DENiM Platform

Frontend User Guide

Document Title: DENiM Platform - Frontend User Guide
Author(s): Idener
Project: DENiM – Digital intelligence for collaborative ENergy management in Manufacturing
Dissemination Level: Public

History of Changes

Version	Date	Authors	Changes
1.0	20/08/2024	Idener Team	Initial version

Table of Contents

1 DENiM Platform frontend overview

The DENiM Platform offers a suite of tools for users to develop, implement, and deploy Energy-Centric Digital Twins (**Figure *1***). It combines Software Services (Data Platform, Data Management Service, Platform Management Service), Functional Services, and Energy Consumption Models to develop, implement, and deploy Digital Twins (DTs).

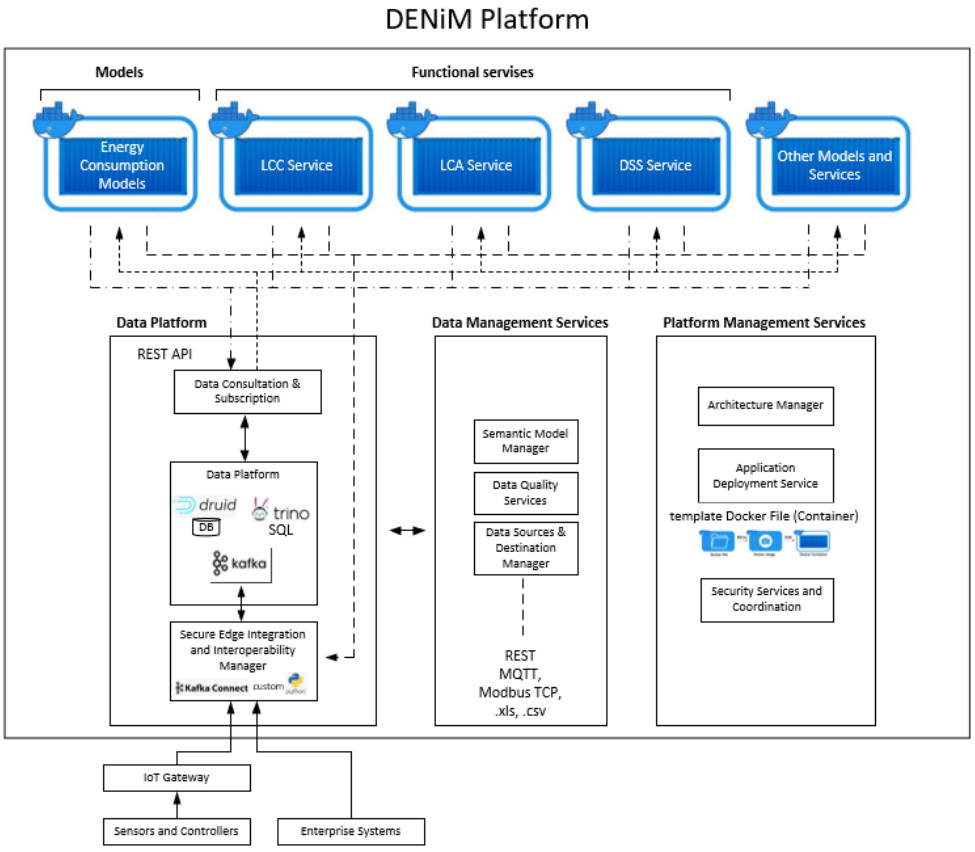

Figure 1. DENiM Platform Architecture

The internal platform services aim to ease the management of data and applications, making it possible to run the DENiM Digital Twin functions. However, this set of tools offers a high level of complexity and requires users to have extensive knowledge. For this reason, a higher level of abstraction is given by a set of graphical interfaces known as the DENiM Platform frontend.

These graphical interfaces are deployed together with the internal platform elements as web interfaces that users can access from their browsers. A test version can be accessed at the following URL:

https://denimplatform-develop.dtenabler.com/

The DENiM Platform frontend encapsulates the use of the most critical components of the DENiM Data and Platform Management Services, as outlined in **Figure *2***. In particular, the services and functionalities accessible are as follows:

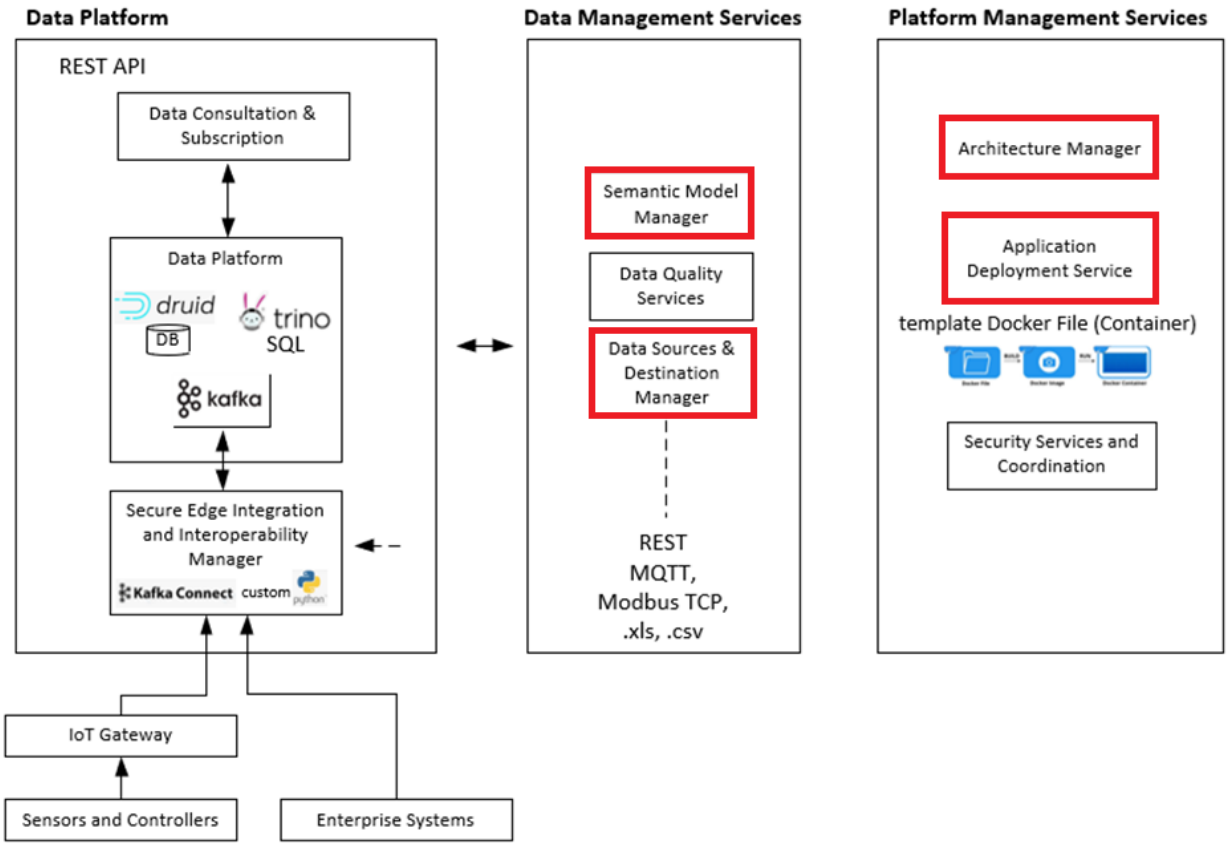

Figure 2. DENiM Platform frontend coverage

1.1 Data Management Services

Data Management Services support the user in integrating their data sources into the platform. They allow for the documentation of the semantic model of the data flows, the connection of the data sources to get data from the real world and the configuration and monitoring of the elements of the data pipeline within the data platform.

1.1.1 Semantic Model Manager

The Semantic Model Manager (SMM) is a tool that allows the definition of the semantic model for a given pilot, use case or scenario. This way, the data sources can be mapped to the semantics and internally, all the data is semantically characterised, ensuring seamless data exchange and interoperability capabilities.

Semantic modelling is supported by the basic concept of Features of Interest (FoI), which represents every individual element of the real world to be tracked in the model (sites, software, sensors, etc.).

The SMM enables semantic querying, for users to be able to interact with data pertaining to elements from the real world without knowing in advance how to reach that data or in which format it is stored. The SMM bridges that gap through the semantic metadata that enriches the data sources and offers the following functionalities:

- Definition of custom types of FoI (ontology management).

- FoI instantiation to describe use case instances.
- Definition of the properties of those FoIs, which map to measurements or fields of the data sources, with all the necessary semantic information (meaning of the variables, units, variable type, frequency, etc). The set of properties belonging to a FoI are grouped in an auto-generated JSON data format to be sent as a single data object to the data platform.

The deployed instances of the semantic models are then stored on a dedicated database and used throughout the data management cycle to fully characterise the acquired and processed data.

1.1.2 Data Sources & Destination Manager

The objective of the Data Sources & Destination Manager is to deploy all the components needed and configure them to establish the complete flow of data from the different sources, which can be either real-time data (i.e., from sensors) or historical data from external databases, to the persistent storage and data serving to end users or applications. To define the flow of data, it is essential to define also the structure of the data. In the DENiM project, an agreement has been achieved about the data schema format that can be used, which consists of a JSON schema with certain format restrictions..

The flow starts with the connection with the data, which can be different communication protocols (e.g., MQTT or Modbus TCP) or data from raw files or existing databases. This process is the most custom-made; however, a support menu generating an example Python code with a Kafka producer has been included to provide a solid starting point for connectors development. A Kafka topic will be created to receive all the information related to this data source. Then, to persist the data, an HDFS directory will be created and a Sink Connector will receive all the information from the Kafka Topic. Finally, to allow access and process of the data, a Druid Supervisor will be created for each data source. The user is provided with the details about the created Kafka topic, Druid table and column names to be able to send data and access it afterwards.

