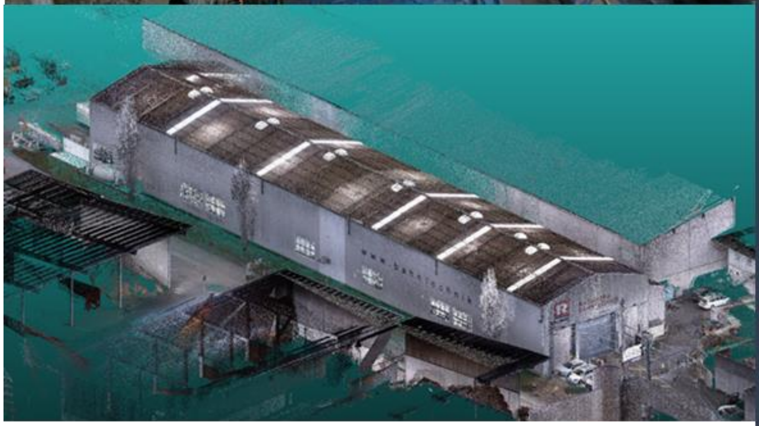


COGITO

CONSTRUCTION PHASE
DIGITAL TWIN MODEL

cogito-project.eu

D3.8 – Visual Data Pre-processing Module v2

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

D3.8 – Visual Data Pre-processing Module v2

Dissemination Level:	Public
Deliverable Type:	Demonstrator
Lead Partner:	CERTH
Contributing Partners:	UCL, UEDIN, NT
Due date:	30-04-2022
Actual submission date:	29-04-2022

Authors

Name	Beneficiary	Email
Thanos Tsakiris	CERTH	atsakir@iti.gr
Apostolia Gounaridou	CERTH	agounaridou@iti.gr
Vasilis Dimitriadis	CERTH	dimvasdim@iti.gr
Michalis Chatzakis	CERTH	mchatzak@iti.gr
Anastasios Sinanis	CERTH	sinanis@iti.gr
Martin Bueno Esposito	UEDIN	martin.bueno@ed.ac.uk
Frederick Bosche	UEDIN	f.bosche@ed.ac.uk

Reviewers

Name	Beneficiary	Email
Socorro Bernardos	UPM	sbernardos@fi.upm.es
Elias Bruno Meusburger	RSRG	Elias.Meusburger@rsrg.com

Version History

Version	Editors	Date	Comment
0.1	CERTH	01.04.2022	Initial contents taken from D3.7, updates to Executive Summary and Chapter 1.
0.2	CERTH	19.04.2022	Updates to Chapter 2.
0.3	CERTH, UEDIN	20.04.2022	Updates to Section 2.1.1.4, 2.1.2, 2.1.4, 2.2, 2.3.3, 2.4, Section 3.
0.4	CERTH	20.04.2022	Submission for peer review.
0.6	UPM, RSRG	26.04.2022	Peer review
0.8	CERTH	28.09.2022	Peer review comments addressed
1.0	CERTH, Hypertech	29.04.2022	Submission to the EC

Disclaimer

Executive Summary

The COGITO Deliverable D3.8 “Visual Data Pre-processing Module v2” documents is the second version of the COGITO Visual Data Pre-processing Module and focuses on the updated design and 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 and 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 second release, more functionalities are implemented with regards to 2D image pre-processing refinement as well as the Pre-processing sub-module for Geometric Data (point clouds enhancing and uploading).

Table of contents

List of Figures

List of Tables

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 M17 to M18 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 Geometric QC and the Visual QC tool) for performing automatic QC compliance.

More specifically, this deliverable reports on the development of the second 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 D3.8 "Visual Data Pre-processing Module v2" are related to the following COGITO tasks and deliverables:

- The second version of the COGITO architecture in the corresponding deliverable "D2.5 COGITO System Architecture v2 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 D3.3 "COGITO Data Model, Ontology Definition and Interoperability Design v2.
- 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 the integration procedure.
- 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 COGITO System Integration.
- The document concludes with Section 3, where the progress, the next steps and the contribution to the overall COGITO objectives are being reported.

1.4 Updates to the first version of the Visual Data Pre-processing Module

Since the submission of D3.7 in M16, the COGITO Visual Data Pre-processing Module has evolved as in the following:

- **Refinements to the sub-module for 2D visual data.** The Image Processing Library was extended and finalized by optimizing extra image processing functionalities (processor types) such as cropping and rescaling.
- **Implementation of handling as-built geometric data.** The first version of the module could be considered as image processing oriented. In this second version, we have focused in handling point clouds, and their uploading to the DT Platform.
- **Refinements to the IFC Element Connector sub-module.** The procedure of the components attribution during the creation of a new pre-processing job was optimized and finalized in this second version of the module.
- **Implementation of IFC 3D Viewer sub-module.** In order to display the 3D BIM model and its components, a 3D Viewer was necessary to be developed within the module. The 3D Viewer was developed and integrated in the second release of the module.
- **Updates to the REST API endpoints.** Additional endpoints were implemented to cover new functionalities of the Visual Data Pre-processing Module such as the uploading of the point cloud, the image processing etc.
- **Updates to functional and non-functional requirements.** Based on the modifications derived from D2.5 “COGITO system architecture v2”, the functional and non-functional requirements table was updated in this second version of the Visual Data Pre-processing Module.
- **Preparative implementation of DT Platform integration.** During these months, some initial steps were carried out to investigate the execution of the efficient communication of the Visual Data Pre-processing Module with the DT Platform as well as with the other necessary COGITO components. Different methods and ways were examined in order to clarify the data exchange within the COGITO components, its structure, format etc.

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, since 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, as 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.

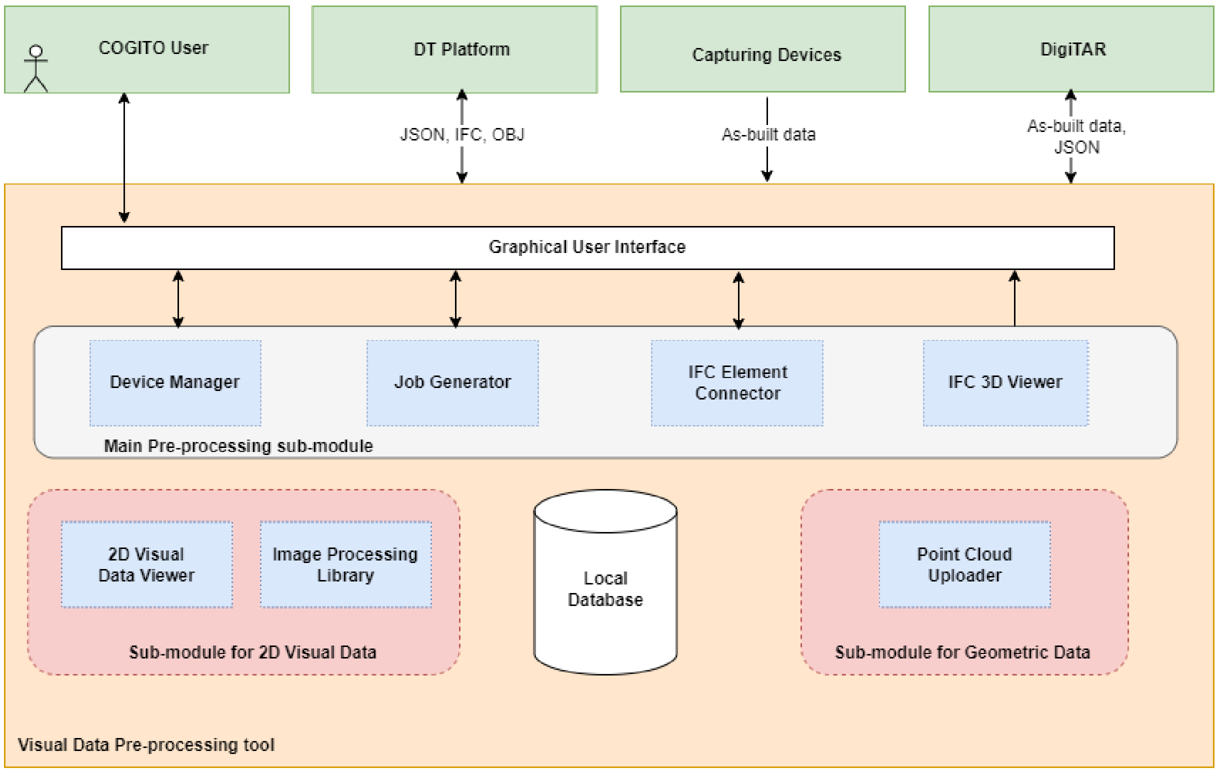


Figure 1 - Overall architecture of the Visual Data Pre-processing Module

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 capturing device in order to associate it with a new pre-processing job. 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.

Once a new device is added, it must be located within the 3D space. The user can visualize the related IFC 3D BIM model of the project (loaded from the DT Platform), 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.

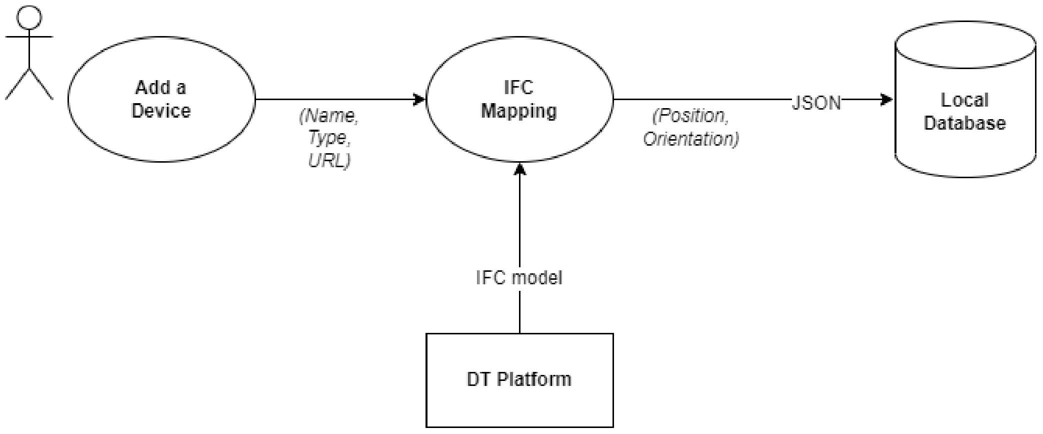


Figure 2 – Adding a device and IFC mapping

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 Visual QC survey work orders provided by the WODM tool, the relevant stakeholder can create a new pre-processing job concerning a specific Visual QC task. 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 sub-group of IFC components involved in a specific Visual QC task, the components that will be related with this specific job and then the as-build data depicting these IFC elements. Hence, the selected components are linked semi-automatically to the new capture job (components attribution) and the produced information is stored temporarily in the local database in a JSON format. Refinements and adjustments concerning the semi-automated process will be carried out during the System Integration (T8.1). Figure 3 illustrates both the two aforementioned steps (adding a new capture job and link the involved IFC elements).

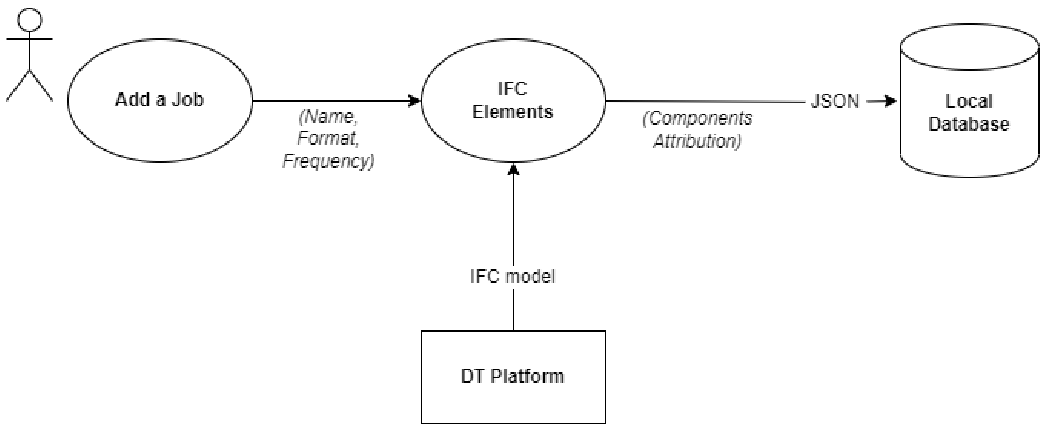


Figure 3 - Adding new job's properties and components' attribution

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. This sub-module is needed during two phases within the Visual Data Pre-processing Module: for adding new capturing devices as well as for the components' attribution to a specific job. Firstly, it is used while declaring the position and the orientation of the capturing device, its specific coordinates, as well as while defining the devices' field of view. In addition, the Viewer is also used when the IFC elements are assigned to a specific job, in order to visualize and relate semi-automatically the visual data with the IFC components. The Viewer was developed and integrated in the final release of the module. It is a WebGL application that is integrated in the Graphical User Interface of the Visual Data Pre-processing Module.

2.1.2 Local Database

Within the Visual Data Pre-processing module, a local database is developed to store all the necessary information. The data stored is related to devices, their jobs and the elements that were assigned to each job as well as the image processing that has been implemented for a job if any. 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. 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 is set and stored in the database.

