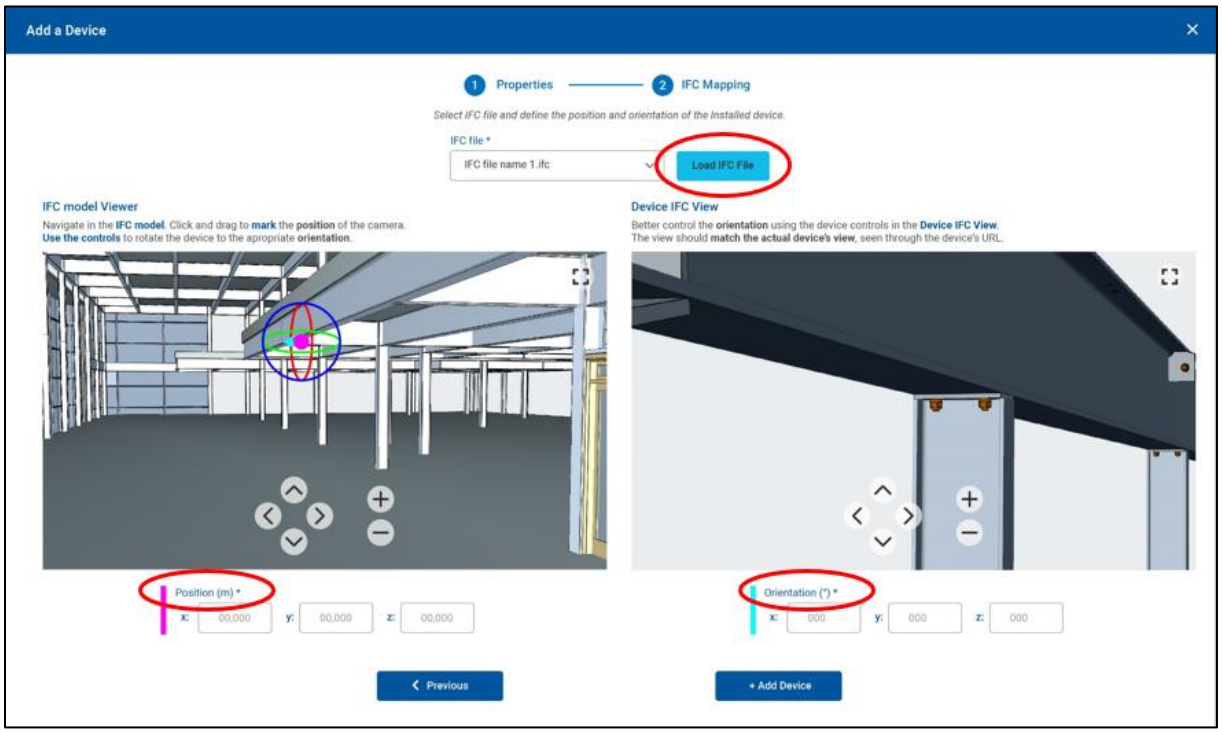


Figure 18 - IFC Mapping of the new device

By clicking the Jobs tab, the available jobs for a specific device are displayed. Every job entry has its own buttons (view details, download results and more actions). There are also general UI buttons for filters, for sorting the jobs, and for adding a new one, as illustrated in Figure 19.