1.2 Platform Management Services

This service allows the platform manager to register and manage endpoints (deployment resources) and deploy the DENiM platform elements (leveraging Docker), both internal architectural elements (the Architecture Manager itself, along with the rest of the platform modules and other necessary elements can be managed through here) and the custom DENiM tools and services to be developed by the end user (Application Deployment Service). It is divided into four different sections:

- Registries/Images: Manages the connection to different registries where the Docker images are located, facilitating efficient image management. This includes creating and deleting images, as well as managing their tags.
- Applications: Create and edit applications. A configurable Docker-Compose provision allows you to specify the services to be deployed, environment variables, and other key elements.
- Servers: Handles the administration of servers, which are an abstract concept to reference the machines, either physical devices or virtual machines, where applications can be deployed. The appropriate software must be running in the machine to be available as a server for the Platform Management Services (more details in the Creating and managing Servers section). The available functionalities include creating and deleting servers and monitoring their main stats (CPU, memory, and disk usage). This allows the user to

dynamically decide when a new application can be deployed onto a certain server or if more resources need to be provided for the platform extension.

- Deployments: Administration and management of deployments, which are the realisation of an application on a certain server. This encompasses creating, deploying, stopping, recreating, and deleting application deployments. It also allows access to containers to view their status and the execution logs.

2 DENiM Platform frontend security

As mentioned before, the DENiM Platform Frontend is presented as a web application that can be accessed from a user's web browser. The specific access depends on the deployment configuration and the need to manage the server IPs, public DNSs, etc.

Access to the graphical interface requires the user to have network visibility with the server where the services are run, and credentials created with the correct authorisation level. The first consideration is also highly dependent on the deployment, as some platform administrators could open the GUI to the public, while others could leave them only accessible from the company's VPN or LAN. The second consideration is also managed by the platform administrators, who should put in place the mechanisms for creating users and role levels for all the workers who need access to the platform.

3 DENiM Platform frontend usage

The DENiM Platform frontend has been created with usability in mind. Therefore, it contains many contextual aids, such as tooltips and integrated help sections. When operating with it, it is recommended to hover over elements and check available "?" signs, as many questions will already be answered this way.

This section provides a detailed guide for operators interested in using the full range of tools available through the DENiM Platform frontend to support users in working with the platform.

3.1 DENiM Platform Workflow Overview

Before describing all the capabilities of the DENiM Platform in-depth, ***Figure 3*** offers a quick overview of its features and presents an overview of a common workflow.

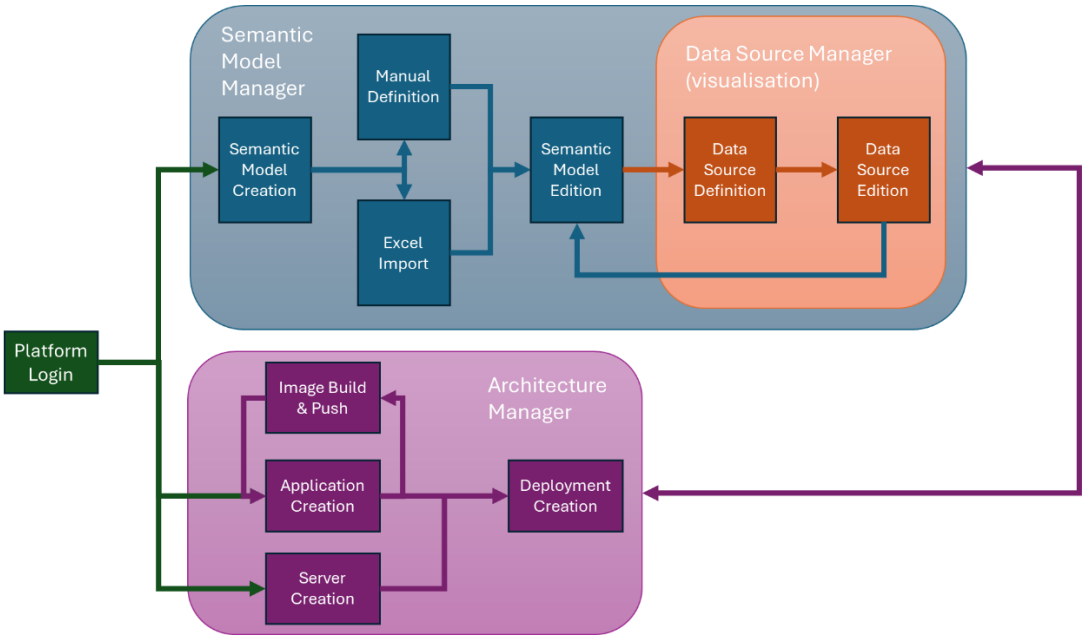

Figure 3. DENiM Platform usage workflow

After logging into the platform, a user can either start working on their semantic model or prepare their software architecture to support said work.

This latest functionality is supported by the Architecture Manager and requires the user to define its Applications, create and push to the DENiM Registry the required Docker Images, and define the available Servers where this application will be run. Once both elements are defined, the user can create Deployments consisting of executing the Applications on a valid Server.

Once the software architecture is complete, the user can start defining their data ingestion. To do so, the Semantic Model Manager is used. First, a Semantic Model representing the problem domain must be created, including defining all its elements (FoIs, Properties, Relations, etc). This can be done either manually or by importing a predefined model from a valid Excel file. The model can be further tuned to the user's needs at any moment.

Once the Semantic Model is defined, the user can connect it to the real-world data by defining Data Sources. This process is also performed from the Semantic Model Manager, while the Data Source Manager allows users to easily visualise all the Data Sources available in the system and their status, among other things.

The process of defining a Data Source is actually automatic, based on the Properties defined in the Semantic Model for a given FoI. So, to edit the Data Sources, users must update their Semantic Model accordingly.

3.2 User registration and login

As mentioned before, a user needs valid credentials to access the Platform. These accounts are to be generated by a DENiM system administrator and configured with the permission levels the user requires.

Once in possession of their assigned user, users can log in to the platform by accessing its main URL and filling in the required fields (**Figure 4**). For their first access to the platform, users must generate their password by using the "Forgot Password?" option. This completes the initial generation of credentials by sending a secure link for password reset into their email.

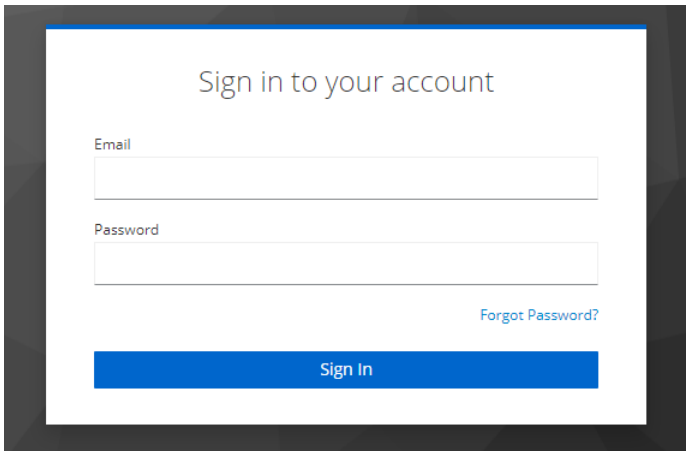

Figure 4. Log in fields

3.3 Main menu

Figure 5 shows the main page of the DENiM Platform frontend. The different sections displayed double as links and brief summations of the Platform's status:

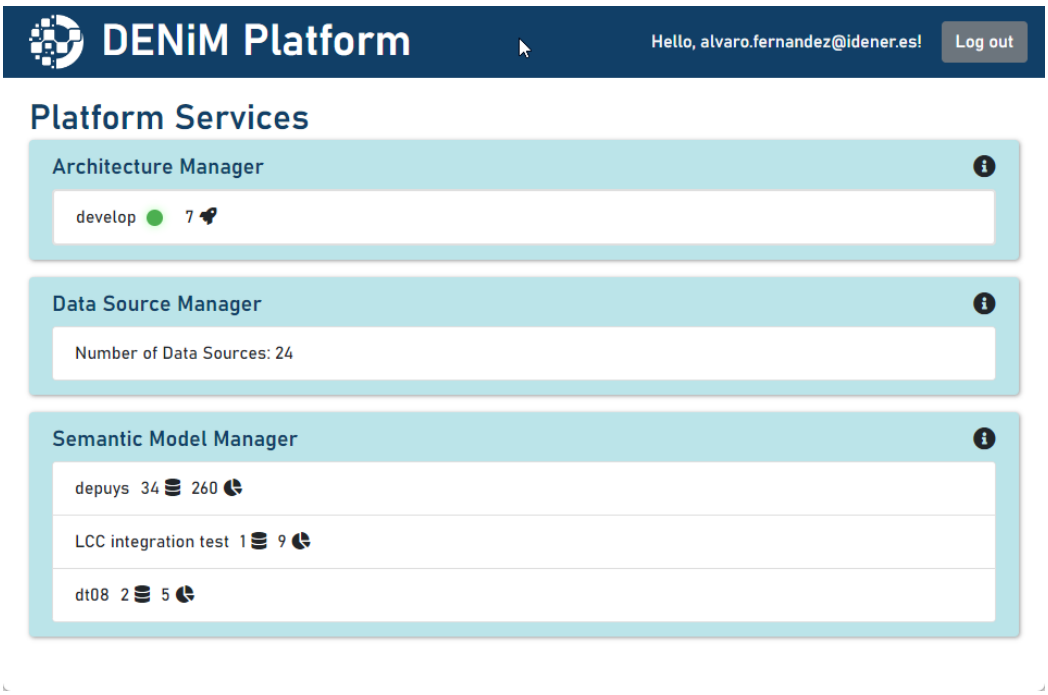

Figure 5. DENiM Platform frontend main page

- Architecture Manager:
  - Links to the Platform Management Services

- Displays the Servers currently connected, their status, and the number of Deployments contained within each.

- Data Source Manager:
  - Links to the Data Sources & Destination Manager
  - Displays the current number of Data Sources.
- Semantic Model Manager
  - Links to the Semantic Model Manager
  - Displays the current existing Data Spaces and the number of FoIs and Properties defined within each one.

3.4 Architecture Manager

The Architecture Manager implements the capabilities to manage the Applications that compose the DENiM architecture and the DTs, the Servers where Applications can be run, and the Deployments of said Applications inside the defined Servers.

3.4.1 Creating and managing Applications

Figure 6 highlights the elements of the Architecture Manager view that should be used when creating or editing Applications for use inside the DENiM Platform.

The Applications area (1) shows the complete list of currently available applications whose details can be checked or modified by clicking on each one (2). The "+" symbol in this area (3) allows the user to create new Applications. Finally, the Images button (4) allows access to the Registry Management and Image Creation view.