The database created is a MySQL database [1] and its structure is illustrated in Figure 4. This database management system is considered 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 relational 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, 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 and the processed data (image or point cloud files) are stored in the job table. Regarding the image processing, multiple image processors can be applied to an image related to a job. In this second release of the module, the initial processor types already implemented in the first version of the module (such as brightness modification, contrast modification and Gaussian blurring) were enriched with extra image processing functionalities such as cropping and rescaling. All of these image processors are stored in a single table called image processing, as illustrated Figure 4. However, within the COGITO System Integration process next months (T8.1), updates and modifications may be carried out in the structure of the database to support all the related COGITO components' demands.

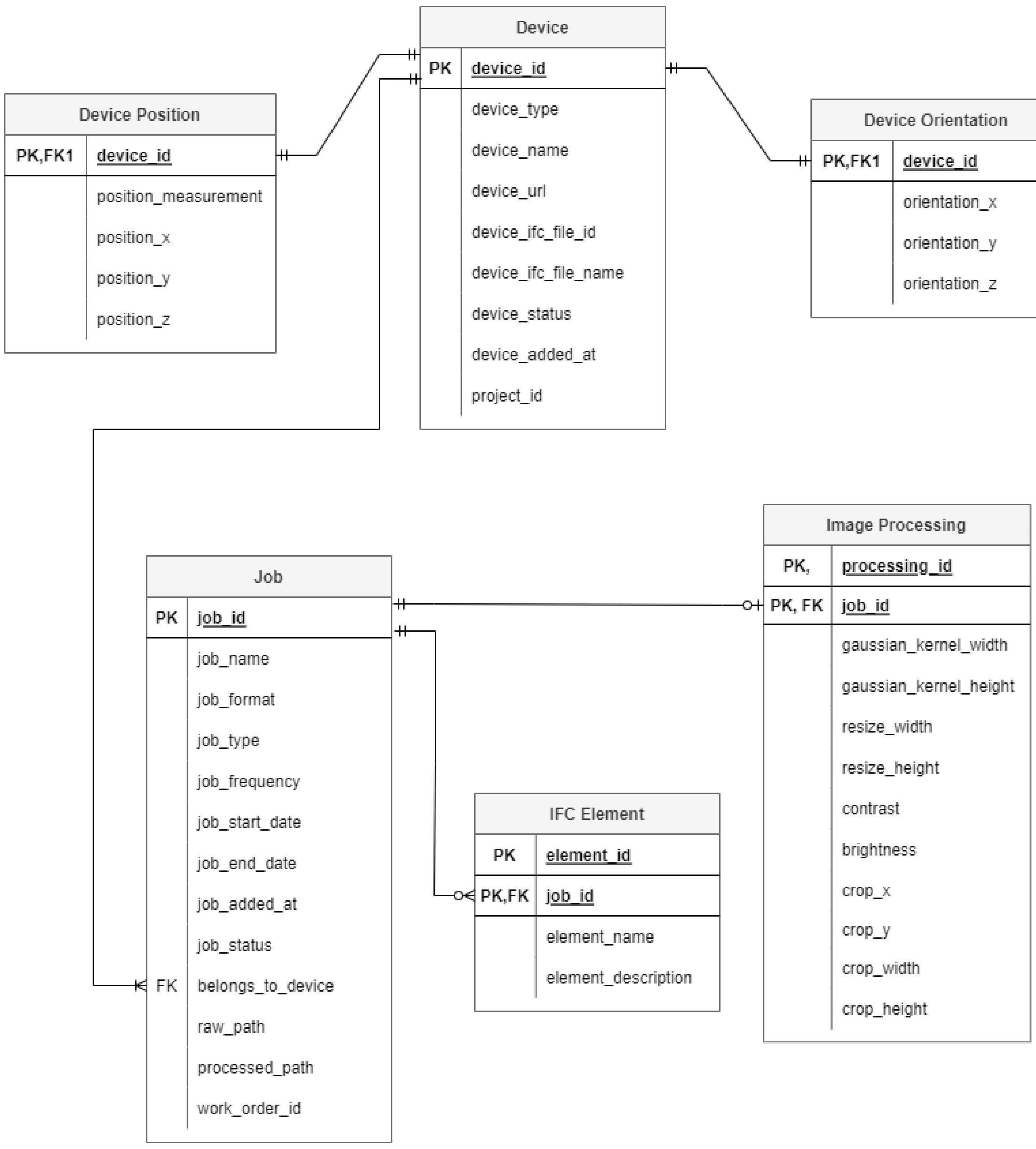


Figure 4 - Entity Relationship Diagram - MySQL database

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 Image Processing Library 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 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 Visual QC. 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.

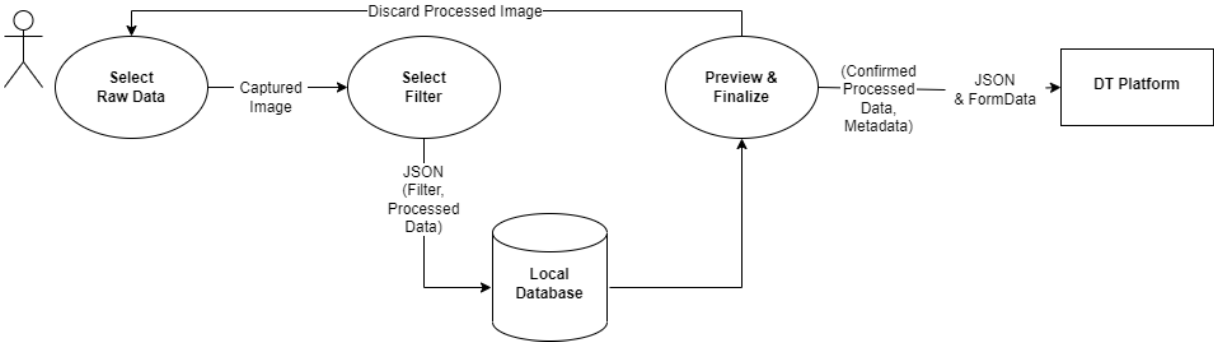


Figure 5 - Adding a new job: filter implementation and finalization

2.1.4 Sub-module for Geometric Data

The Pre-processing sub-module for Geometric Data is in charge of uploading 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 work order from the WODM tool, containing a list with the IFC elements that need to be geometrically checked by the Geometric QC tool. The relevant stakeholder (usually the surveyor) 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 stakeholder is also required to provide the as-built data already registered together (either providing the individual registered scans or a fully registered file) in the same coordinate frame as the original BIM model. It is also optional to remove outliers, clutter data, to help the Geometric QC tool to obtain a better performance. Once the stakeholder pre-processes the as-built data offline with they preferred techniques/tools, 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 work order. 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 DT Platform is ready to distribute all the necessary information to the Geometric QC component 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 management system for storing the necessary data. Furthermore, several libraries and packages are used for the development of the application (see Table 1).

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
Unity 3D Editor	2020.5	Software licensing

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

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

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].

Unity is a Game Engine created by Unity Technologies. The engine is widely used to build 3D and 2D applications, simulations, and games as well as Virtual Reality (VR) and Augmented Reality (AR) applications [10]. Programs in Unity can be written in the object-oriented programming language C#. C# enables the development of a variety of secure and robust applications that run in the .NET ecosystem. The scripts to implement the WebGL application for the IFC 3D viewer sub-module are developed in C#.

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 data:

- **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 data:

- **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 data:

- **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 data:

- **Raw and 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 second version is implemented in this final release of the Visual Data Pre-processing Module. The API is mainly used for storing, retrieving, updating and deleting data regarding devices and the related jobs [11]. Several requests for both the **Device** and **Job** cases have been built and tested in Postman API Platform[3]. The API can also be used to process an image related to a job using the relevant **Image Processing** requests. A few examples of all the available requests are presented in the next sub-sections. However the communication with the DT Platform will be established in within the T8.1 "End-to-end ICT System Integration, Testing and Refinement" (M18-M34) and therefore the API will be updated and refined next months.

2.3.3.1 Device Requests

The Device requests that can be made are summarized in Table 2.

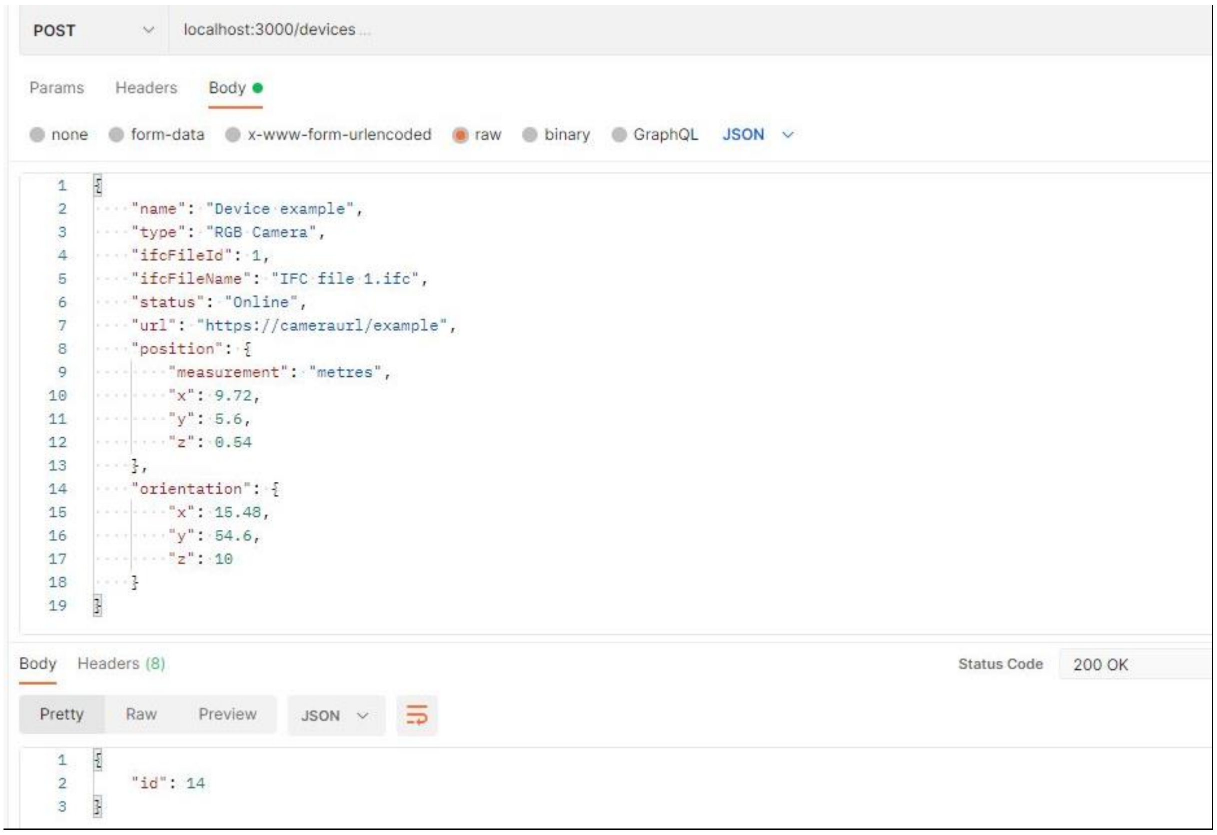
Table 2 - Device requests

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 a specific device using its ID	API endpoint to get the data from a specific device.	GET	**GET** localhost:3000/devices/{{:id}}	JSON
Get the number of on-going jobs of a device	API endpoint to count the number of jobs of a specific device that have an "On-going" status.	GET	**GET** localhost:3000/devices/{{:id}}/jobs	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/



POST localhost:3000/devices

```
{
    "name": "Device example",
    "type": "RGB Camera",
    "ifcFileId": 1,
    "ifcFileName": "IFC file 1.ifc",
    "status": "Online",
    "url": "https://cameraurl/example",
    "position": {
        "measurement": "metres",
        "x": 9.72,
        "y": 5.6,
        "z": 0.54
    },
    "orientation": {
        "x": 15.48,
        "y": 54.6,
        "z": 10
    }
}
```

Status Code 200 OK

```
{
    "id": 14
}
```

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.