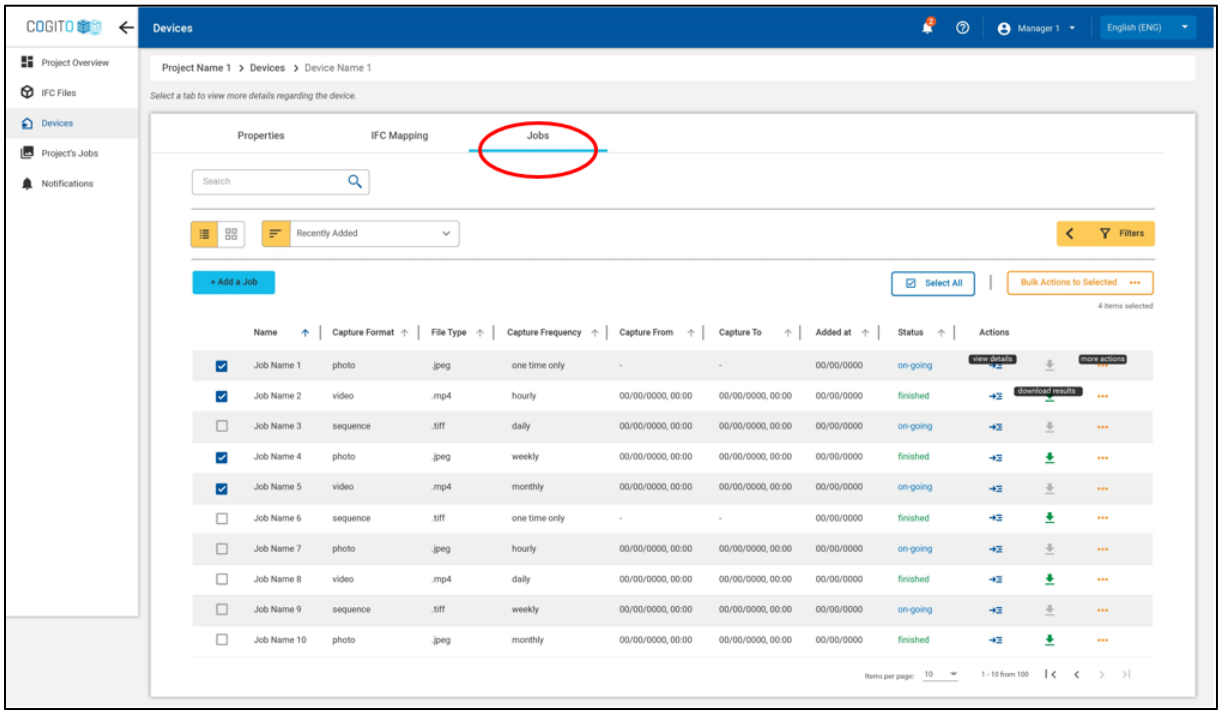


Figure 19 - Jobs tab for a specific device

By clicking the “Add a Job” Button, the user can add for this specific device a new job and its properties (name, capture format, capture frequency, etc.) (see Figure 20).