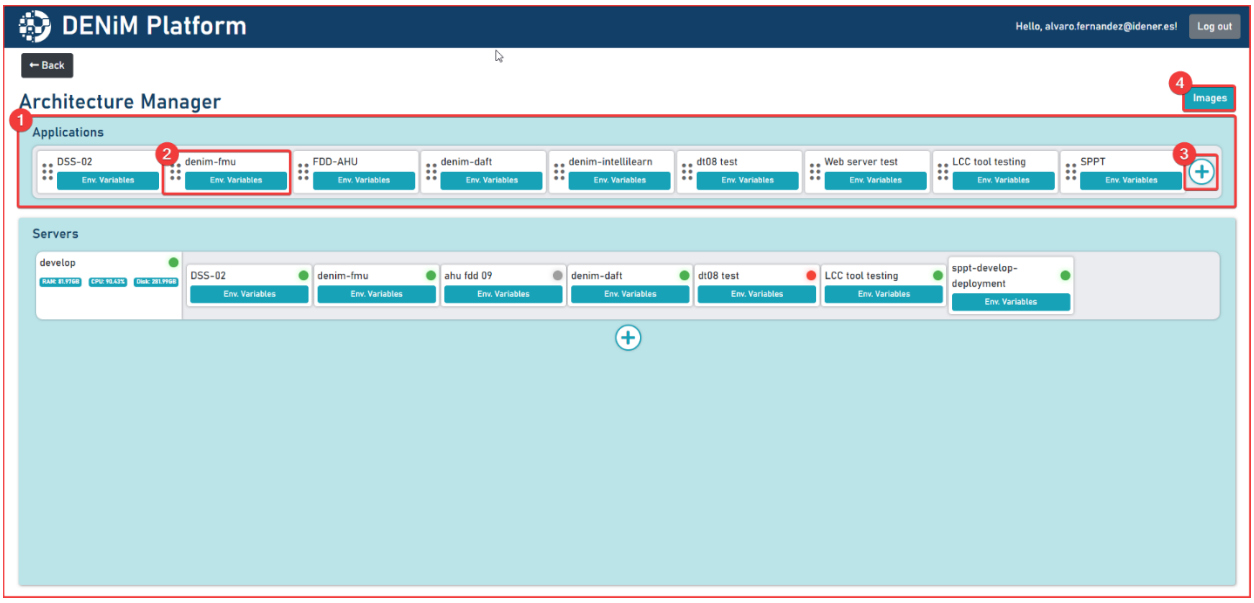

Figure 6: Architecture Manager elements related to Applications

3.4.1.1 Managing Registries and Images

Users may need to create new Docker images to add a certain service to their Application's Docker Compose. They can do so by accessing the Registries and Images view by clicking on the "Images" button (4).

The Registries and Images view (**Figure 7**) allows users to access Docker Images stored in different types of registries, including the default DENiM Registry, and use them to build their applications. The default DENiM registry also allows users to build new images, store them, and delete existing images.

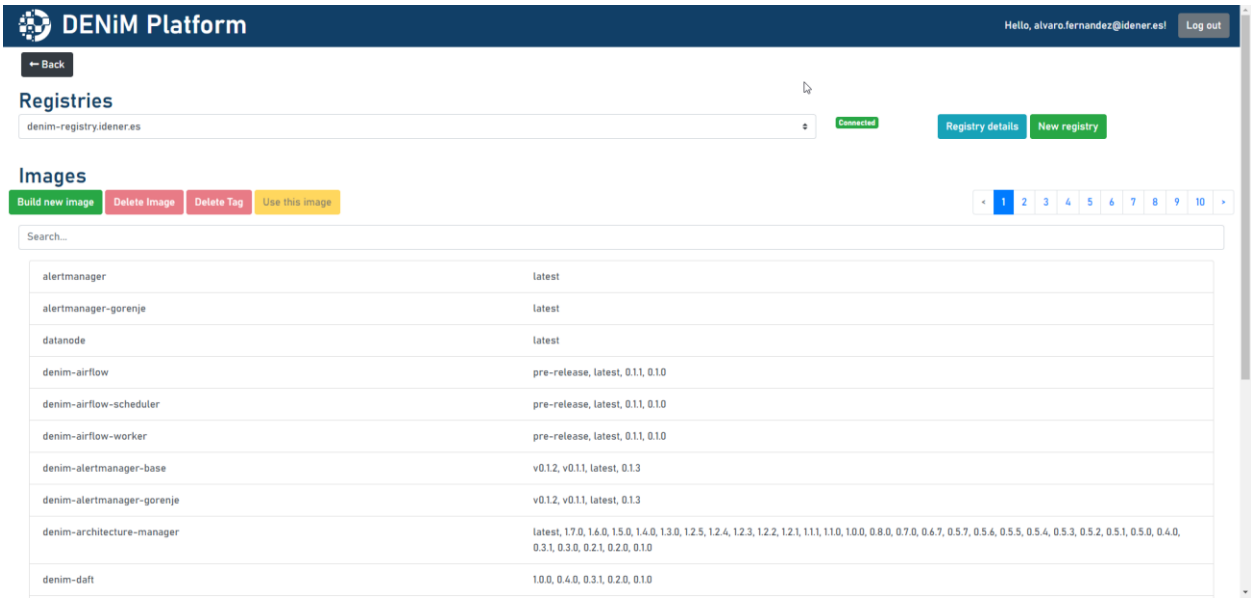

Figure 7. Registries & Images View

The selector available at the top of the page allows cycling through the available registries, while the Images section below will be updated with the Images available, listing the existing tags.

To add a new Registry, click the "New Registry" button and the Registry Creation Modal (**Figure 8**) will be displayed. Supported registries are DockerHub, GitLab Registry, and custom registries compatible with the Docker Registry API (normally self-managed registries).

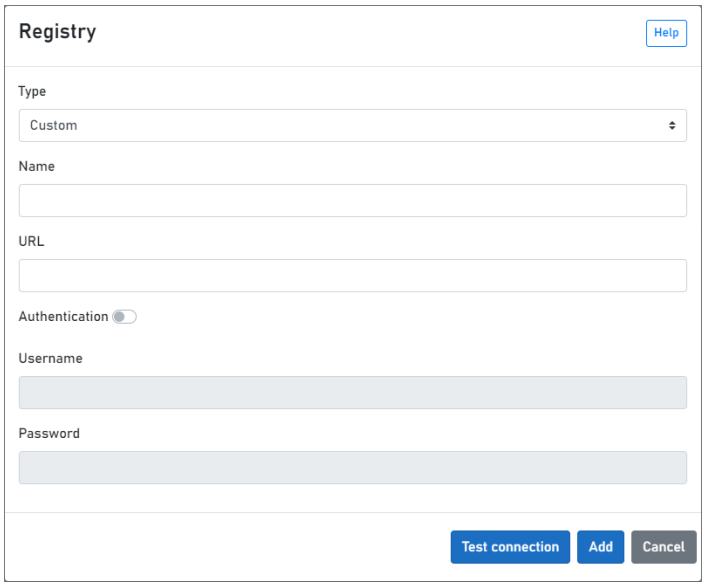

Figure 8. Registry creation modal

Fill in the necessary information in the modal, including security credentials, in case the registry requires them. Note that the modal will display differently according to the selected Type. More information is available by clicking the "Help" button.

Before creating the registry, it is recommended that the user tests the connection with the built-in "Test connection" button. Failure to connect will prevent the registry from being added.

Existing registries can be updated or deleted by clicking the "Registry Details" button, displaying their associated information (**Figure 9**). If changes are made to them, click the "Update connection" button to test and apply. Note that the default DENiM Registry cannot be deleted.

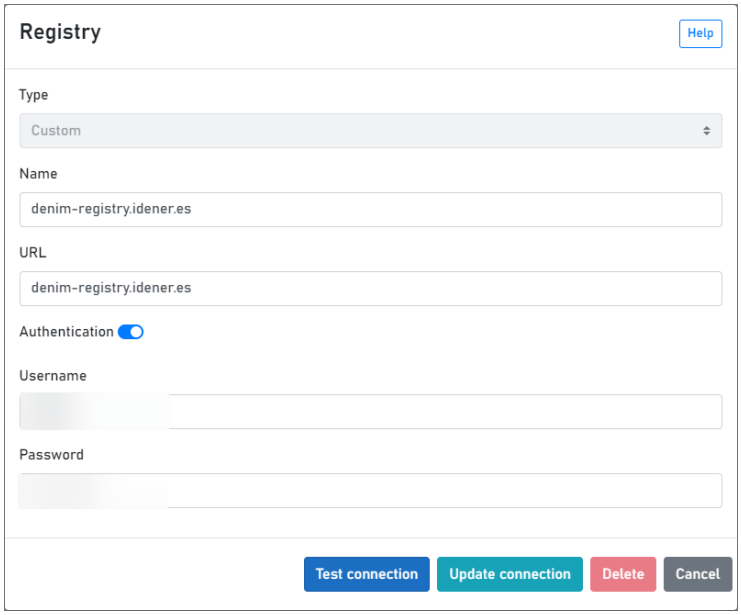

Figure 9. Registry Details modal

Regarding the use of Images, selecting one from the registry list will enable their related options (**Figure *10***). The "Use this image" button will provide the path for each available tag of the image so it can be used in Applications (**Figure *11***). The options to Delete images or specific tags and to build and store a new image will only be available for the default DENiM registry. For safety reasons, the AM is not usable for image management on other registries, but just to be able to consult and instantiate them.

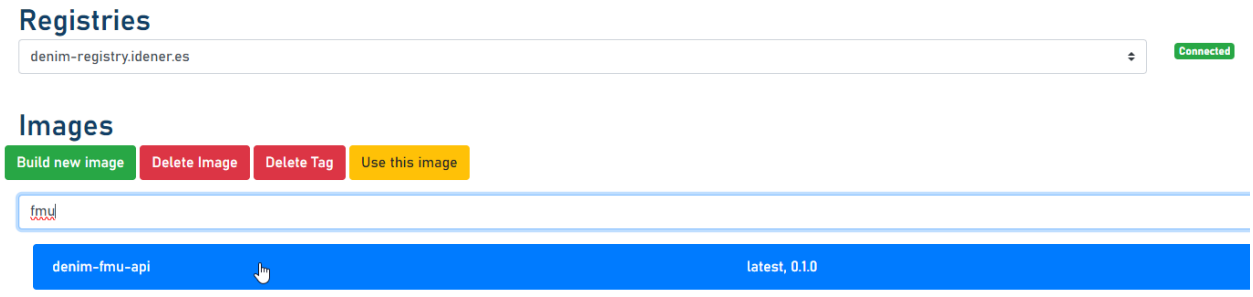

Figure 10. Available actions over a Docker image

Figure 11. Image usage information

To build an image, the user must provide either a valid Dockerfile or a complete tarball containing all the files required, including also a Dockerfile, in the Build Image Modal (**Figure *12***). The image will be built and pushed to the DENiM Default Registry from here. Once this process is complete, the image will be available in this registry image list.