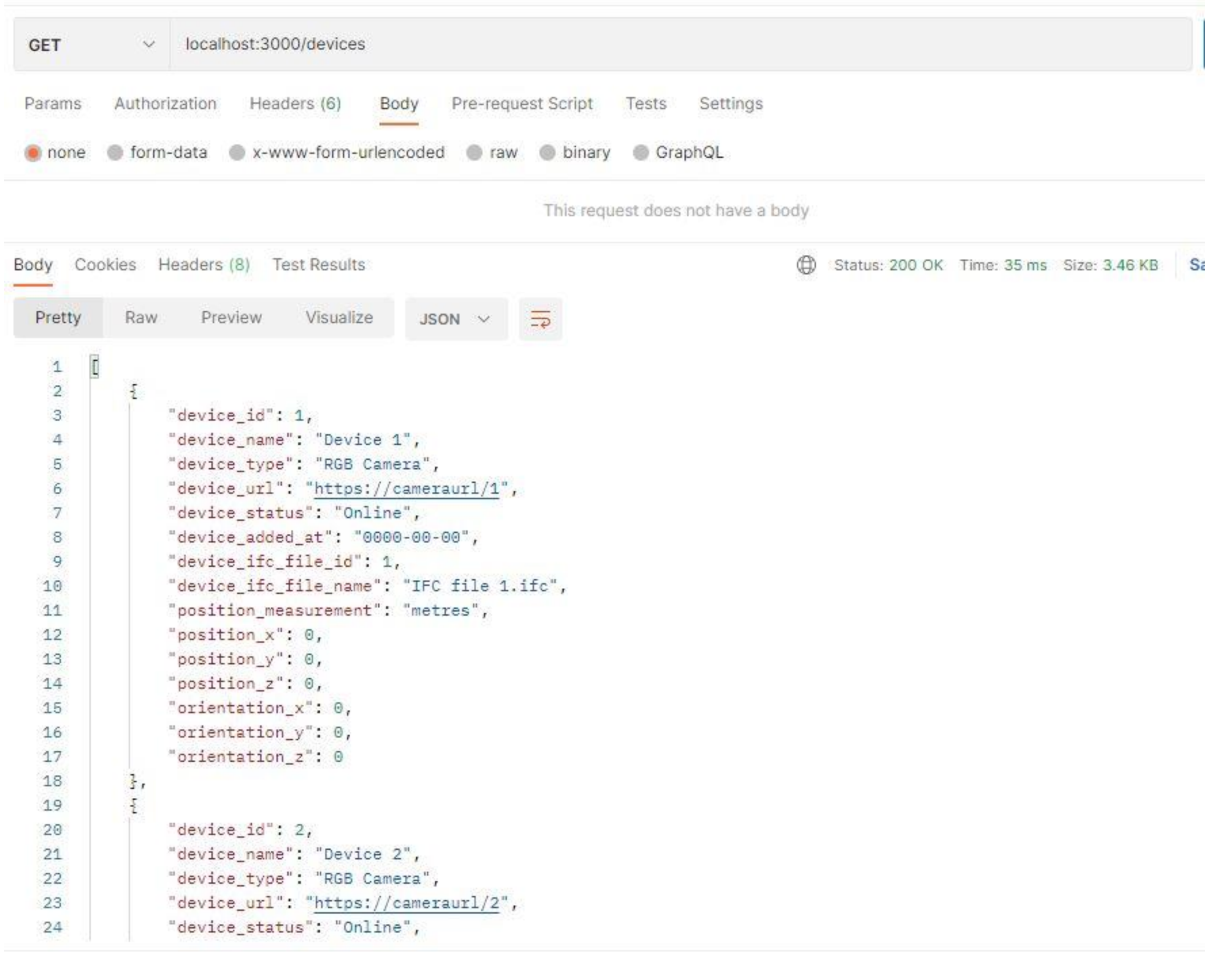
Table 5 - Sub-properties of orientation

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 with different names in 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.



```
[
    {
        "device_id": 1,
        "device_name": "Device 1",
        "device_type": "RGB Camera",
        "device_url": "https://cameraurl/1",
        "device_status": "Online",
        "device_added_at": "0000-00-00",
        "device_ifc_file_id": 1,
        "device_ifc_file_name": "IFC file 1.ifc",
        "position_measurement": "metres",
        "position_x": 0,
        "position_y": 0,
        "position_z": 0,
        "orientation_x": 0,
        "orientation_y": 0,
        "orientation_z": 0
    },
    {
        "device_id": 2,
        "device_name": "Device 2",
        "device_type": "RGB Camera",
        "device_url": "https://cameraurl/2",
        "device_status": "Online",
```

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 summarized in Table 6.

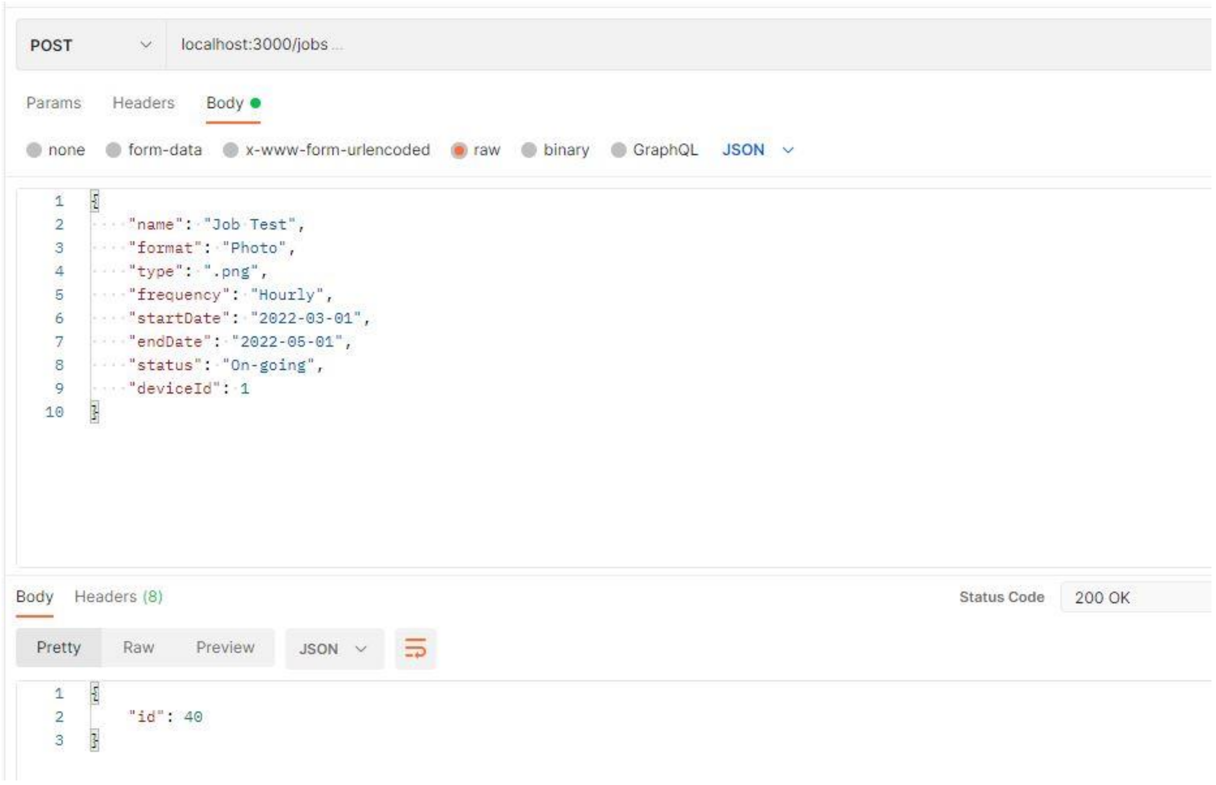
Table 6 - Job requests

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}}/rawImage	JSON
Upload the raw point cloud of a job	API endpoint to upload the raw point cloud of a job	PUT	**PUT** localhost:3000/jobs/{{:id}}/rawPointCloud	JSON
Get the raw data of a job	API endpoint to get the raw data 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}}/processedImage	JSON
Upload the processed point cloud of a job	API endpoint to upload the processed point cloud of a job	PUT	**PUT** localhost:3000/jobs/{{:id}}/processedPointCloud	JSON
Get the processed data of a job	API endpoint to get the processed data of a specific job	GET	**GET** localhost:3000/jobs/{{:id}}/processed	File of the processed photo or pointcloud

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.



```
{
    "name": "Job Test",
    "format": "Photo",
    "type": ".png",
    "frequency": "Hourly",
    "startDate": "2022-03-01",
    "endDate": "2022-05-01",
    "status": "On-going",
    "deviceId": 1
}
```

```
{
    "id": 40
}
```

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.

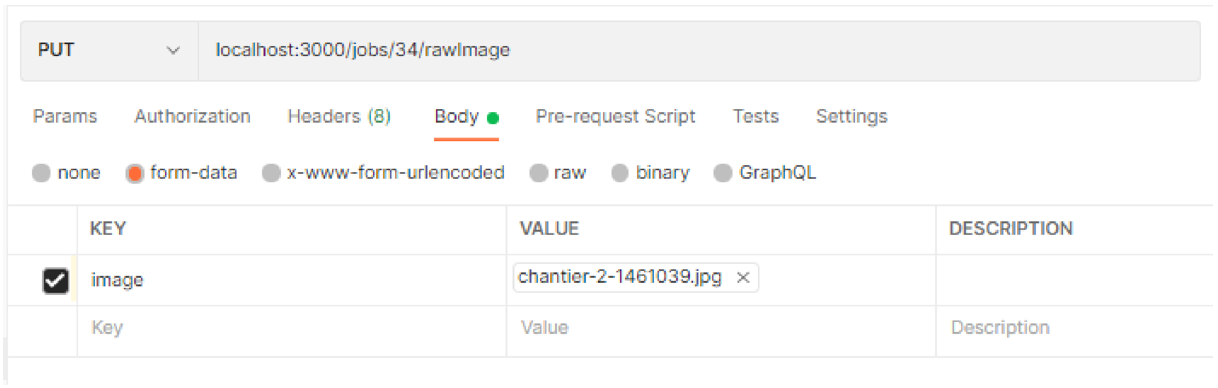
Table 7 - Job properties

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.


PUT localhost:3000/jobs/34/rawImage

Params | Authorization | Headers (8) | Body ● | Pre-request Script | Tests | Settings

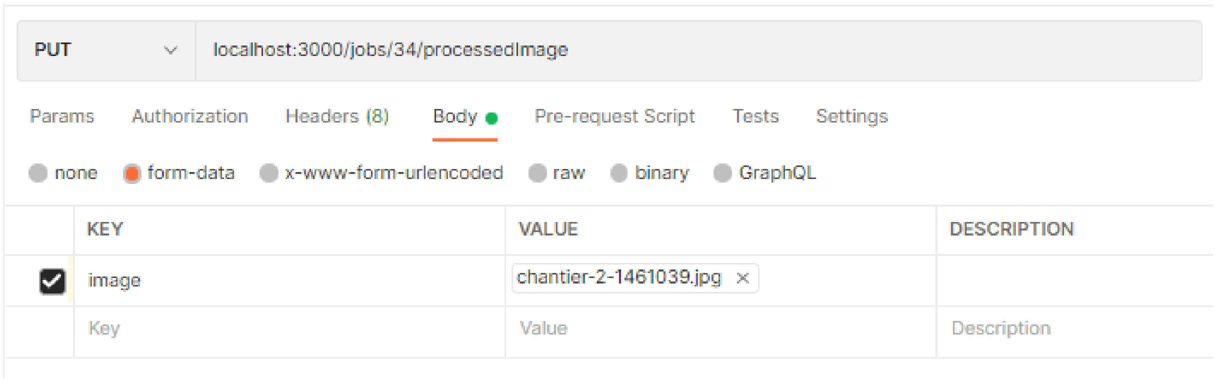
○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
✓	image	chantier-2-1461039.jpg ×	
	Key	Value	Description

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.


PUT localhost:3000/jobs/34/processedImage

Params | Authorization | Headers (8) | Body ● | Pre-request Script | Tests | Settings

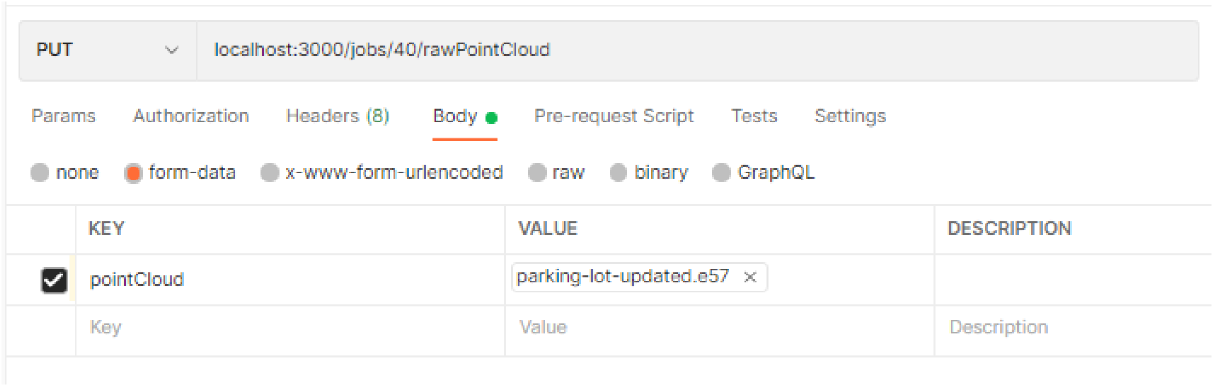
○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
✓	image	chantier-2-1461039.jpg ×	
	Key	Value	Description

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

Example 4: Upload the raw point cloud of a job

The API endpoint for uploading the raw point cloud of a job is illustrated in Figure 11. 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 “pointCloud” and its “value” being the file of the point cloud.


PUT localhost:3000/jobs/40/rawPointCloud

Params | Authorization | Headers (8) | Body ● | Pre-request Script | Tests | Settings

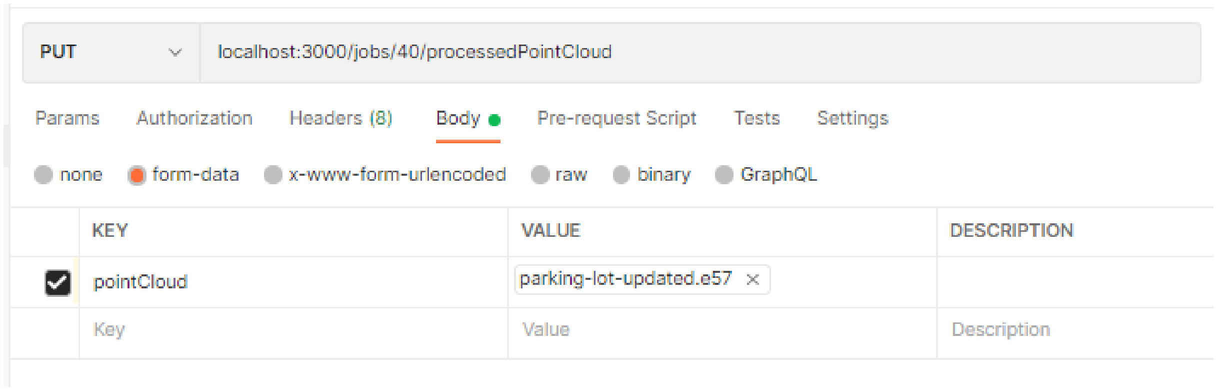
○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
✓	pointCloud	parking-lot-updated.e57 ×	
	Key	Value	Description

Figure 11 - API endpoint for uploading the raw point cloud of a job

Example 5: Upload the processed point cloud of a job

The API endpoint for uploading the processed point cloud of a job is depicted in Figure 12. 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 "pointCloud" and its "value" being the file of the point cloud.