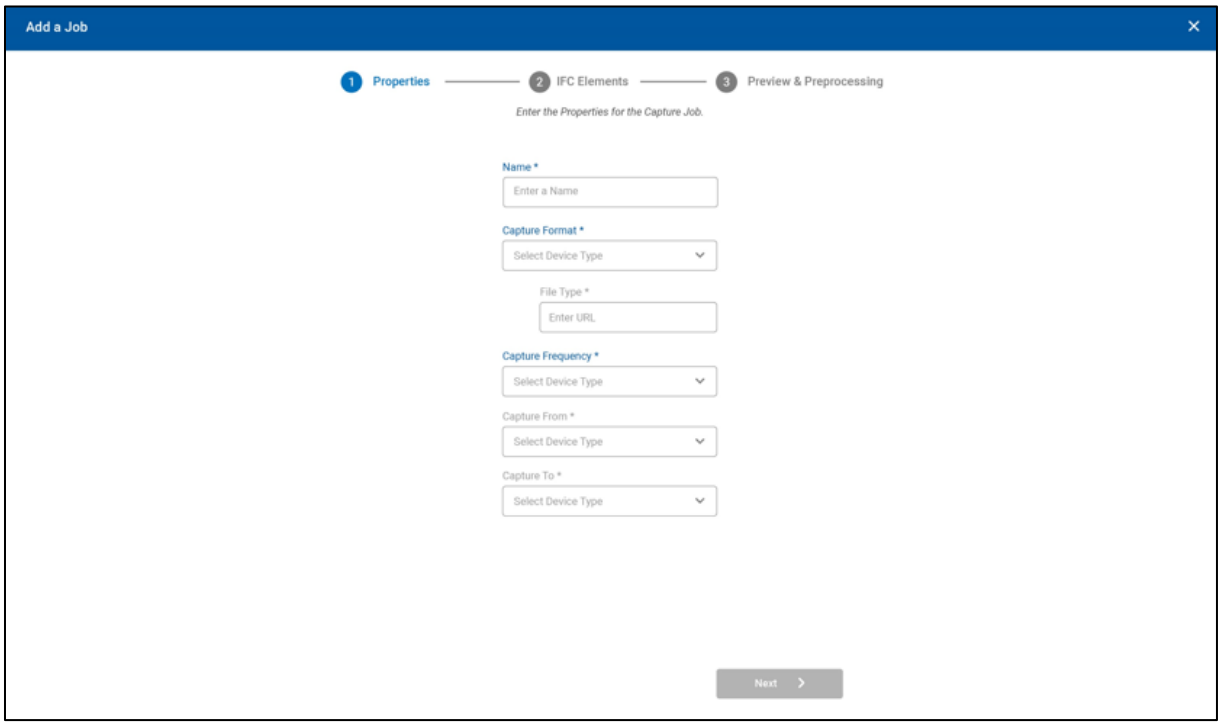


Figure 20 - Add a new capturing job

In addition, the user can assign the relevant IFC components to this specific job, based on the list of components of a specific working order, provided by the WODM tool. This way, the visual data is registered with the actual structural data of the BIM model (see Figure 21).