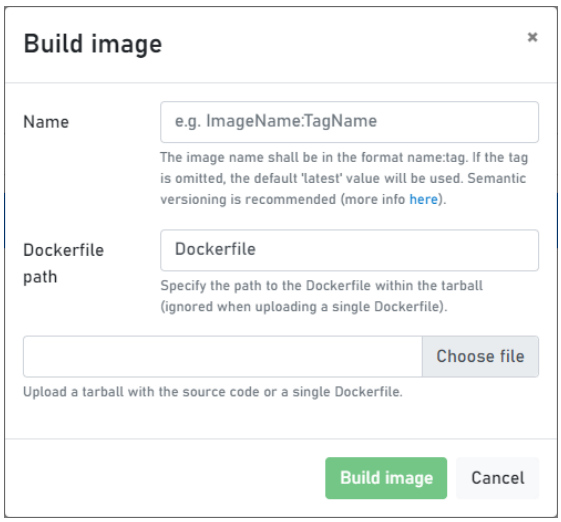

Figure 12. Build Image Modal

When building and pushing images to the DENiM Default registry, the following points must be taken into account:

- When using a simple Dockerfile as input (no additional source code files needed), the Dockerfile path will be ignored. This field indicates the path to the Dockerfile within the inputted compressed folder.
- Dockerfile files, either individually or in conjunction with other files, must be verified beforehand on the user's local machine. Errors during the building process will not be reported back to the Platform or the user. It is advised to make a test build on their own machine to correct any possible errors beforehand.
    - Tarball validation can be achieved locally by doing using the following command with the generated tarfile as input:

```
docker build -f /path/to/Dockerfile - < yout-tarball.tar
```

- Tarballs for image building must be correctly generated and compressed with one of the following algorithms: identity (no compression), gzip, bzip2 or xz.

- Tools such as 7zip on Windows or the *tar* command on Linux CLI are recommended for compression. Example command using *tar:*

```
tar -cvf your-tarball.tar /path/to/your/files
```

- Depending on the image size, building and pushing it to the Registry can take a long time. Allow a suitable wait time for the image to appear in the list. A success pop-up message will be shown when the build process has finished, after which users must reload the page to display the updated list, containing their new image. If something goes wrong, it is suggested to double-check the input files and values, validate locally and try again.

3.4.1.2 Working with Applications

The complete process of creating an Application is as follows:

1- Click the "+" button (3). The Application Creation Modal (**Figure *13***) will be shown.

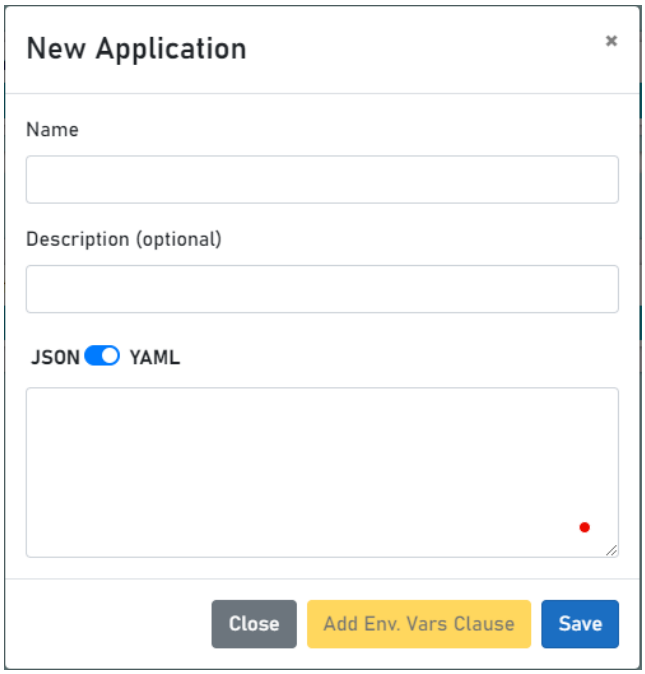

Figure 13. Application creation modal

2- Fill in the Application information, including a Name (required) and Description (optional). The JSON/YAML field must contain the Docker Compose that defines the Application and can be edited using both syntaxes by changing the selector. The field will automatically validate its contents and not allow proceeding if errors are detected. Note that, despite performing the JSON/YAML syntax validation, that doesn't ensure a valid docker-compose configuration. Users are encouraged to execute and test their compose files in a similar environment before jumping to deploying through AM, working with already validated configurations.

3- If the Application requires any specific Environment Variables, click the "Add Env. Vars Clause" button, which will be enabled once the Docker Compose has been filled and validated. All the containers in the application will have access to the Environment Variables. The environment variables are configured from a dedicated view (**Figure *14***), which is

accessed at a later stage of the process. When defined at the Application level, Environment Variables act as a template for the needed variables and default values, but they can be customised in each specific deployment (which they normally are), overriding their contents according to the specific needs of each Server and specific Deployment configuration.

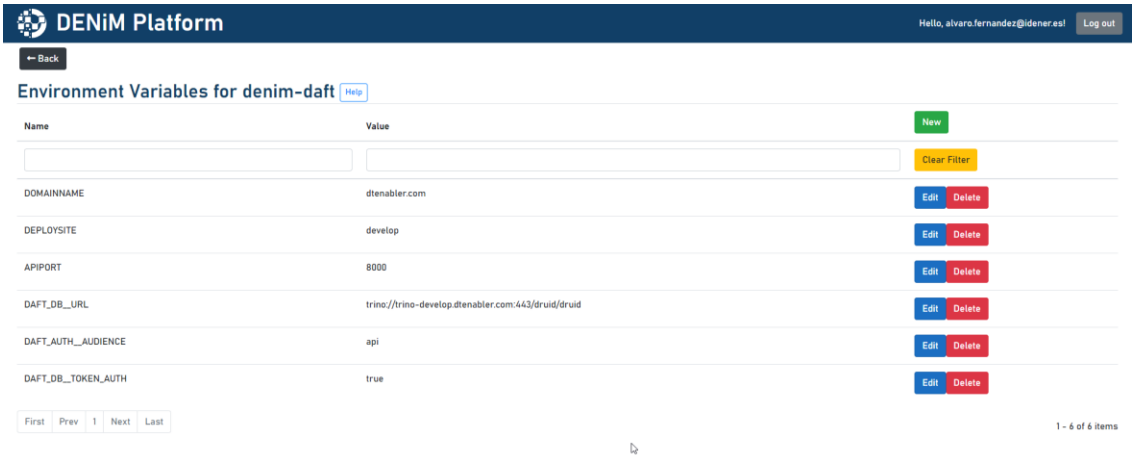

Figure 14. Environment Variables view

4- Once everything has been filled in, click the "Save" button to create the Application, which will be added to the Application area (2).

In case a later edit to the Applications is needed, the user can do so by clicking on the relevant Application (2) inside the Applications Area (1), which will display an already filled-out Applications Modal (**Figure *15***). From here, the Application may be edited or even deleted.

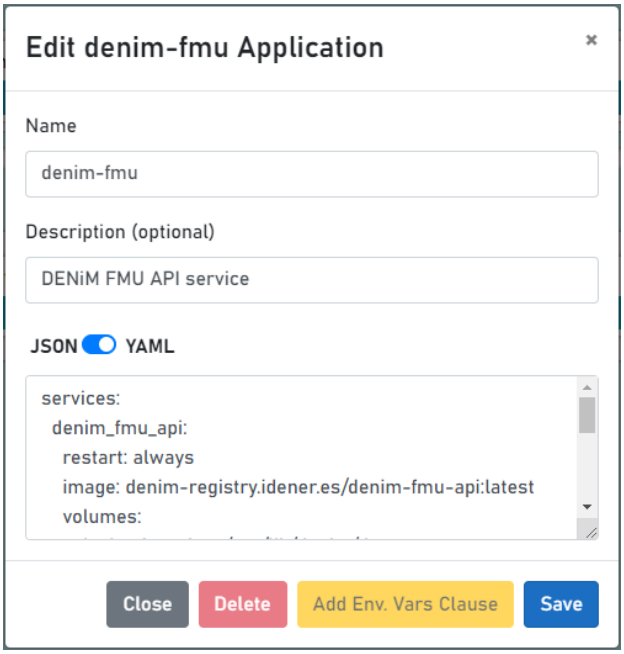

Figure 15. Application Modal for editing

Note that once the Application is created, the user must use the dedicated button available for each Application in the Application area (***Figure 16***) to access the Environment Variables view (**Figure *14***). Follow the instructions in the "Help" button to learn how to define Environment Variables common to all Docker Compose services in the Application or Variables specific to each one.

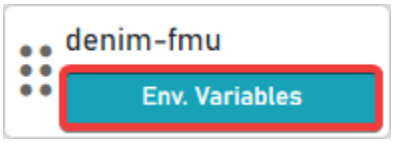

Figure 16. Application dedicated Env. Variables button

3.4.2 Creating and managing Servers

To host and execute the Applications, the user needs to define Servers, machines or domains where said Applications will be deployed.

For a server to be available in the AM it must fulfil the following requirements:

- To be reachable and available through the network
- To execute an instance of the platform's endpoint agent software, which will allow the AM to interact with it and perform adequate functionalities.

The AM will always make its own local machine available as a server, auto-configuring everything for the user. Users must perform the configuration themselves to use any additional machines (physical or virtual).