PUT localhost:3000/jobs/40/processedPointCloud

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

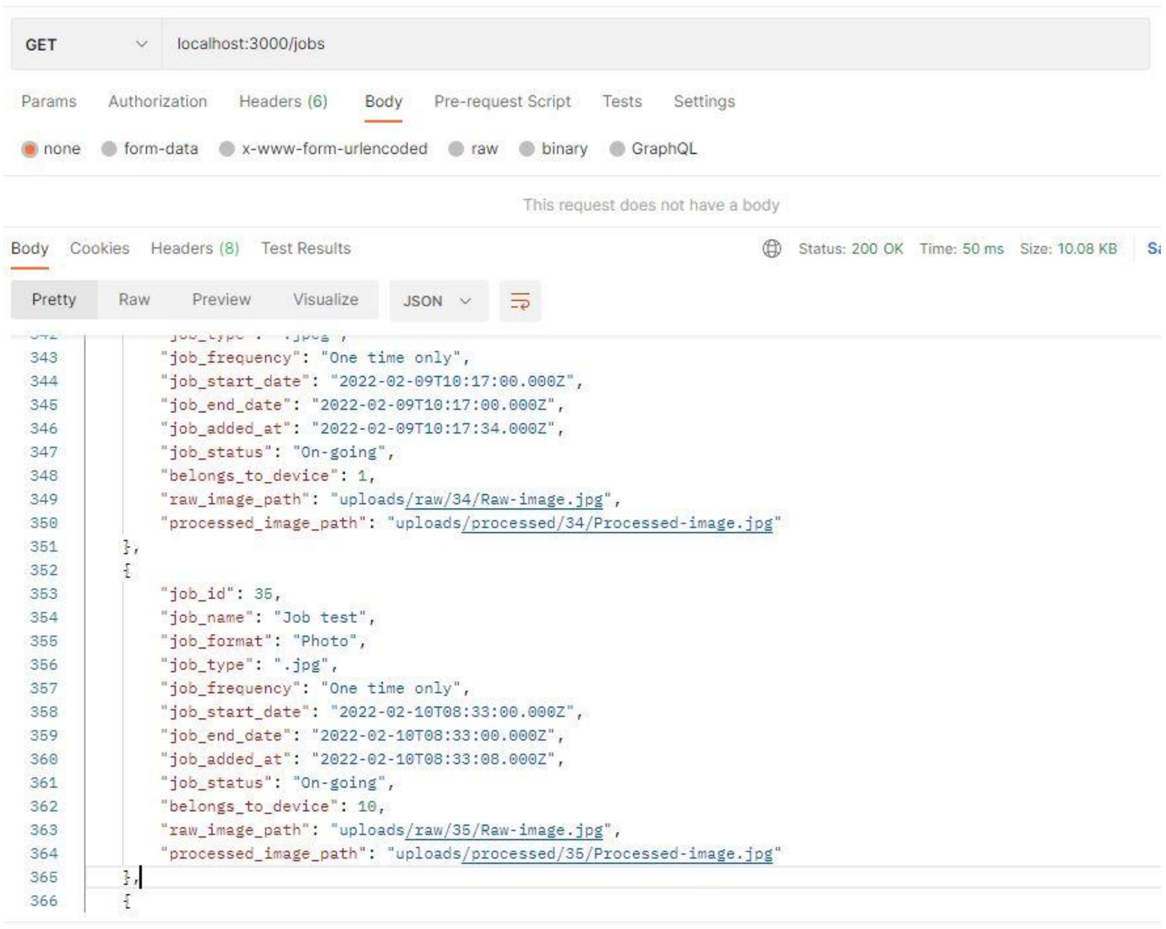
○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	KEY	VALUE	DESCRIPTION
✓	pointCloud	parking-lot-updated.e57 ×	
	Key	Value	Description

Figure 12 - API endpoint for uploading the processed point cloud of a job

Example 4: Get all jobs

The API endpoint for getting all the jobs is depicted in Figure 13. A successful request returns HTTP 200 Status and the properties of all jobs are displayed on the screen.



This request does not have a body

Status: 200 OK Time: 50 ms Size: 10.08 KB

```
        "job_frequency": "One time only",
        "job_start_date": "2022-02-09T10:17:00.000Z",
        "job_end_date": "2022-02-09T10:17:00.000Z",
        "job_added_at": "2022-02-09T10:17:34.000Z",
        "job_status": "On-going",
        "belongs_to_device": 1,
        "raw_image_path": "uploads/raw/34/Raw-image.jpg",
        "processed_image_path": "uploads/processed/34/Processed-image.jpg"
    },
    {
        "job_id": 35,
        "job_name": "Job test",
        "job_format": "Photo",
        "job_type": ".jpg",
        "job_frequency": "One time only",
        "job_start_date": "2022-02-10T08:33:00.000Z",
        "job_end_date": "2022-02-10T08:33:00.000Z",
        "job_added_at": "2022-02-10T08:33:08.000Z",
        "job_status": "On-going",
        "belongs_to_device": 10,
        "raw_image_path": "uploads/raw/35/Raw-image.jpg",
        "processed_image_path": "uploads/processed/35/Processed-image.jpg"
    },
    {
```

Figure 13 - API endpoint for getting all the jobs- request and response

2.3.3.3 *Image Processing Requests*

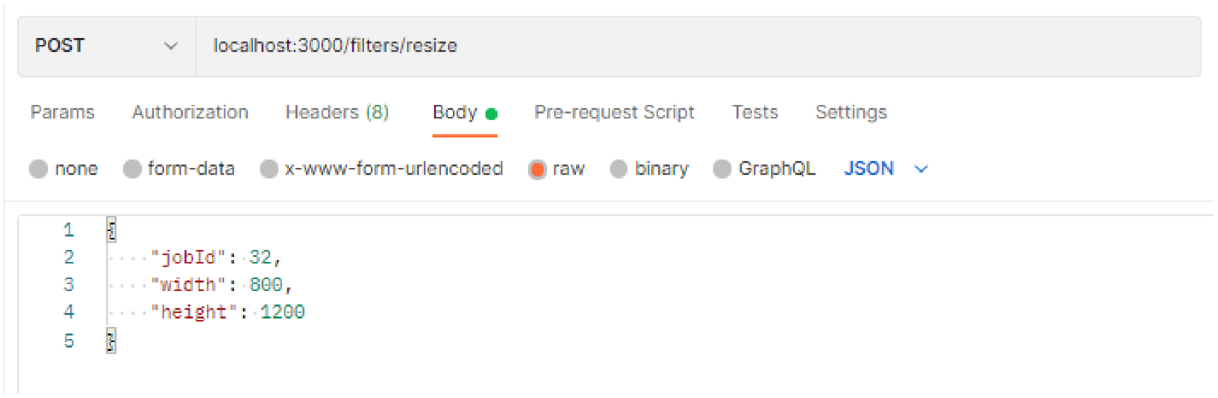
The Device requests that can be made are summarized in Table 8.

Table 8 - Image Processing requests

Request	Description	Method	Endpoints	Response
Start image processing	API endpoint to start the image processing and get the raw image that is related to a job	POST	**POST** localhost:3000/filters/start	Raw photo
Resize the image	API endpoint to resize the image.	POST	**POST** localhost:3000/filters/resize	Processed photo
Blur the image	API endpoint to blur the image.	POST	**POST** localhost:3000/filters/blur	Processed photo
Change contrast and brightness of the image	API endpoint to change the contrast and brightness of the image.	POST	**POST** localhost:3000/filters/contrastAndBrightness	Processed photo
Crop the image	API endpoint to crop the image.	POST	**POST** localhost:3000/filters/crop	Processed photo
Reset the image	API endpoint to reset the image.	POST	**POST** localhost:3000/filters/reset	Raw photo

Example 1: Resize the image

The API endpoint for resizing the image related to a job is depicted in Figure 14. A successful request returns HTTP 200 Status and the processed image.


POST localhost:3000/filters/resize

```
{
    "jobId": 32,
    "width": 800,
    "height": 1200
}
```

Figure 14 - API endpoint for resizing an image related to a job

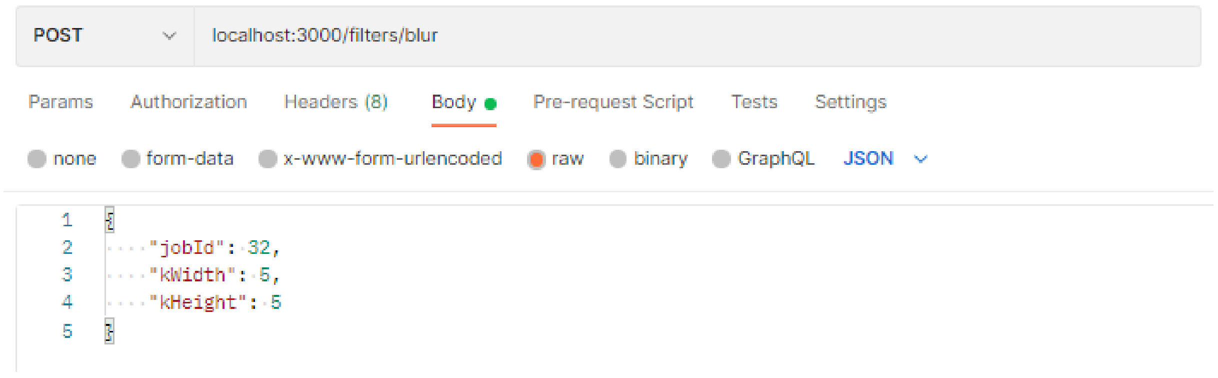
The properties that should be passed in the JSON body request are presented in Table 9.

Table 9 - Resize properties

Property	Description
jobId	The unique identifier of the related job
width	The desirable width of the processed image
height	The desirable height of the processed image

Example 2: Blur the image

The API endpoint for blurring the image [12] [13] related to a job is depicted in Figure 15. A successful request returns HTTP 200 Status and the processed image.


POST localhost:3000/filters/blur

```
{
    "jobId": 32,
    "kWidth": 5,
    "kHeight": 5
}
```

Figure 15 - API endpoint for blurring an image related to a job

The properties that should be passed in the JSON body request are presented in Table 10.

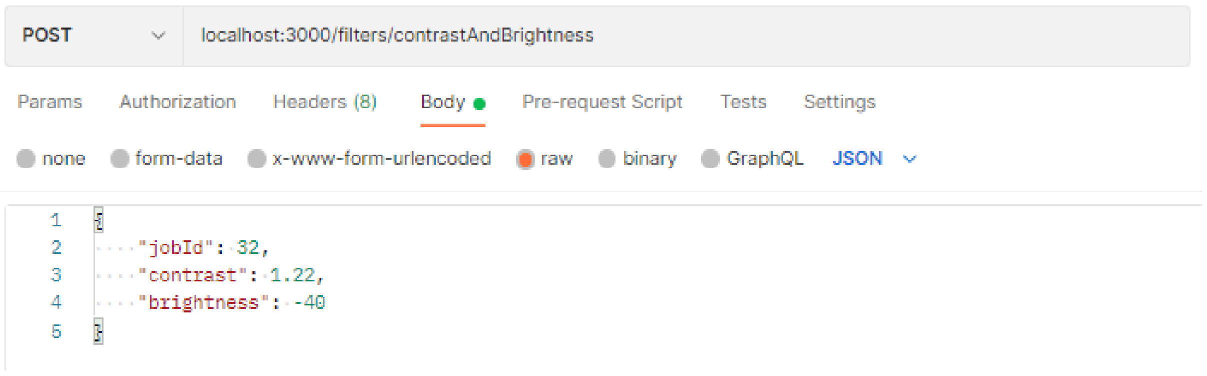
Table 10 - Blur properties

Property	Description
jobId	The unique identifier of the related job
kWidth	The width of the kernel used for Gaussian Blurring (should be odd)

kHeight	The height of the kernel used for Gaussian Blurring (should be odd)

Example 3: Change contrast and brightness of the image

The API endpoint for adjusting the contrast and brightness of the image [14] related to a job is depicted in Figure 16. A successful request returns HTTP 200 Status and the processed image.



```
{
    "jobId": 32,
    "contrast": 1.22,
    "brightness": -40
}
```

Figure 16 -API endpoint for changing contrast and brightness of an image related to a job

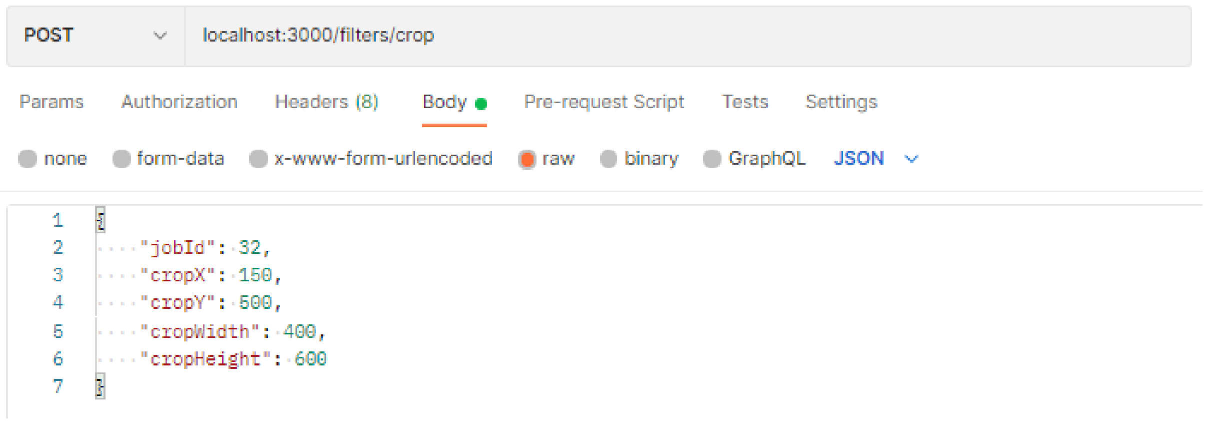
The properties that should be passed in the JSON body request are presented in Table 11.