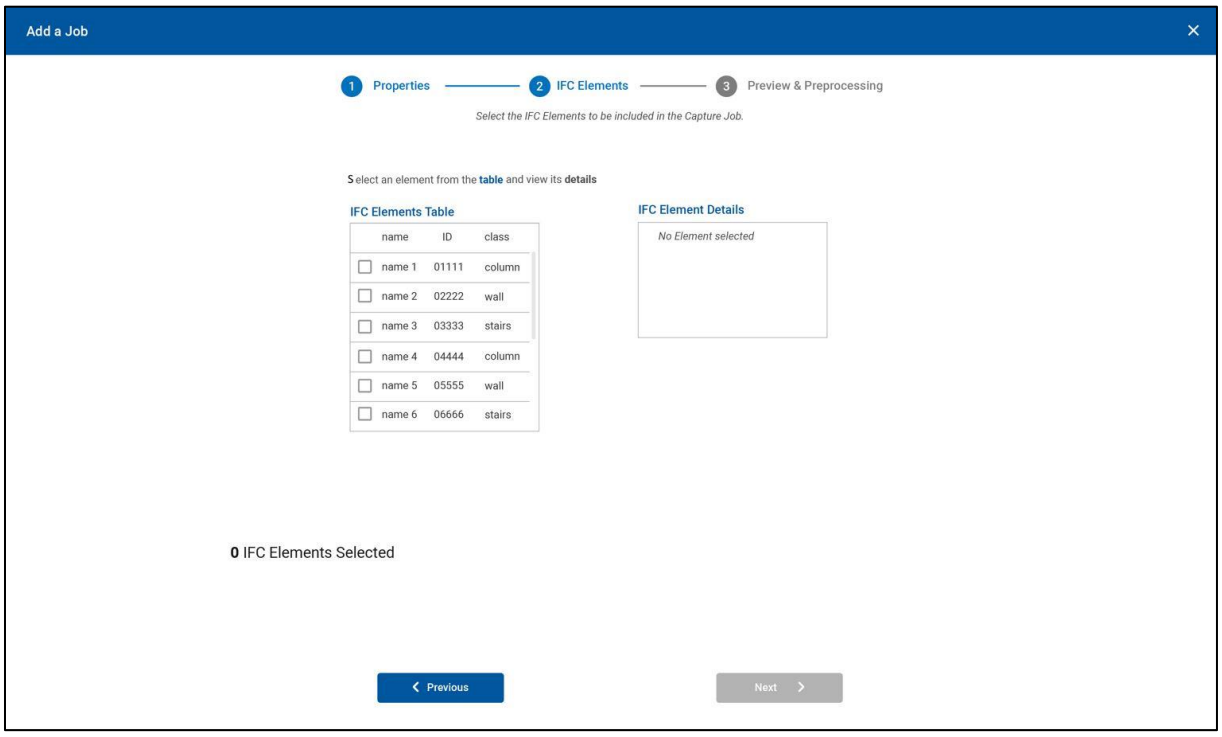


Figure 21 - Components attribution to a new capturing job

Afterwards, the user selects the pre-processing filters. The processed image is displayed on the screen and the user decides whether the processed data is accepted or not to be related with the new job, as illustrated in Figure 22.

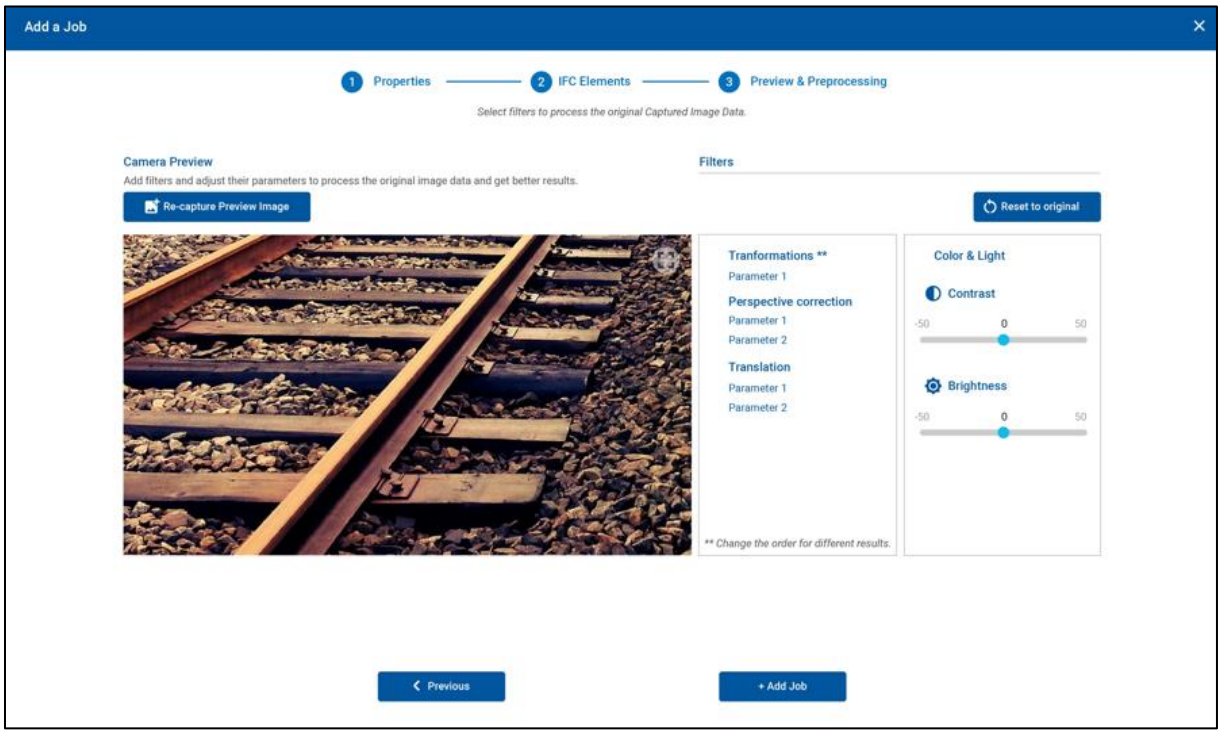


Figure 22 - Image processing and filter implementation

Once the job is added, it can be submitted to the DT Platform for further exploitation (Figure 23).