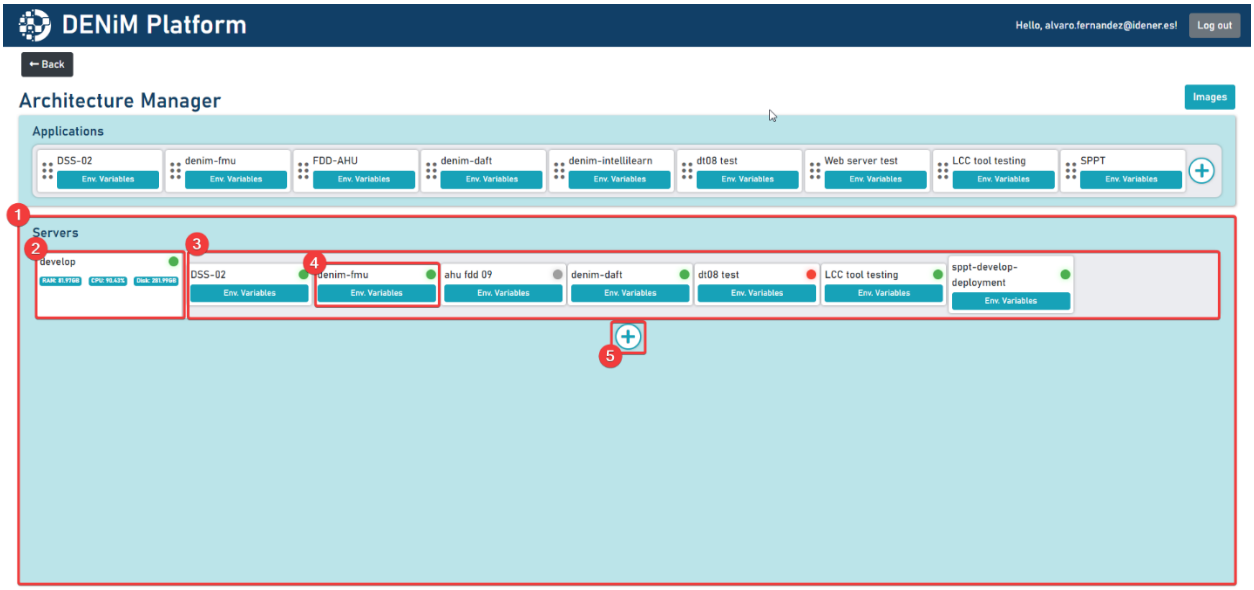

Figure 17. Architecture Manager elements related to Servers

Figure 17 outlines the elements involved in operating with Servers inside the DENiM Platform frontend. The Servers Area of the Architecture Manager View (1) contains all the available Servers (2) on the left side of the area. Related to each one, the Deployment Zone (3) shows all the Applications

(4) currently deployed inside said Server. Finally, the "+" symbol in this area (5) allows the creation of more Servers.

The process of creating a Server is as follows:

1- Click the "+" button (5) in the Servers Area (1). The Server Create Modal will be displayed.

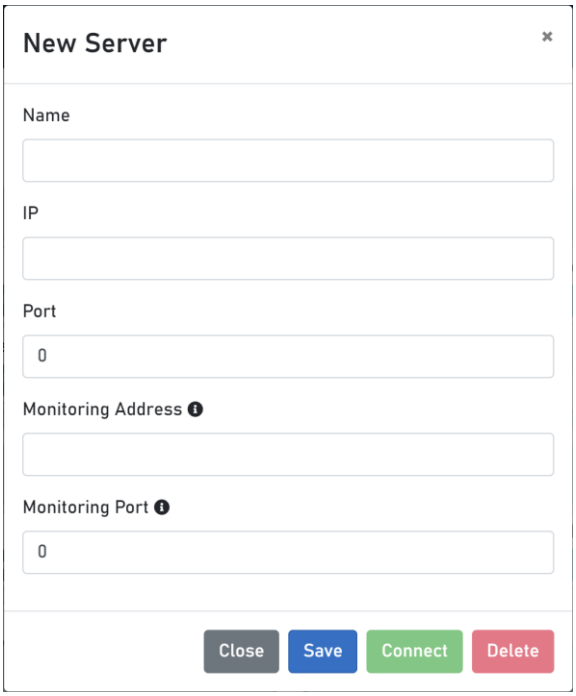

Figure 18. Server Create Modal

2- Fill in the information for the server, including its IP address and Port. If available, the Monitoring Adress and Port fields are used to retrieve metrics regarding Server performance from a Prometheus API running on it. The endpoint agent software mentioned above (managed by platform admins) also enables this remote monitoring API.
3- Click the Save button to create the Server. It will appear as a new entry on the Servers list (2) alongside its own Deployment Area (3). This entry will display a status colour (see **Figure *19***), which initially will be grey, indicating an idle/not yet connected to the server status.
4- The server can be edited anytime by clicking on it. Click the "Connect" button inside the Modal to connect to the server. The corresponding entry in the Servers list (2) will update its

status colour, with green indicating a successful connection and red when a connection error occurs. See **Figure *19*** for details.

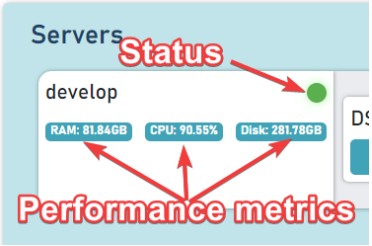

Figure 19. Server brief info overview

3.4.3 Creating and managing Deployments

A deployment represents each of the specific instantiation of an application within a certain server. The same application can have multiple deployments with separate configurations (through env. vars customisation). For instance, one would want to create multiple instances of a data connector, each of them gathering data from a specific sensor and pushing into the platform.

Once a Server is correctly created and connected, it can be used to host deployments. To create a Deployment, drag an Application from the Applications list into the Deployment Area of the desired Server. The Deployment Modal will be displayed (**Figure 20**). Fill in the Name field and click on the "Save" button.

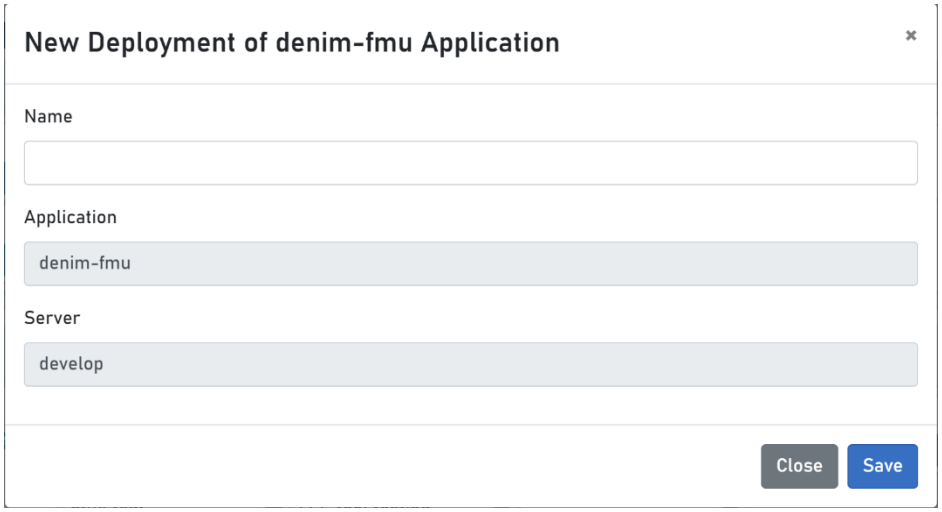

Figure 20. Deployment creation modal

The desired Deployment will be created but not automatically deployed or started. The Deployments Status indicator, which uses the following colour code, will signal this (**Figure *21***):

- Gray: Not deployed
- Green: Deployed and running
- Red: Deployed and stopped (might be due to an error or simply to the service finishing if it is not supposed to be made of persistent containers, e.g., initialisation containers).

Figure 21. Pending deployment status

To deploy the application, click on the Deployment to display its Edit Deployment modal, containing the following controls:

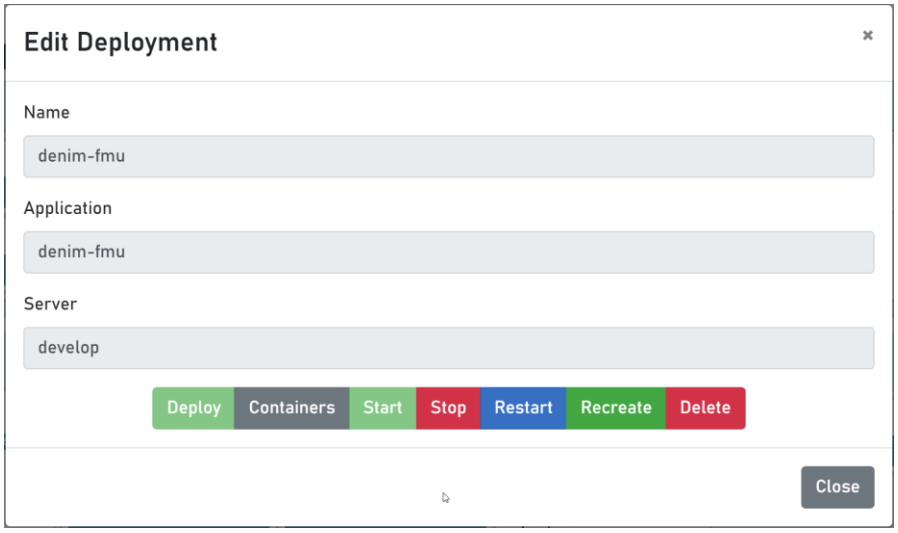

Figure 22. Edit Deployment modal

1- Deploy: Perform the deployment. This operation is done asynchronously so the user can perform other tasks in the meantime. A notification will be shown when the deployment either succeeds or fails, although if the user moves to a different view in the Platform, the notification won't appear.
2- Containers: See the Docker Containers list for this Deployment (**Figure 23**), their status and execution logs

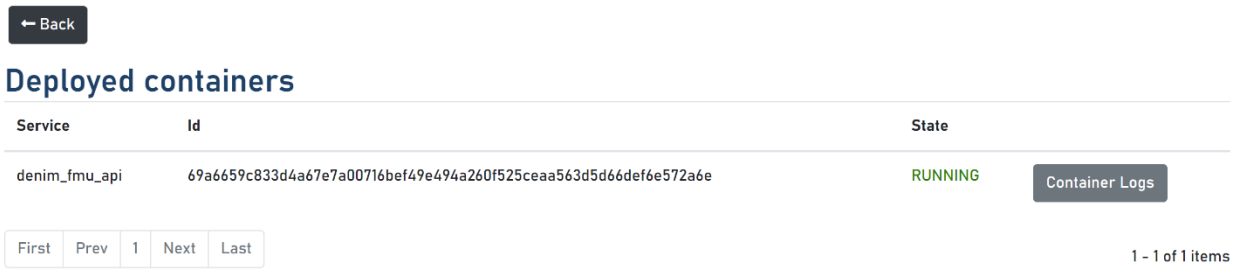

Figure 23. Containers list for a Deployment

3- Start, Stop, Restart: Control the application execution.
4- Recreate: This restarts the Deployment, applying any changes that may have been made to its related Application. Note that if changes are made to an Application's Docker Compose, simply Restarting its associated Deployments won't apply those changes; the Deployment will need to be recreated. During this operation, users will have the option of pulling the latest available version of the docker images, in case changes have been made to them; if not

indicated, the versions already available will be used. In a similar manner, modifications to the Environment Variables also won't take effect until the Deployment is recreated.