Table 11 – Contrast and brightness properties

Property	Description
jobId	The unique identifier of the related job
contrast	The desirable contrast of the processed image (values from 0.30 – 2.00)
brightness	The desirable brightness of the processed image (values from -127 – 127)

Example 4: Crop the image

The API endpoint for cropping the image related to a job is depicted in Figure 17. A successful request returns HTTP 200 Status and the processed image.



```
{
    "jobId": 32,
    "cropX": 150,
    "cropY": 500,
    "cropWidth": 400,
    "cropHeight": 600
}
```

Figure 17 - API endpoint for cropping an image related to a job

The properties that should be passed in the JSON body request are presented in Table 12.

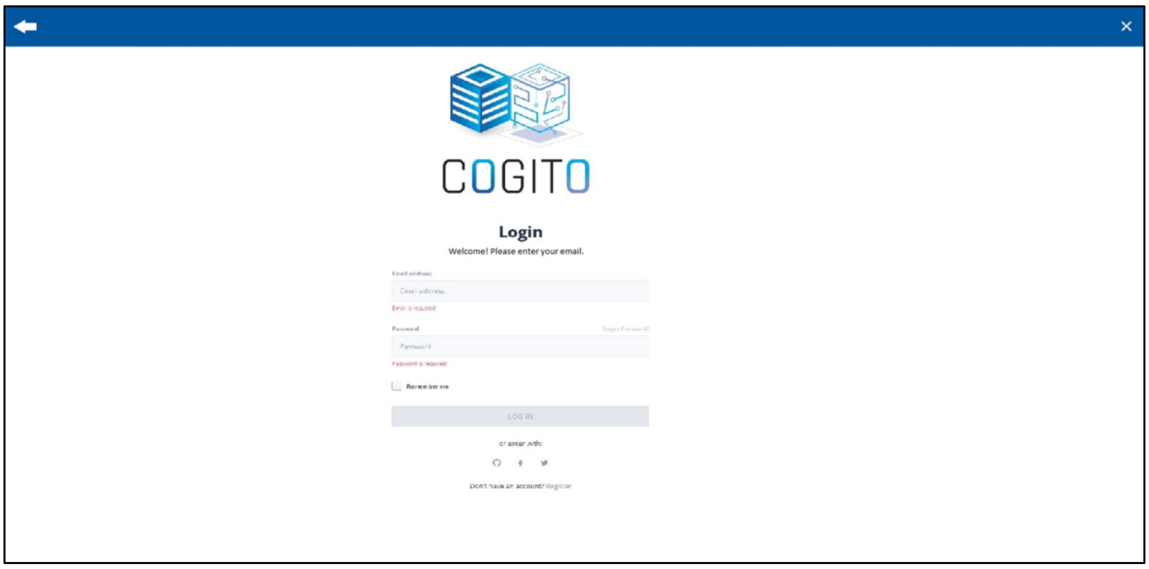
Table 12 - Cropping properties

Property	Description
jobId	The unique identifier of the related job

cropX	X position of the starting cropping point (upper left point)
cropY	Y position of the starting cropping point (upper left point)
cropWidth	The desirable cropping width
cropHeight	The desirable cropping height

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 18.



Welcome! Please enter your email.

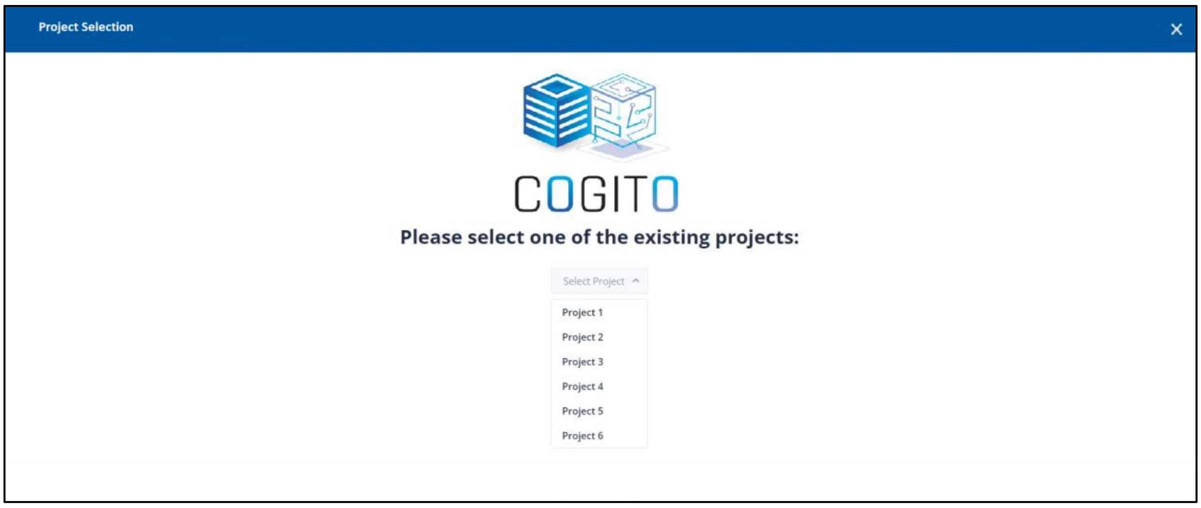
Email is required

Password is required

Don't have an account? Register

Figure 18 - Visual Data Pre-processing Module user's login page

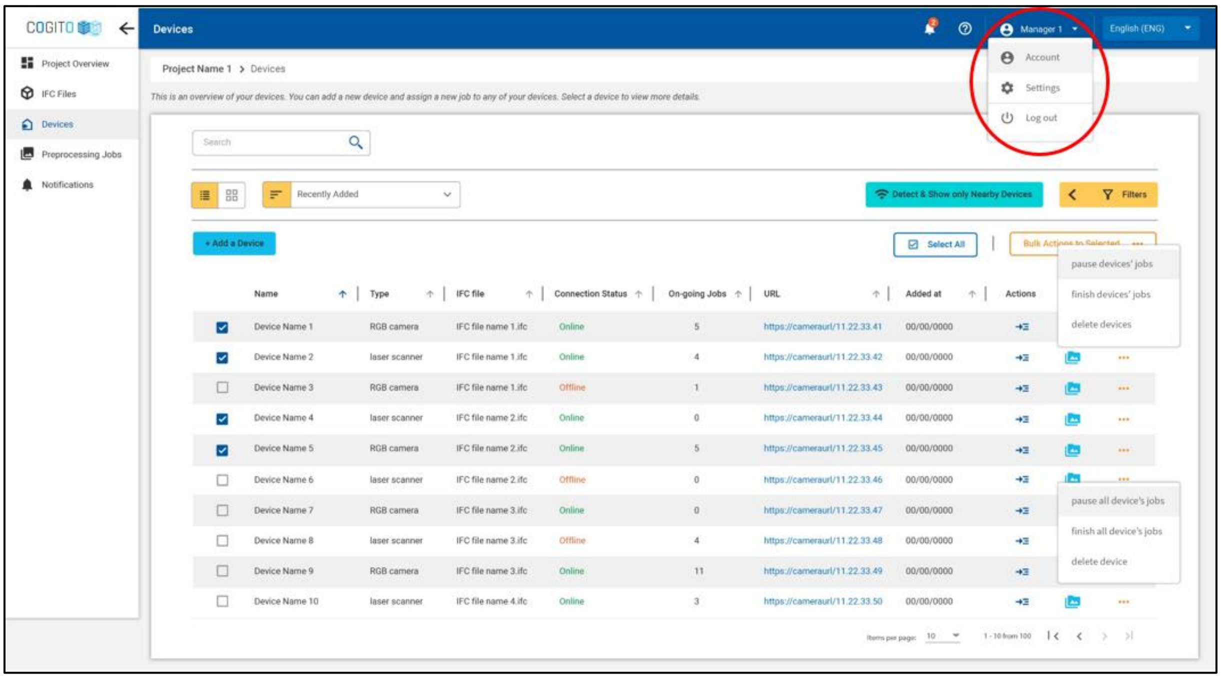
In addition, the desired project can be selected from a list of projects provided by the DT Platform, based on the user's credentials (see Figure 19).



Please select one of the existing projects:

Figure 19 - Project Selection

In Figure 20, the accounts' options are illustrated. The user can make changes regarding their account profile and configure the general settings according to preference.

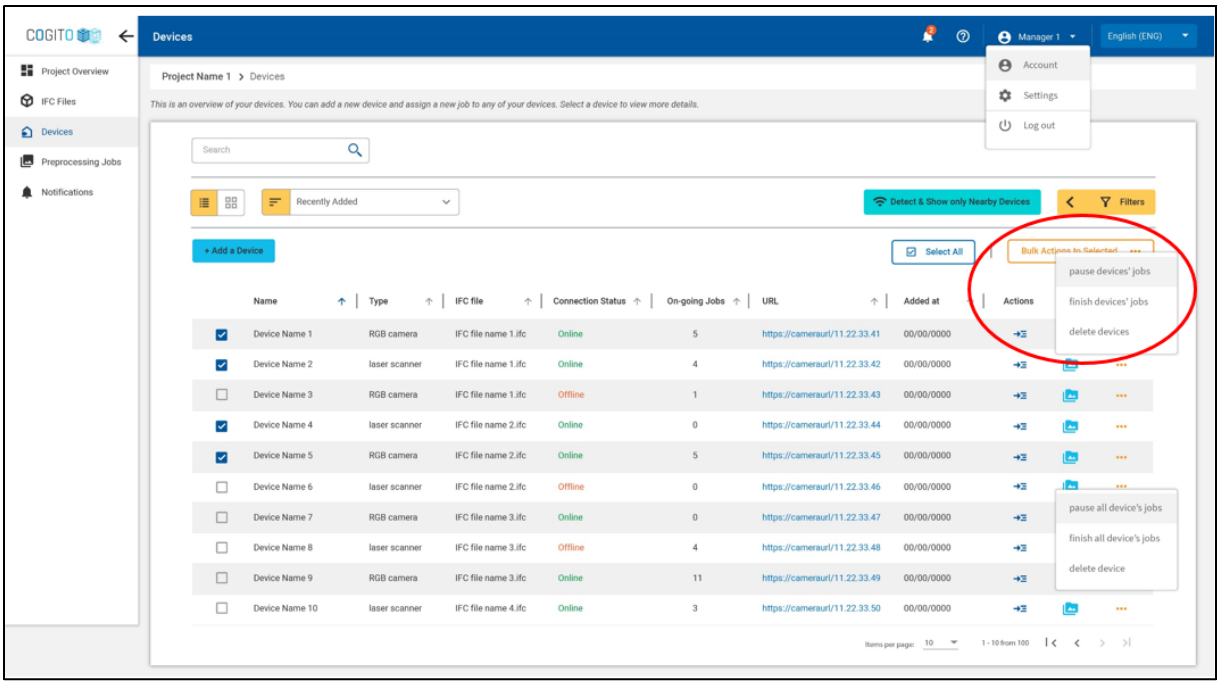


This is an overview of your devices. You can add a new device and assign a new job to any of your devices. Select a device to view more details.

	Name	Type	IFC file	Connection Status	On-going Jobs	URL	Added at	Actions
✓	Device Name 1	RGB camera	IFC file name 1.ifc	Online	5	https://cameraurl/11.22.33.41	00/00/0000	
✓	Device Name 2	laser scanner	IFC file name 1.ifc	Online	4	https://cameraurl/11.22.33.42	00/00/0000	
☐	Device Name 3	RGB camera	IFC file name 1.ifc	Offline	1	https://cameraurl/11.22.33.43	00/00/0000	
✓	Device Name 4	laser scanner	IFC file name 2.ifc	Online	0	https://cameraurl/11.22.33.44	00/00/0000	
✓	Device Name 5	RGB camera	IFC file name 2.ifc	Online	5	https://cameraurl/11.22.33.45	00/00/0000	
☐	Device Name 6	laser scanner	IFC file name 2.ifc	Offline	0	https://cameraurl/11.22.33.46	00/00/0000	
☐	Device Name 7	RGB camera	IFC file name 3.ifc	Online	0	https://cameraurl/11.22.33.47	00/00/0000	
☐	Device Name 8	laser scanner	IFC file name 3.ifc	Offline	4	https://cameraurl/11.22.33.48	00/00/0000	
☐	Device Name 9	RGB camera	IFC file name 3.ifc	Online	11	https://cameraurl/11.22.33.49	00/00/0000	
☐	Device Name 10	laser scanner	IFC file name 4.ifc	Online	3	https://cameraurl/11.22.33.50	00/00/0000	

Figure 20 - 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 21.