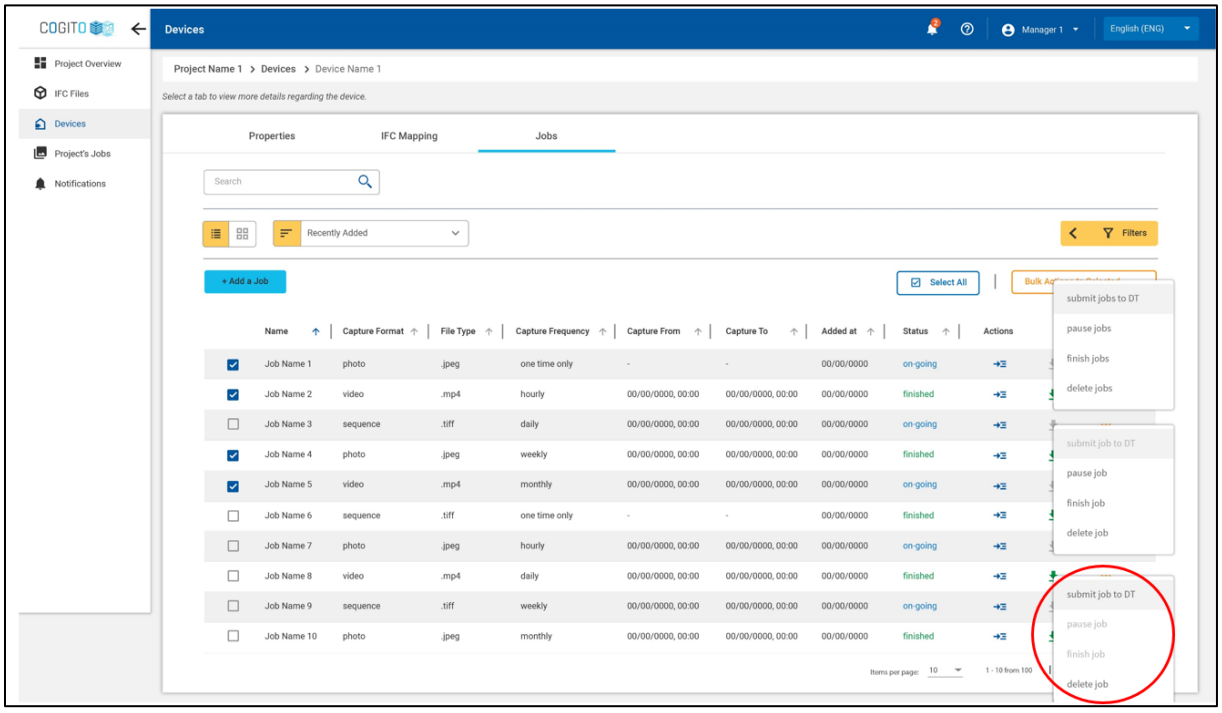


Figure 23 - Submitting a job to the DT Platform

2.5 Licensing

The Visual Data Pre-processing Module is a closed source component.

2.6 Installation Instructions

The Visual Data Pre-processing Module is available as a web-based application, thus no installation is required.

2.7 Development and Integration Status

The Visual Data Pre-processing module is currently under development. The alpha version of the tool has been developed and focuses on the 2D image handling. The main functionalities, such as job addition, data storage and basic image processing filters (contrast, brightness, Gaussian blurring) have already been implemented and are ready to be used. Regarding the second release of the component in M18, the intention is to focus on the pre-processing of the point cloud and the geometric data (Point Cloud Uploader sub-module). In addition, it is planned to expand the functionalities of the image processing (adding extra visual filters or refine the existing ones) and implement the online IFC 3D Viewer. Finally, the Visual Data Pre-processing module interacts with the DT Platform and all the capturing devices; hence, in the final version of the module, the communication with all the relevant external tools should be established.