5- Delete: Delete the Deployment from the Server. This will also delete the containers created.

3.5 Data Source Manager

The Data Source Manager presents the capabilities to explore the existing Data Sources, connecting real-world sources of information to Semantic entities.

DISCLAIMER: The Data Source Manager is limited to read-only functionalities. The Semantic Model Manager allows complete control of the Data Sources.

3.5.1 Exploring existing Data Sources

Figure *24* shows the Data Source Manager View, which presents a list of all the Data Sources available in the system, with their related Features of Interest, Kafka topic and Druid Table information.

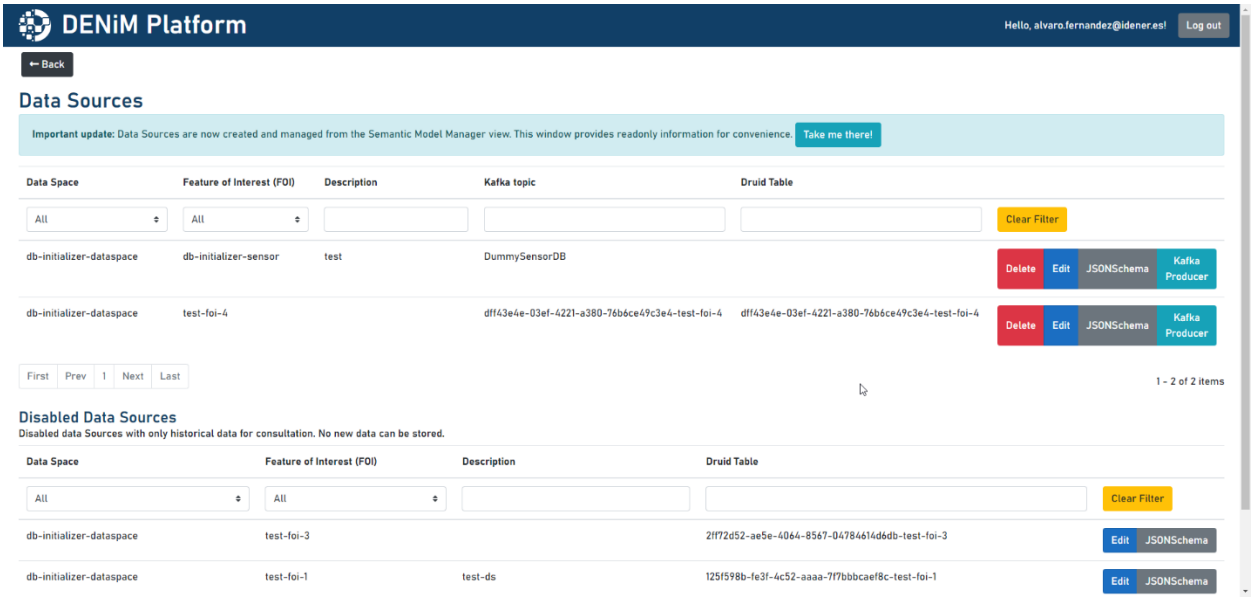

Figure 24. Data Source Manager view

Also, this view will also display if the system contains any Obsolete Data Sources. These are Data Sources that have ingested data but have since been deleted from their related FoI and are preserved as Disabled to prevent data loss. More information about this is given in the Deleting Data Sources section.

The Edit button only allows changing the Description field, while the JSON Schema and Kafka Producer buttons display modals with the complete fields for the selected Data Source, which work as follows:

- JSON Schema: it represents the format that the DENiM Data platform is expecting to correctly process incoming messages for this specific Data Source. This field is read-only and autogenerated from the FoI properties. However, it can be modified by changing said

properties from the Semantic Model Manager (see Creating and managing Features of Interest, below) and refreshing the Data Source.

- Kafka Producer: provides sample Python code to generate and send data using this specific Topic and Schema. Users are encouraged to use this code as a basis to experiment and test in environments such as Jupyter. To this end, convenience options for copying its contents to the clipboard or downloading it as a Python script (**Figure 25**) are offered.

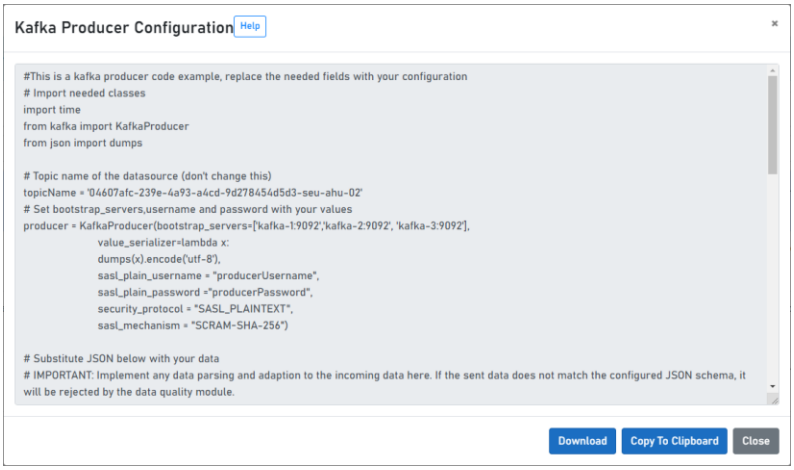

Figure 25. Read-only Kafka Producer info

3.6 Semantic Model Manager

The Semantic Model Manager allows users to create and manage semantic entities, define their properties and the relationships between them, and connect them to specific data sources to consume data coming from real-life sensors. This facilitates the creation of a Digital Twin of the industrial process. All these are organised inside independent Data Spaces, ensuring that different use cases, sites and areas can be managed perfectly inside the same instance of the DENiM Platform. The separation in Data Spaces also allows future data governance within the industrial site, by enabling the data-space-level separation of users' access and permission. This is a future implementation plan at the moment.

3.6.1 Base Ontology Management

The SMM defines an underlying, semi-fixed ontology that constitutes the base for defining all the semantic models across the DENiM Data Platform. It determines the available values for FoI Types, FoI Property Types and FoI Relation Types.

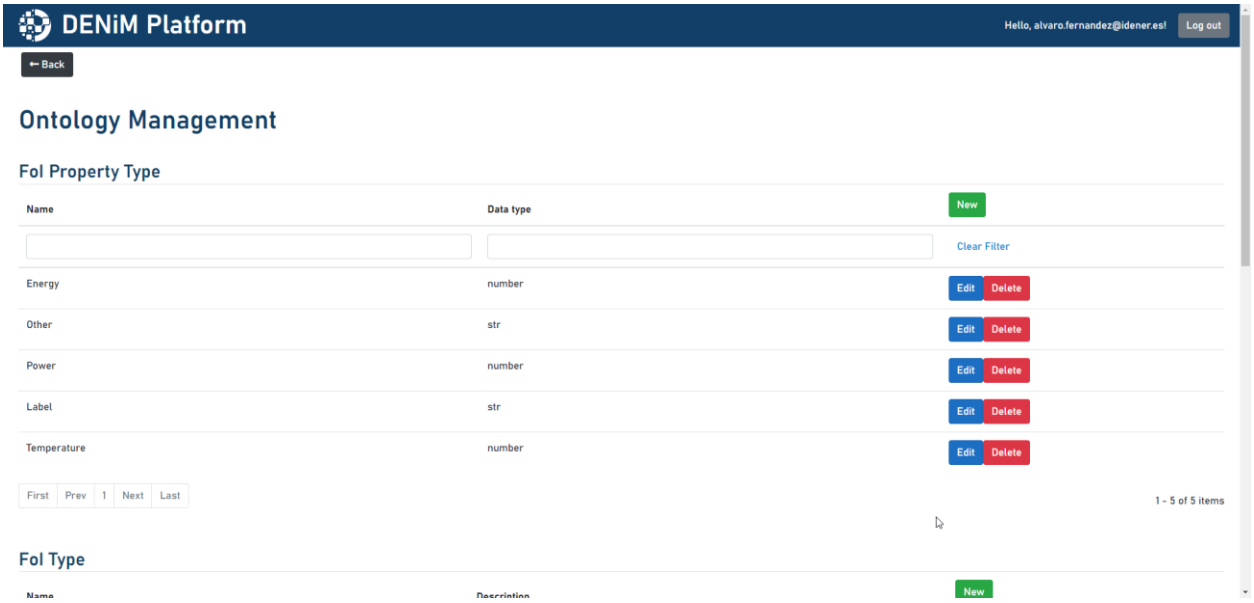

Figure 26. Ontology Management View

It is very important to note that the ontology should not be modified freely but only when it is identified that something cannot be represented with the existing types. DENiM pretends to use a single ontology derived from the standard SAREF and supported by Project Haystack, but minor additions in case of need are accepted as soon as they are consensual and done consistently at a project level. Therefore, only platform admins can perform the ontology edition at the moment. **Figure 27** highlights the Ontology Management button, which gives access to this view. It will only be displayed for admin users.