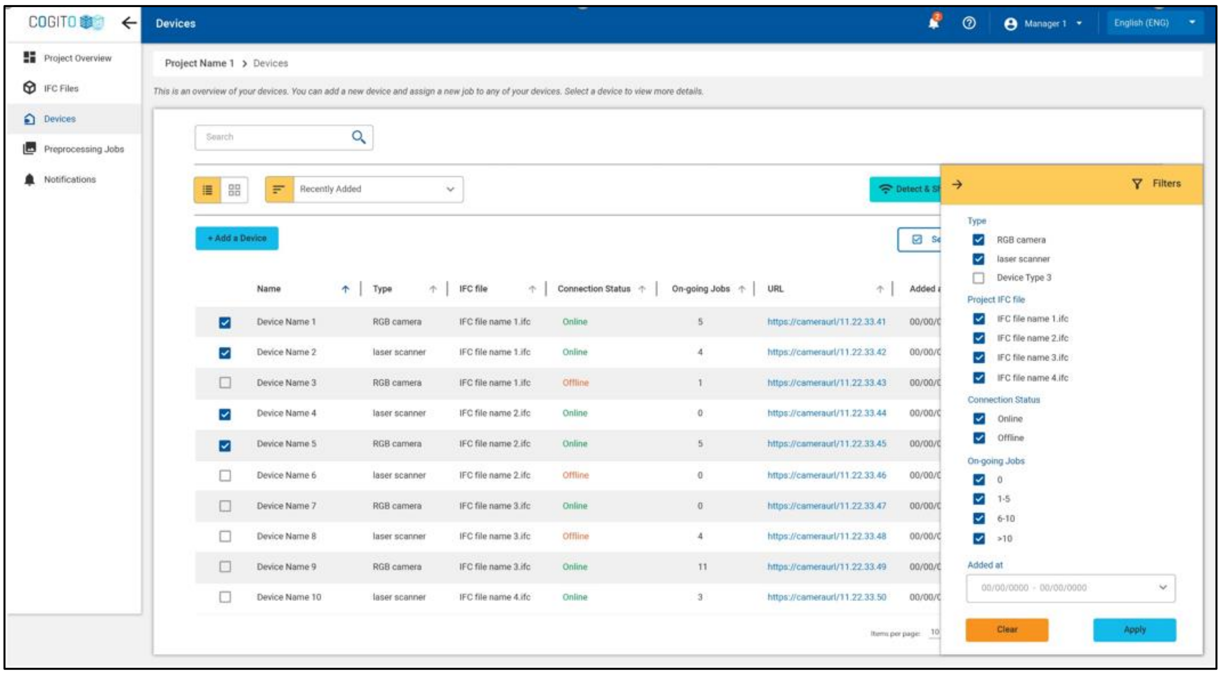


This is an overview of your devices. You can add a new device and assign a new job to any of your devices. Select a device to view more details.

	Name	Type	IFC file	Connection Status	On-going Jobs	URL	Added at	Actions
✓	Device Name 1	RGB camera	IFC file name 1.ifc	Online	5	https://cameraurl/11.22.33.41	00/00/0000	
✓	Device Name 2	laser scanner	IFC file name 1.ifc	Online	4	https://cameraurl/11.22.33.42	00/00/0000	
☐	Device Name 3	RGB camera	IFC file name 1.ifc	Offline	1	https://cameraurl/11.22.33.43	00/00/0000	
✓	Device Name 4	laser scanner	IFC file name 2.ifc	Online	0	https://cameraurl/11.22.33.44	00/00/0000	
✓	Device Name 5	RGB camera	IFC file name 2.ifc	Online	5	https://cameraurl/11.22.33.45	00/00/0000	
☐	Device Name 6	laser scanner	IFC file name 2.ifc	Offline	0	https://cameraurl/11.22.33.46	00/00/0000	
☐	Device Name 7	RGB camera	IFC file name 3.ifc	Online	0	https://cameraurl/11.22.33.47	00/00/0000	
☐	Device Name 8	laser scanner	IFC file name 3.ifc	Offline	4	https://cameraurl/11.22.33.48	00/00/0000	
☐	Device Name 9	RGB camera	IFC file name 3.ifc	Online	11	https://cameraurl/11.22.33.49	00/00/0000	
☐	Device Name 10	laser scanner	IFC file name 4.ifc	Online	3	https://cameraurl/11.22.33.50	00/00/0000	

Figure 21 - 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 22).



Project Name 1 > Devices

This is an overview of your devices. You can add a new device and assign a new job to any of your devices. Select a device to view more details.

	Name	Type	IFC file	Connection Status	On-going Jobs	URL	Added
✓	Device Name 1	RGB camera	IFC file name 1.ifc	Online	5	https://cameraurl/11.22.33.41	00/00/0
✓	Device Name 2	laser scanner	IFC file name 1.ifc	Online	4	https://cameraurl/11.22.33.42	00/00/0
☐	Device Name 3	RGB camera	IFC file name 1.ifc	Offline	1	https://cameraurl/11.22.33.43	00/00/0
✓	Device Name 4	laser scanner	IFC file name 2.ifc	Online	0	https://cameraurl/11.22.33.44	00/00/0
✓	Device Name 5	RGB camera	IFC file name 2.ifc	Online	5	https://cameraurl/11.22.33.45	00/00/0
☐	Device Name 6	laser scanner	IFC file name 2.ifc	Offline	0	https://cameraurl/11.22.33.46	00/00/0
☐	Device Name 7	RGB camera	IFC file name 3.ifc	Online	0	https://cameraurl/11.22.33.47	00/00/0
☐	Device Name 8	laser scanner	IFC file name 3.ifc	Offline	4	https://cameraurl/11.22.33.48	00/00/0
☐	Device Name 9	RGB camera	IFC file name 3.ifc	Online	11	https://cameraurl/11.22.33.49	00/00/0
☐	Device Name 10	laser scanner	IFC file name 4.ifc	Online	3	https://cameraurl/11.22.33.50	00/00/0

Items per page: 10

Type
- ✓ RGB camera
- ✓ laser scanner
- ☐ Device Type 3

Project IFC file
- ✓ IFC file name 1.ifc
- ✓ IFC file name 2.ifc
- ✓ IFC file name 3.ifc
- ✓ IFC file name 4.ifc

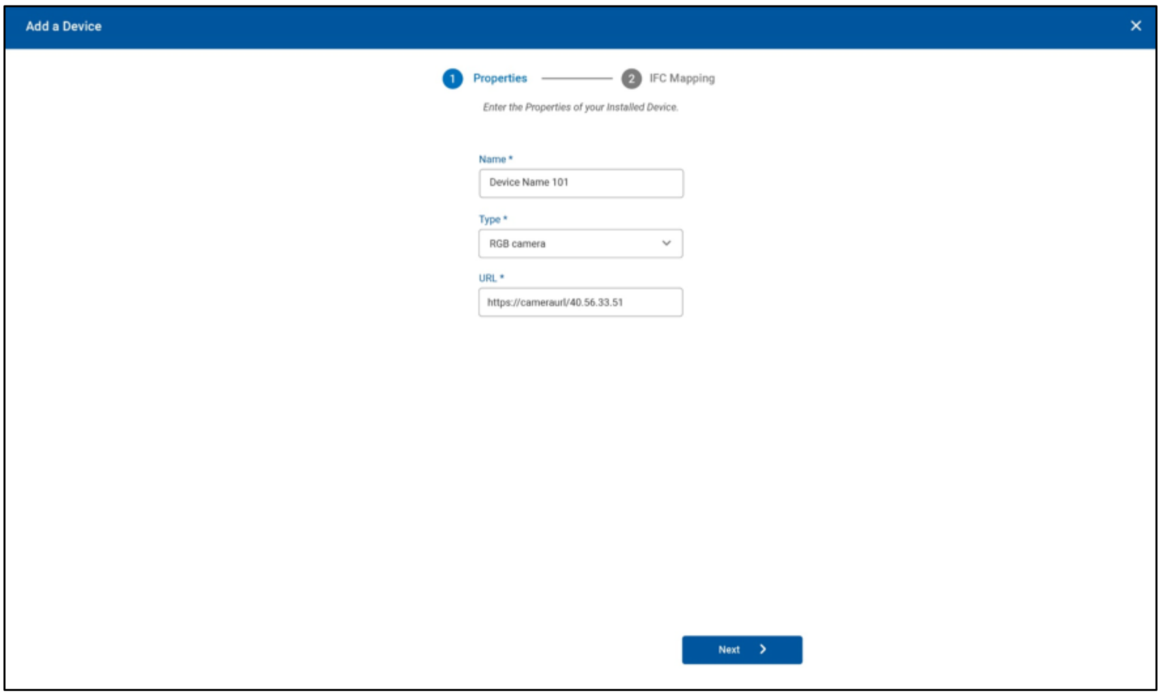
Connection Status
- ✓ Online
- ✓ Offline

On-going Jobs
- ✓ 0
- ✓ 1-5
- ✓ 6-10
- ✓ >10

Added at: 00/00/0000 - 00/00/0000

Figure 22 - 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 23.



Enter the Properties of your Installed Device.

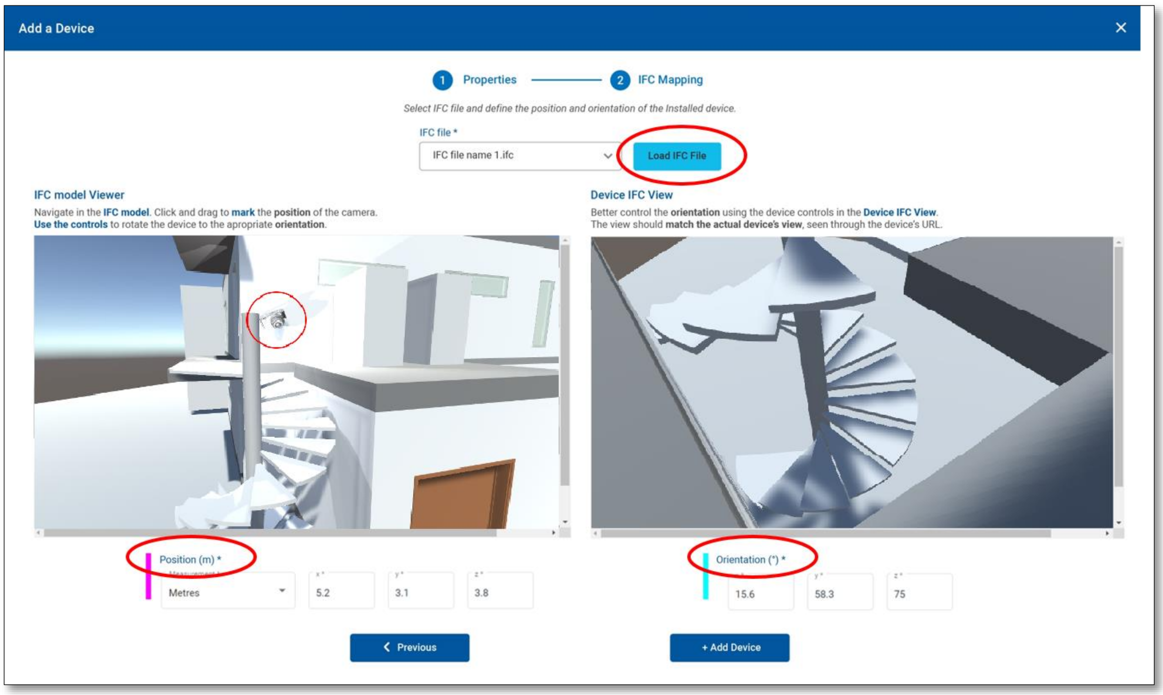
Name *: Device Name 101

Type *: RGB camera

URL *: https://cameraurl/40.56.33.51

Figure 23 - Adding 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. The user can navigate to the IFC model Viewer (Figure 24 left window) using the arrow keys of the keyboard or the left and right mouse buttons to control their direction and orientation in the model. In addition, they can click and drag to place a capturing device in a chosen location in the 3D model. Subsequently, at the Device IFC View (Figure 24 right window) the field of view of the placed virtual device is visualized. They can also adjust the orientation of the device and its field of view by rotating the placed device in the left window of the screen. At this point, the Device IFC View should match the actual device's view, as seen through the device's URL.



Select IFC file and define the position and orientation of the Installed device.

IFC file *: IFC file name 1.ifc

Navigate in the **IFC model**. Click and drag to **mark** the **position** of the camera.
Use the controls to rotate the device to the apropriate **orientation**.

Better control the **orientation** using the device controls in the **Device IFC View**.
The view should **match the actual device's view**, seen through the device's URL.