2.8 Requirements Coverage

Table 8 presents the stakeholders requirements that have been documented in D2.1 and are relevant to this component [11]. COGI-CS-1 and COGI-CS-4 are covered simply by the programming language that has been selected for the development of the Visual Data Pre-processing tool. With regards to COGI-WF-2, the tool allows the Project Manager (PM) and Site Manager (SM) to share information (design data, photos, point cloud etc.). In addition, concerning COGI-WF-5, the SM is able to share information with Subcontractors, Foreman, and Workers (design data, photos etc.)

Table 8 - Visual Data Pre-processing tool: Stakeholders' requirements coverage from D2.1

ID	Description	Type	Priority	Status
COGI-CS-1	Runs on desktop or laptop PC	• Operational	Must	Achieved
COGI-CS-4	Runs on Windows	• Operational	Must	Achieved
COGI-WF-2	Allows the PM and Site Manager to share information (design data, photos etc.)	• Functional • Design constraint	Must	Partially achieved
COGI-WF-5	Allows the SM to share information with Subcontractors, Foreman, and Workers (design data, photos etc.)	• Functional • Design constraint	Should	Partially achieved

The functional and non-functional requirements are presented in Table 9. Concerning the functional requirements of the tool, Req.1-1 is covered while a local database is developed for storing the raw visual data temporarily. In addition, Req.1-2 is partially achieved, while several filters (such as contrast, brightness, Gaussian blurring, cropping and scaling) have been already implemented at the raw visual data. Req.1-3 concerning the user's ability to define the location and the orientation of the capturing device will be covered in the second release of this component. As for the Req.1-4, it is partially achieved and it will be well covered in the final version of the Visual Data Pre-processing module. Concerning the non-functional requirements, the status of Req-2.1 is considered to be partially achieved given that the web application needs to be further extended. Since Req-2.2 is well covered, Req-2.3, Req-2.4, Req-2.5, Req-2.6 and Req-2.7 can also be considered as well covered by this version of the component.

Table 9 - Visual Data Pre-processing tool: Functional and Non-Functional Requirements coverage from D2.4

ID	Description	Type	Status
Req-1.1	Stores the raw data in a local database	Functional	Achieved
Req-1.2	Filters the raw data input (Smoothing, de-noising, enhancing)	Functional	Partially achieved

Req-1.3	Registers the visual inputs to location, direction, and time series data	Functional	Not yet supported
Req-1.4	Sends Images and associated data to DT platform	Functional	Partially achieved
Req-2.1	Web based App	Non-Functional	Partially achieved
Req-2.2	Offers efficiently structured database to store and retrieve data	Non-Functional	Achieved
Req-2.3	Reliability	Non-Functional	Achieved
Req-2.4	Scalability	Non-Functional	Achieved
Req-2.5	Performance	Non-Functional	Achieved
Req-2.6	Security	Non-Functional	Achieved
Req-2.7	Data Integrity	Non-Functional	Achieved

2.9 Assumptions and Restrictions

The first version of the Visual Data Pre-processing Module is accompanied by certain assumptions and restrictions, which are presented in the following:

- The current version of the module does not contain a 3D viewer for the IFC Visualization. It will be implemented in the final version of the module (M18).
- The current version of the module's REST API supports queries regarding jobs and devices. Additional REST API endpoints regarding the filters' implementation will be included in the final version of the Visual Data Pre-processing Module.
- The current version of the module is image processing-oriented. While experiments regarding handling point clouds have not been performed yet, the topic will be covered in the final version of the module.
- A first version of declaring the device position and orientation (X,Y,Z and field of view) has been initiated, but is not completed yet. This initial version will be enriched and evolved in the final version of the module, conforming to the tool's new requirements

3 Conclusions

This deliverable introduced the main functional components of Visual Data Pre-processing Module and their use. Several information regarding the development tools, the data exchange and an alpha version of the API were also presented.