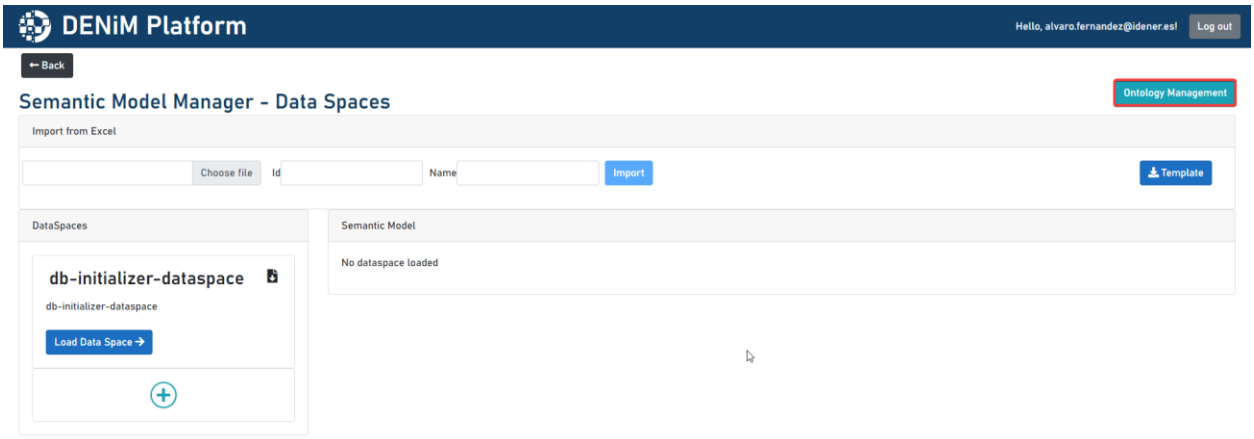

Figure 27. The Ontology Management button is only available to platform admins.

3.6.2 Creating and managing Data Spaces

When entering the Semantic Model Manager main view (**Figure *28***), the user is presented with a list of available Data Spaces (3), which can be loaded by pressing the Load Data Space button (4). This will display all available FoIs in this Data Space in the Semantic Model section (6).

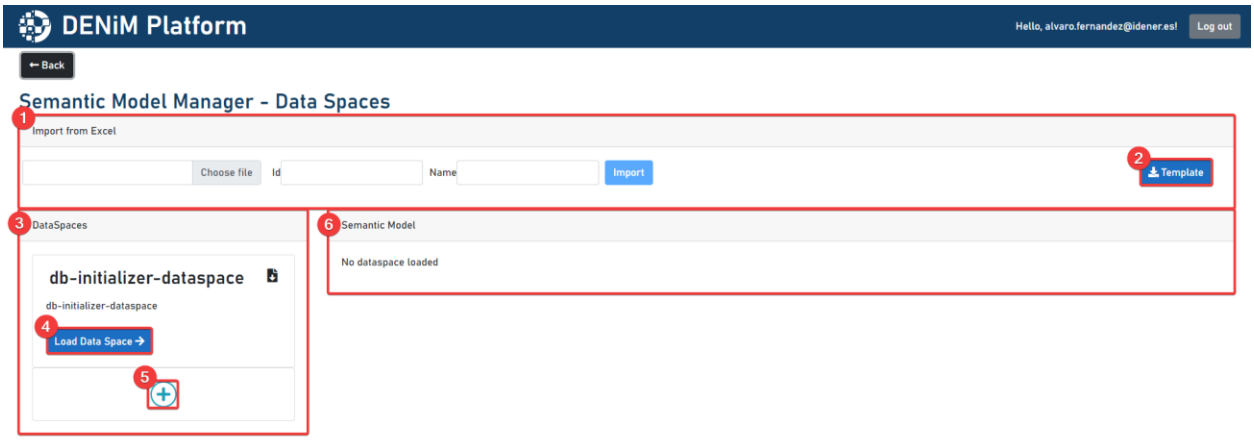

Figure 28. Semantic Model Manager main view

To create a new Data Space, there are two possible ways:

1- Use the Import Excel file area (1) to import an Excel file describing the complete semantic model, including FoIs, properties, and relationships). Users can download a template file using the Template button (2).

2- Add a new empty Data Space directly by pressing the "+" button (5) in the Data Spaces area. This will display a form to specify the basic information of the Data Space (**Figure *29***). Press the Create button to finish the process.

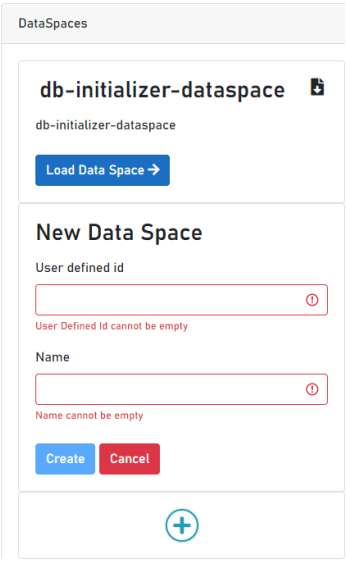

Figure 29. Create Data Space form

For convenience, an option to download a complete Data Space as an Excel file is offered. Press the download button next to each Data Space Id (highlighted in **Figure *30***).

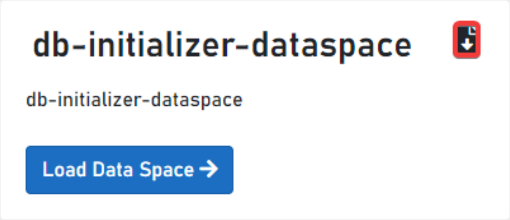

Figure 30. Download Data Space Excel

A typical workflow would be to download the semantic model as an Excel, do a bulk edition with all the necessary changes, and re-import into a new semantic model, cleaning the old one after validation. Once the data space is complete and data sources are created, this workflow is not possible anymore, as the links to the data sources would be broken. Therefore, users are suggested to manually edit the data space within the GUI when it cannot be removed.

3.6.3 Creating and managing Features of Interest

Features of Interest (FoIs) are the basic components of the Semantic Model, translating each of the real-life elements of the industrial environment to a semantic entity capable of receiving data from Data Sources.

To create a new FoI, a Data Space must be loaded first, following the instructions provided in the previous section. This will populate the Semantic Model section with the current Data Space's FoIs (**Figure *31***).

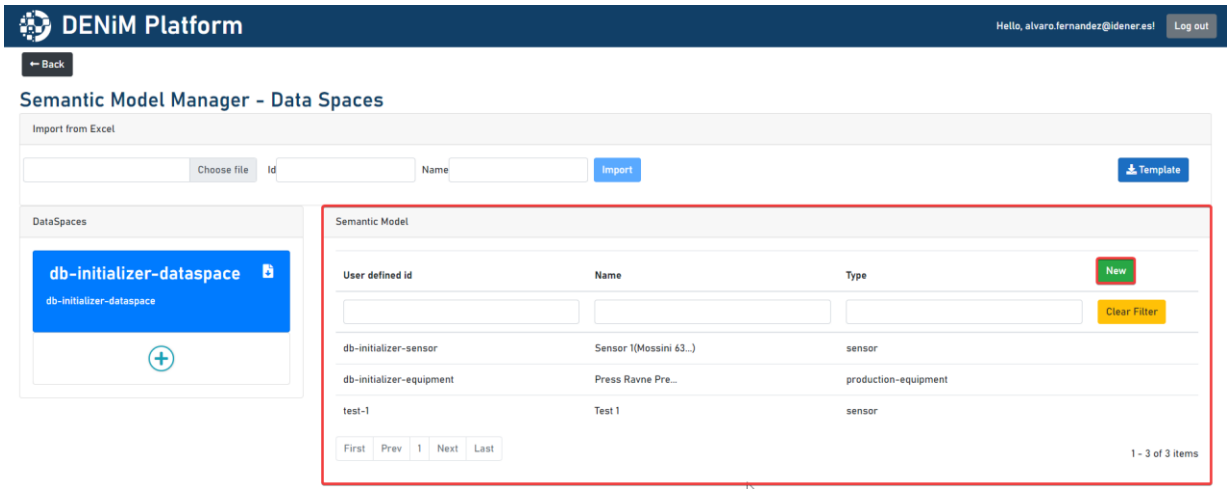

Figure 31. Populated FoIs area and New button

Having loaded the desired Data Space, press the New button to display the FoI Creation Modal (**Figure *32***). Once filled, press the Create button, and the Data Space will be refreshed to include the newly created FoI. Note that the available FoI types to classify the Features are pre-defined by the internal ontology (**Figure *33***).

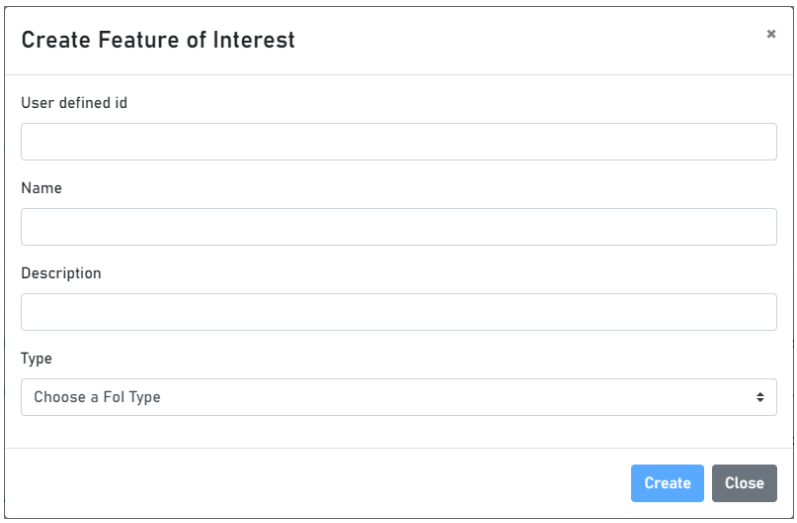

Figure 32. Create FoI modal

Figure 33. Available FoI Types

Once created, double-clicking the FoI entry in the Semantic Model list will display its Edit modal (**Figure *34***). Here, further functionalities are available, which will be described in the following sections:

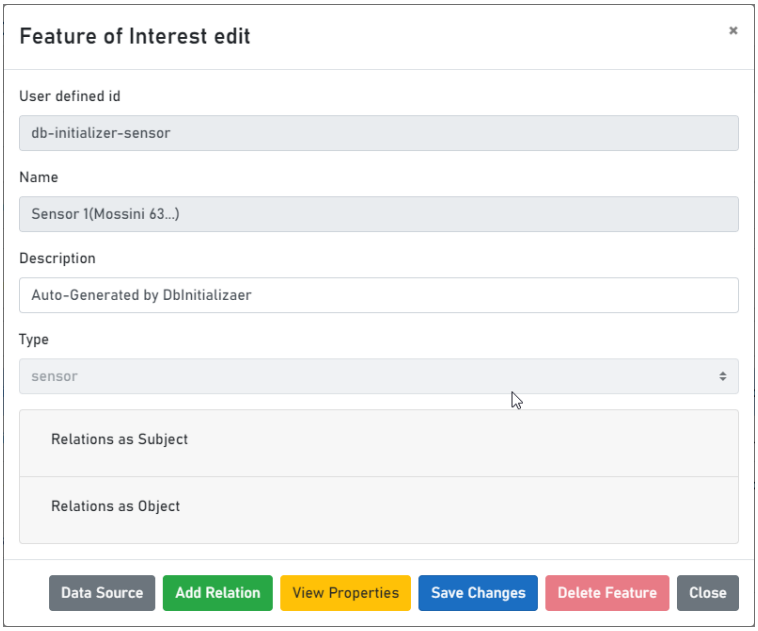

Figure 34. FoI Edit modal

3.6.3.1 General considerations for editing FoIs

The following conditions apply when editing FoIs:

- Once created, a FoI's ID and Name are immutable. Only its description may be edited.
- A FoI can be deleted from the Data Space if it's not connected to a Data Source. Once connected, deletion won't be allowed to prevent data loss.