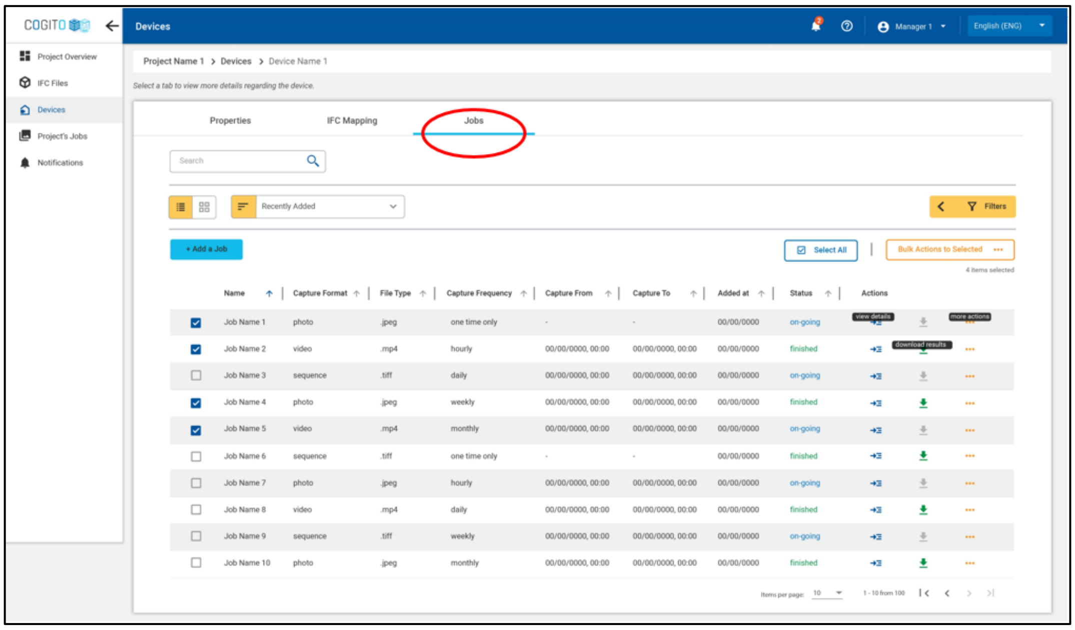
Position (m) *: Metres, x: 5.2, y: 3.1, z: 3.8

Orientation (°) *: x: 15.6, y: 58.3, z: 75

Figure 24 - IFC Mapping of the new device

It is worth mentioning that the matching level of the two views depends on the precision of the registration (i.e., the precision of the alignment between the 3D BIM model of the site and the physical site). Since a precise and thorough registration process can achieve spatial accuracy of few centimetres, the deviation between the two views is expected to be negligible. Furthermore, at the two fields in the bottom screen of Figure 24 (position and orientation) the respective coordinates are recorded in order to be sent to the DT Platform as metadata of this specific device.

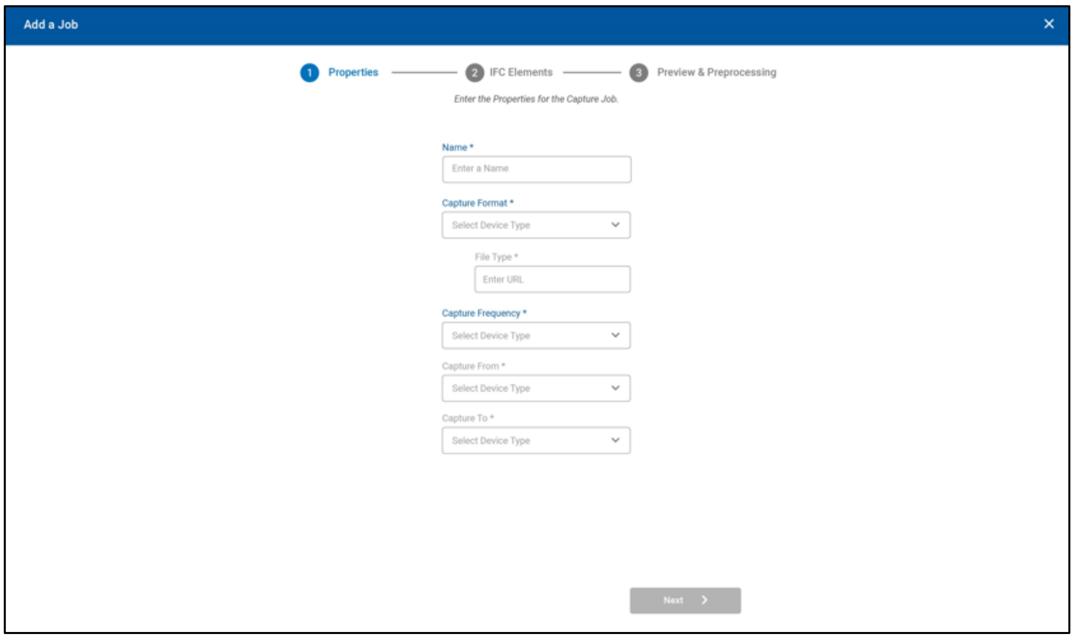
By clicking the Jobs tab, the available jobs for a specific device are displayed. Every job entry has its own buttons (view details, download results and more actions). There are also general UI buttons for filters, for sorting the jobs, and for adding a new one, as illustrated in Figure 25.



	Name	Capture Format	File Type	Capture Frequency	Capture From	Capture To	Added at	Status
☑	Job Name 1	photo	.jpeg	one time only	-	-	00/00/0000	on-going
☑	Job Name 2	video	.mp4	hourly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	finished
☐	Job Name 3	sequence	.tiff	daily	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	on-going
☑	Job Name 4	photo	.jpeg	weekly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	finished
☑	Job Name 5	video	.mp4	monthly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	on-going
☐	Job Name 6	sequence	.tiff	one time only	-	-	00/00/0000	finished
☐	Job Name 7	photo	.jpeg	hourly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	on-going
☐	Job Name 8	video	.mp4	daily	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	finished
☐	Job Name 9	sequence	.tiff	weekly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	on-going
☐	Job Name 10	photo	.jpeg	monthly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	finished

Figure 25 - 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 26).



Enter the Properties for the Capture Job.

Name *
Enter a Name

Capture Format *
Select Device Type

File Type *
Enter URL

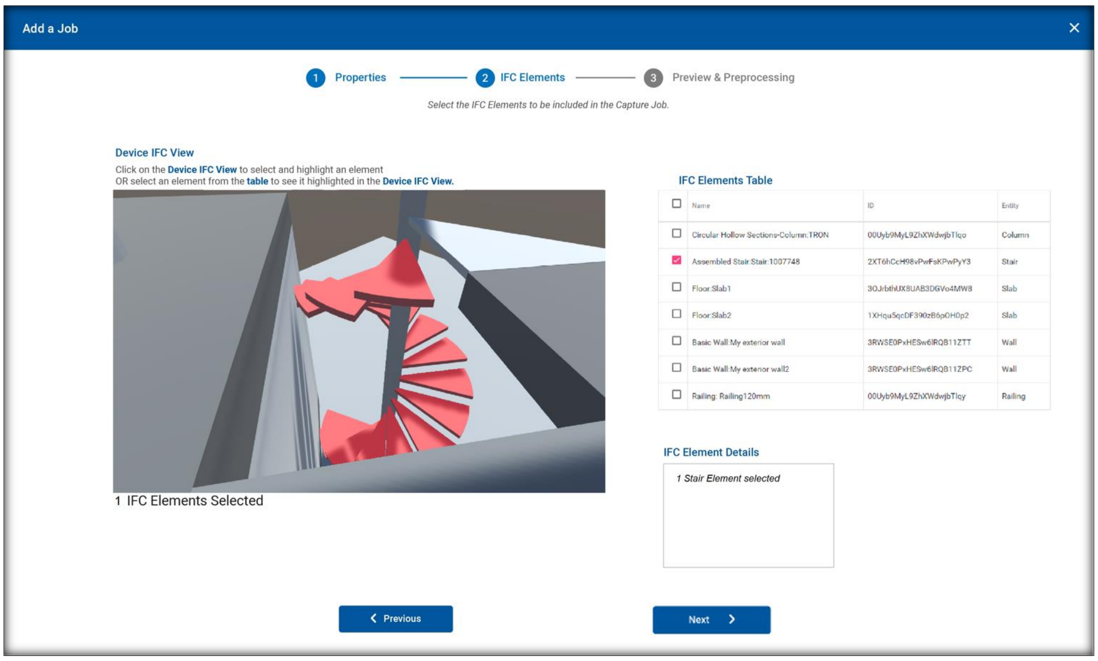
Capture Frequency *
Select Device Type

Capture From *
Select Device Type

Capture To *
Select Device Type

Figure 26 - Adding a new capturing job

In addition, the user can assign the relevant IFC components to this specific job, based on a specific work order, provided by the WODM tool. More specifically, they can select which IFC elements of the related work order are included in the capturing device's field of view by clicking on them. The selected IFC elements are highlighted to assist the user and if they are clicked again, they are deselected. This way, the visual data is registered with the actual structural data of the BIM model (see Figure 27).



Select the IFC Elements to be included in the Capture Job.

Device IFC View

Click on the **Device IFC View** to select and highlight an element
OR select an element from the **table** to see it highlighted in the **Device IFC View.**

1 IFC Elements Selected

IFC Elements Table

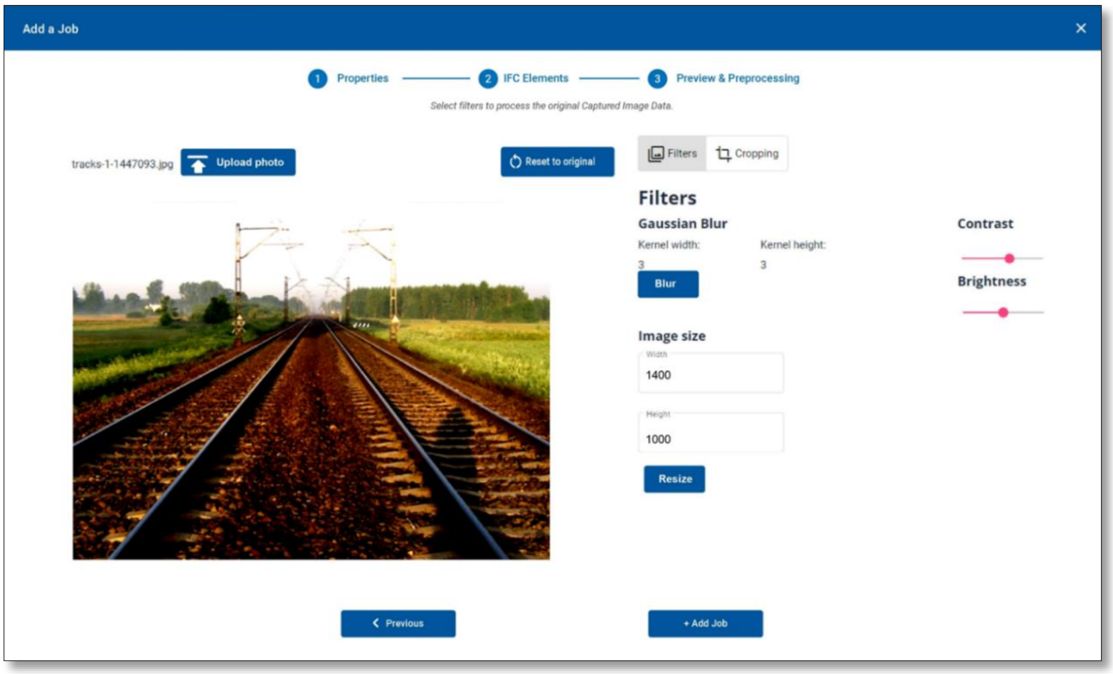
	Name	ID	Entity
☐	Circular Hollow Sections-Column:TRON	00Uyb9MyL9ZhXWdwjbTlqo	Column
✓	Assembled Stair:Stair:1007748	2XT6hCcH98vPwFsKPwPyY3	Stair
☐	Floor:Slab1	3OJrbthUX8UAB3DGVo4MW8	Slab
☐	Floor:Slab2	1XHqu5qcDF390zB6p0H0p2	Slab
☐	Basic Wall:My exterior wall	3RWSE0PxHESw6lRQB11ZTT	Wall
☐	Basic Wall:My exterior wall2	3RWSE0PxHESw6lRQB11ZPC	Wall
☐	Railing: Railing120mm	00Uyb9MyL9ZhXWdwjbTlqy	Railing

IFC Element Details

1 Stair Element selected

Figure 27 - Components attribution to a new capturing job

Afterwards, the user selects the desired 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. In Figure 28 and Figure 29 the contrast and resizing implementation and the cropping implementation are illustrated respectively.



Select filters to process the original Captured Image Data.

tracks-1-1447093.jpg

Filters

Gaussian Blur

Kernel width: 3

Kernel height: 3

Image size

Width: 1400

Height: 1000

Figure 28 - Image processing: contrast and resizing filter implementation

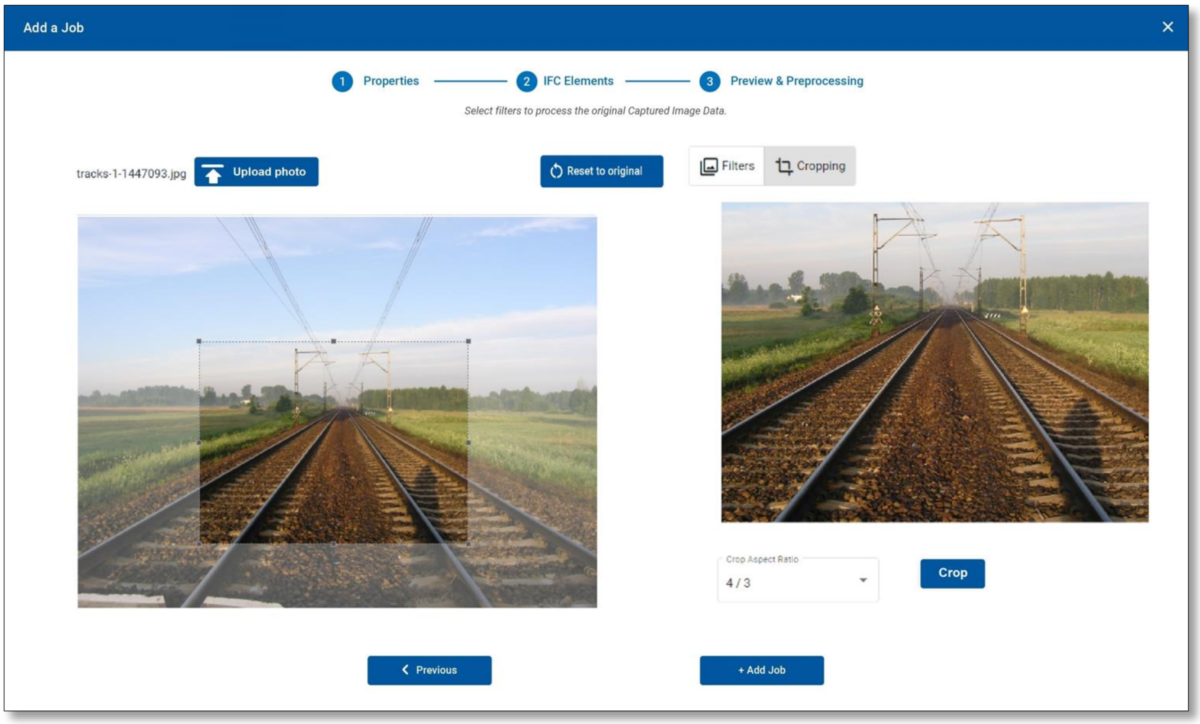


Figure 29 - Image processing: cropping implementation

As illustrated in Figure 30, in case of the as-built geometric data the GUI of the pre-processing step is slightly different. The user can upload the point cloud without previewing it in order to create a new job.