As documented in this deliverable (D3.7), the Visual Data Pre-processing Module is a web-based application, which offers many functionalities to the user. The Visual Data Pre-processing Module is composed of three sub-modules: the Main Pre-processing sub-module, the Pre-processing sub-module for 2D Visual Data and the Pre-processing sub-module for Geometric Data. In addition, a local database is developed to store all the relevant data. Within the Visual Data Pre-processing Module, the user is able to handle as-built data (point clouds and 2D images) obtained through laser scanning surveying or static cameras during construction. The modified processed/ enhanced data is delivered to the DT Platform in the appropriate form for further exploitation by other COGITO components (Geometric and Visual Quality Control tools). Furthermore, the module allows the manual connection of the visual data with the BIM components; thus, the visual data is linked with the IFC elements and related to specific jobs.

The work presented here introduces a first version of the Visual Data Pre-processing Module. Thus far, the module has been tested only with basic artificial test data. In the future and as COGITO tools evolve, more functionalities will be added to the module, to support all the client demands as well as the connectivity and communication with the other COGITO tools. The final version of the Visual Data Pre-processing Module will be provided at M18 and will be tested on the pre-validation (T8.2) and validation (T8.4) sites with real case data during M21-30 and M28-34 respectively.

References

[1] [Online]. Available: https://bestinterviewquestion.medium.com/top-5-database-for-web-applications-d71a4229fa37.

[2] Riaz, Z., Parn, E., A., Edwards, D. J., Arslan, M., Shen, C., Pena-Mora, F., "BIM and sensor-based data management system for construction safety monitoring," *Journal of Engineering, Design and Technology,* 2017.

[3] [Online]. Available: https://gisuser.com/2020/08/how-should-you-choose-the-best-database-for-web-applications/.

[4] Veen, J. S. van der, Waaij, B. van der, Meijer, R. J. , "Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual," in *2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 431-438.

[5] [Online]. Available: https://nodejs.org/en/docs/.

[6] [Online]. Available: https://angular.io/docs.

[7] [Online]. Available: https://www.mysql.com/.

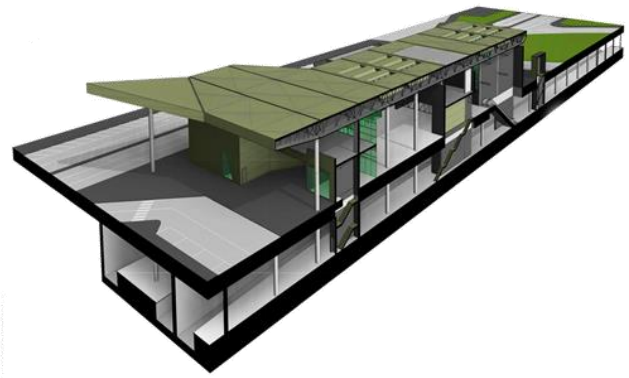
[8] [Online]. Available: https://www.npmjs.com/.

[9] [Online]. Available: https://akveo.github.io/ngx-admin/.

[10] [Online]. Available: https://www.ibm.com/cloud/learn/api.

[11] D2.1-COGITO, "Deliverable 2.1: Stakeholder requirements for the COGITO system," 2021.

[12] D2.4-COGITO, "Deliverable 2.4: COGITO System Architecture v1," 2021.



COGITO

CONSTRUCTION PHASE DIGITAL TWIN MODEL

cogito-project.eu

HYPERTECH energy labs

UCL

AARHUS UNIVERSITET

THE UNIVERSITY of EDINBURGH

BoC www.boc-group.com

iti

POLITÉCNICA

QUE TECHNOLOGIES

Novitech New Information Technologies

ASM

ferrovial construction

ΟΛΥΜΠΙΑ ΟΔΟΣ

RHOMBERG SERSA RAIL GROUP

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310