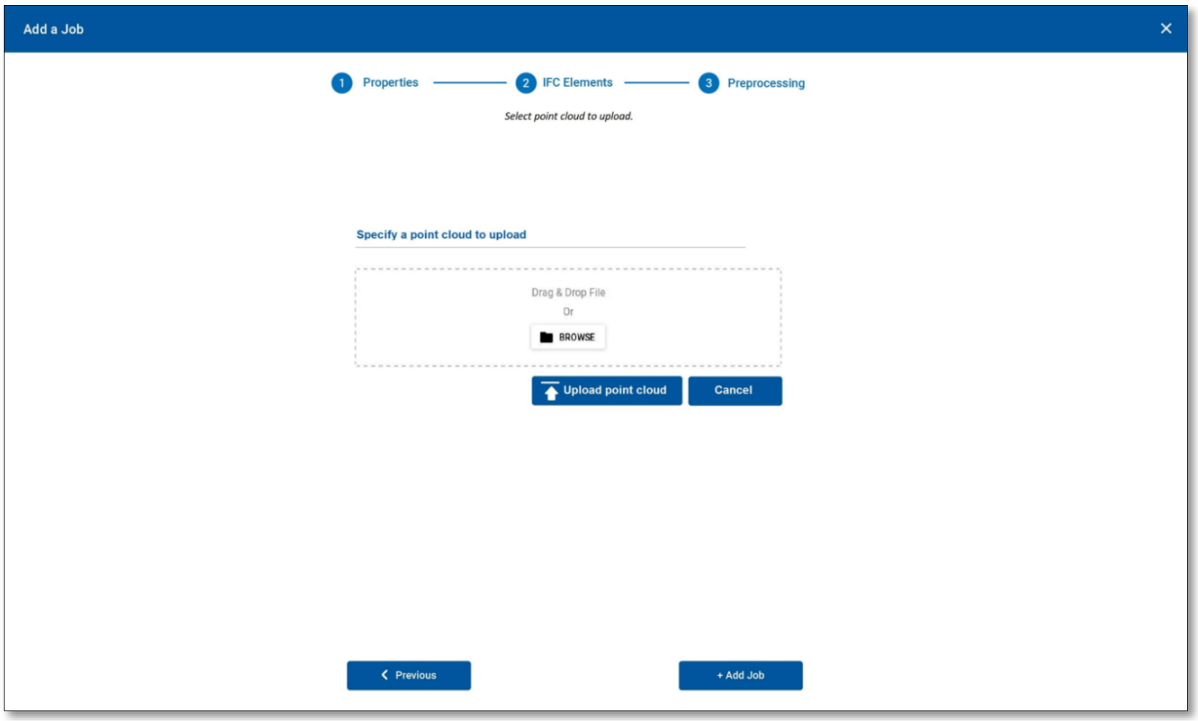
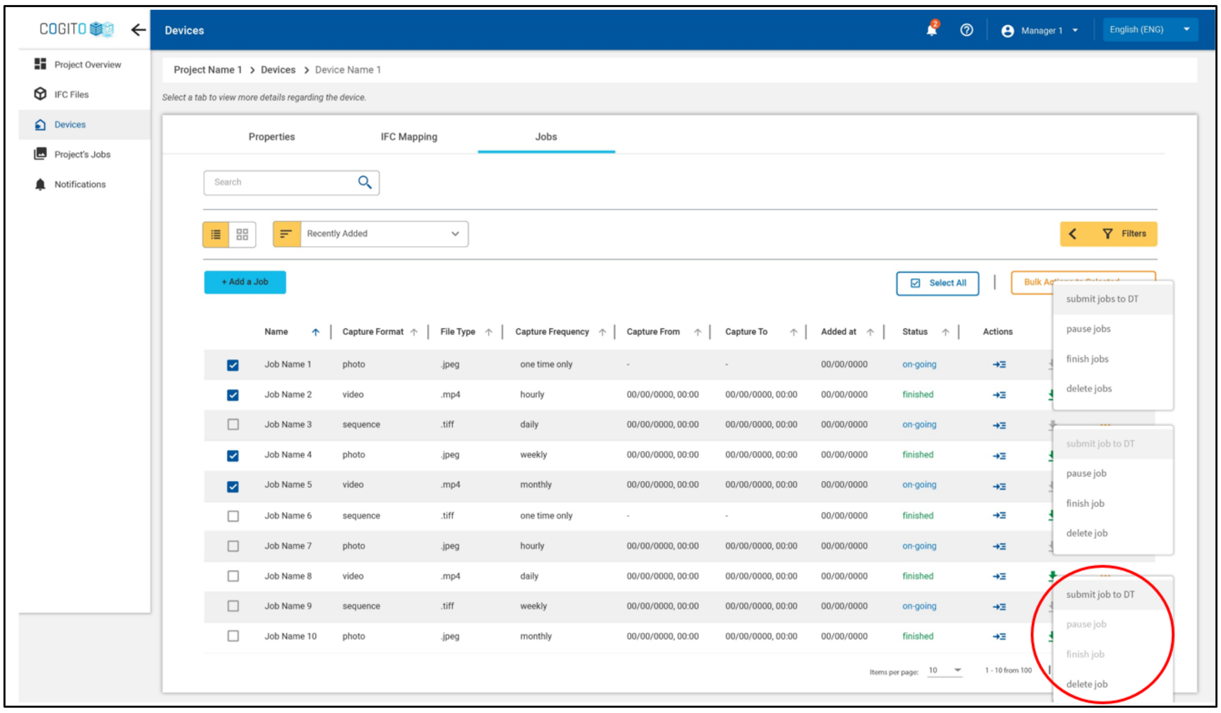


Figure 30 - Point cloud uploading

Once the job is added, it can be submitted to the DT Platform for further exploitation (see Figure 31).



Select a tab to view more details regarding the device.

	Name	Capture Format	File Type	Capture Frequency	Capture From	Capture To	Added at	Status	Actions
✓	Job Name 1	photo	.jpeg	one time only	-	-	00/00/0000	on-going	
✓	Job Name 2	video	.mp4	hourly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	finished	
☐	Job Name 3	sequence	.tiff	daily	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	on-going	
✓	Job Name 4	photo	.jpeg	weekly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	finished	
✓	Job Name 5	video	.mp4	monthly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	on-going	
☐	Job Name 6	sequence	.tiff	one time only	-	-	00/00/0000	finished	
☐	Job Name 7	photo	.jpeg	hourly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	on-going	
☐	Job Name 8	video	.mp4	daily	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	finished	
☐	Job Name 9	sequence	.tiff	weekly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	on-going	
☐	Job Name 10	photo	.jpeg	monthly	00/00/0000, 00:00	00/00/0000, 00:00	00/00/0000	finished	

Figure 31 - 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 in this final release is considered developed, given that all the necessary functionalities have been already implemented. This version focuses on handling the geometric as-built data (point clouds uploading) and completes the alpha version of the tool (D3.7) which had focused on the 2D image handling. In this second version, updates, optimizations and refinements had been carried out concerning main functionalities (such as job addition, data storage, IFC components attribution). In addition, the Image Processing Library sub-module was expanded (extra visual filters such as cropping and rescaling were added). Furthermore, the IFC 3D Viewer was developed from scratch to be used for visualizing the 3D BIM model online. Finally, the REST API was updated providing extra endpoints for the added functionalities (such as point cloud uploading). However, the Visual Data Pre-processing Module must interact with the DT Platform and all the capturing devices (including DigiTAR). The communication within all the COGITO components will be established during the T8.1 “End-to-end ICT System Integration, Testing and Refinement” from M18 to M34.

2.8 Requirements Coverage

Table 13 presents the stakeholders requirements that have been documented in D2.1 and are relevant to this component [15]. 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.). Therefore, COGI-WF-2 and COGI-WF-5 will be considered totally achieved at the end-to-end ICT COGITO System Integration.

Table 13 - 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 were updated based on D2.5 "COGITO system architecture v2" [16] and are presented in Table 14. 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 achieved, while several filters (such as contrast, brightness, Gaussian blurring, cropping and scaling) have been already implemented and finalized in the second release of the module. As for the Req.1-3, it is partially achieved, while the user is able to upload both images and point cloud, however the communication with the DT Platform will be implemented in the T8.1 "End-to-end ICT System Integration, Testing and Refinement" (M18-M34).

Concerning the non-functional requirements, the status of Req-2.1 is now considered to be achieved given that the web application has been updated, improved and adjusted to meet the requirements. Req-2.2, Req-2.3, Req-2.6 and Req-2.7 are covered since MySQL considers being well structured, one of the most secure and reliable database management systems and ideal for applications that require data integrity (as already analysed in sub-section 2.1.2). In addition, Req-2.4 is considered to be achieved since the architecture and technologies used in Visual Data Pre-processing module allow for both vertical and horizontal scaling [17]. Finally, the implementation of the component was accomplished using state-of-the-art software frameworks and libraries; hence, the performance of the tool (Req-2.5) is considered being optimized.

Table 14 - Visual Data Pre-processing tool: Functional and Non-Functional Requirements coverage from D2.5

ID	Description	Type	Status
Req-1.1	Stores the raw data in a local database	Functional	Achieved
Req-1.2	Processes the raw visual data input (Smoothing, enhancing)	Functional	Achieved
Req-1.3	Sends images or point clouds and associated data to DT Platform	Functional	Partially achieved
Req-2.1	Web based App	Non-Functional	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 final version of the Visual Data Pre-processing Module has few assumptions and restrictions, which include:

- The REST API is considered updated and enriched with additional endpoints comparing with the first version of the module (endpoints for uploading geometric as-built data, additional filter parameters etc.). However, the communication with the DT Platform and the rest COGITO components will be established and tested next months during the integration process (T8.1 End-to-end ICT System Integration, Testing and Refinement) and therefore the API should be adjusted to arising needs.
- In this final version of the module, no credentials are used to login to the web application. The authentication and authorization of the users will be implemented through Keycloak in the next months, during the System Integration process.
- Thus far, storing a job in the database is organised as follows: a job is related with only one image or only one point cloud for the Visual and Geometric QC respectively. Depending on the structure of the list of the work orders/tasks assigned for Quality Control (provided by WODM), this relation could be modified in one to many (one job, many images or point clouds) during next months. Furthermore, based on the above, additional refinements could be made during the System Integration process, concerning the semi-automated IFC elements' assignment to capturing jobs.
- In this second version of the module, we have focused on handling the geometric as-built data (uploading point clouds). However, updates and refinements concerning the point clouds pre-processing may be carried out during the COGITO System Integration to reflect the user's needs.

3 Conclusions

This deliverable describes the work carried out within T3.5 "Visual Data Pre-processing Module". The tool is a web-based application offering many functionalities to the user and 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 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 semi-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 basic implementation of the main functionalities of the Visual Data Pre-processing Module has already been achieved. In addition, in this deliverable information concerning the development tools, the data exchange and the implemented version of the API are also presented. The achievements so far include the following:

- Addition or selection of a capturing device and its localization within the 3D BIM model
- Addition of a pre-processing job, its capturing parameters, IFC components assignment to the job etc.
- 3D Visualization of the IFC model online
- 2D Visualization of the image data online
- GUI for uploading the as-built data (point clouds and 2D images)
- Implementation of image processing libraries (applying visual filters in 2D raw images)

The work presented here describes the final version of the Visual Data Pre-processing Module. Thus far (M18), the module has been tested only with basic artificial test data. In the future and as COGITO tools evolve, all the necessary updates and refinements will be carried out within T8.1 "End-to-end ICT System Integration, Testing and Refinement" from M18 to M34, to support all the client demands as well as to establish the connectivity and communication with the other COGITO tools. Finally, the Visual Data Pre-processing Module 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://unity.com/.

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

[12] [Online]. Available: https://aryamansharda.medium.com/image-filters-gaussian-blur-eb36db6781b1.

[13] [Online]. Available: https://hackaday.com/2021/07/21/what-exactly-is-a-gaussian-blur/.

[14] [Online]. Available: https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html.

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

[16] D2.5-COGITO, "Deliverable 2.5: COGITO System Architecture v2," 2022.

[17] [Online]. Available: https://user3141592.medium.com/how-to-scale-mysql-42ebd2841fa6.

COGITO

CONSTRUCTION PHASE
DIGITAL TWIN MODEL

cogito-project.eu

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