3.6.3.2 Managing FoI Properties

Clicking the View Properties button will take the user to the dedicated Properties page for that FoI (***Figure 35***). A list of existing Properties will be displayed, detailing the information for each one at a glance.

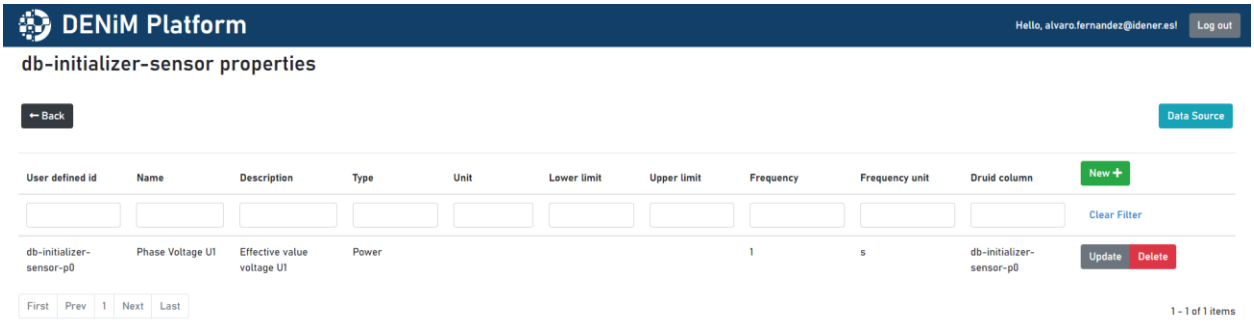

Figure 35. FoI Properties view

Press the New button to display the Property Creation Modal (**Figure 36**) to create a new Property. Fill in the information required, including the Property and Variable Types, which again are limited to the internal ontology options (**Figure 37**).

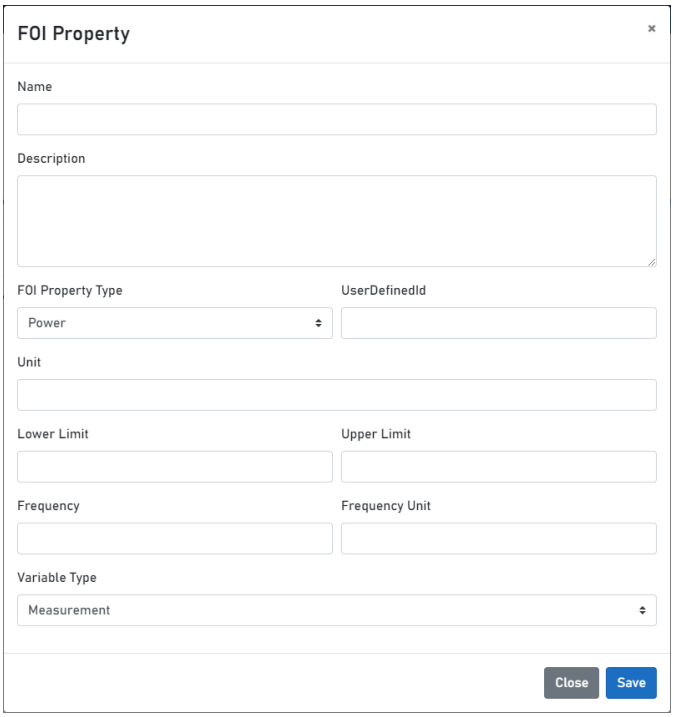

Figure 36. Property creation modal

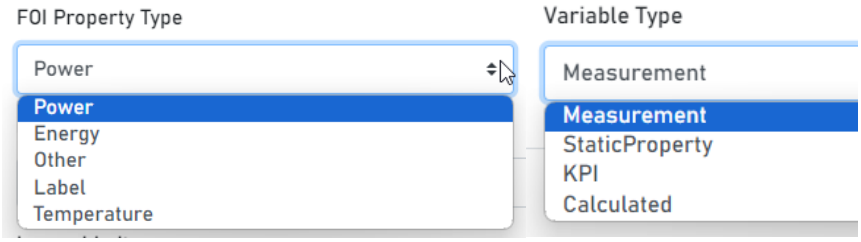

Figure 37. FoI Property and Variable type options

3.6.3.3 *Managing FoI Relations*

Relations between Features of Interest serve to translate the compositions of the industrial environment under study to the semantic model representing it. FoIs can either be the Subject or the Object of a relationship, indicating how they relate to each other.

Information about what Relations include a certain FoI is quickly accessible from that FoI's Edit modal (**Figure *34***). The sections are folded by default, so to display the information, simply click on the "Relations as Subject" and "Relations as Object" sections inside the modal (**Figure *38***).

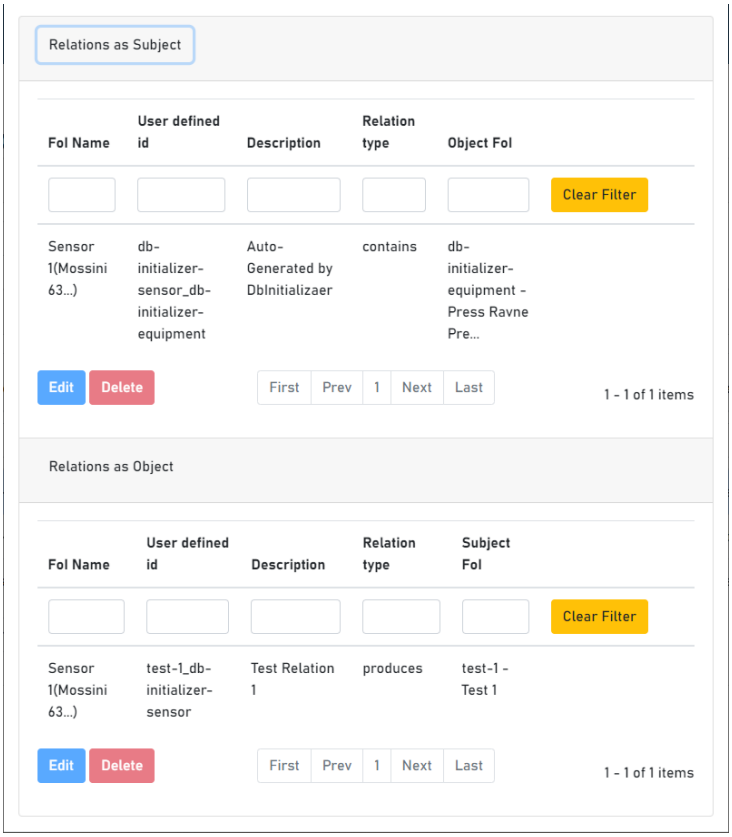

Figure 38. Relations information inside a FoI Edit modal

Relations must be created from the FoI that will act as Subject in it. To create a Relation, click the Add Relation button inside the FOI's Edit modal (**Figure *34***) to display the Relation Create modal (**Figure *39***).

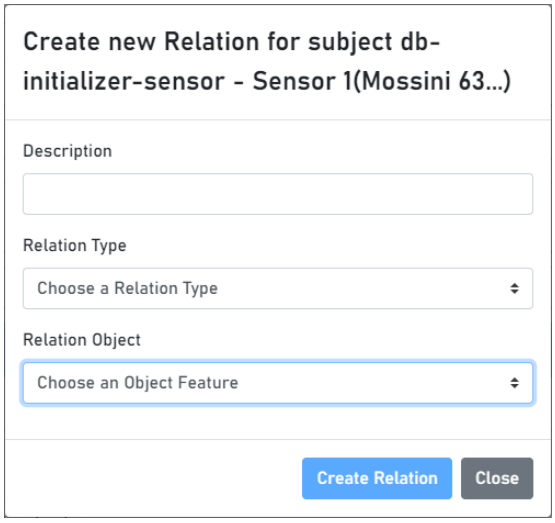

Figure 39. Create Relation modal

Tip: When creating a Relation between two FoIs, consider the Relation Type as the verb that connects the subject and the object in a "phrase", providing semantic meaning to their connection, e. g.: "A contains B", with 'A' being the subject, and 'B' being the object.

Once again, the available Relation Types are determined by the internal ontology (**Figure *40***). The Relation Object list will be automatically populated with the available FoIs in the current Data Space.

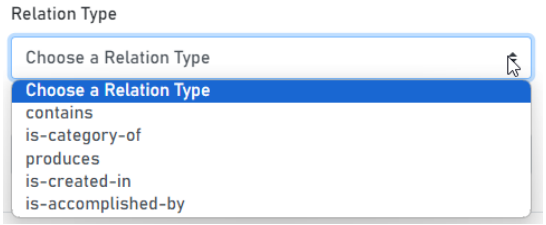

Figure 40. Available Relation types

Existing Relations can be edited or deleted from either their Subject or Object Fois' Edit modal (***Figure 38***). Select the desired Relation from the list and press the Edit or Delete button (**Figure *41***).

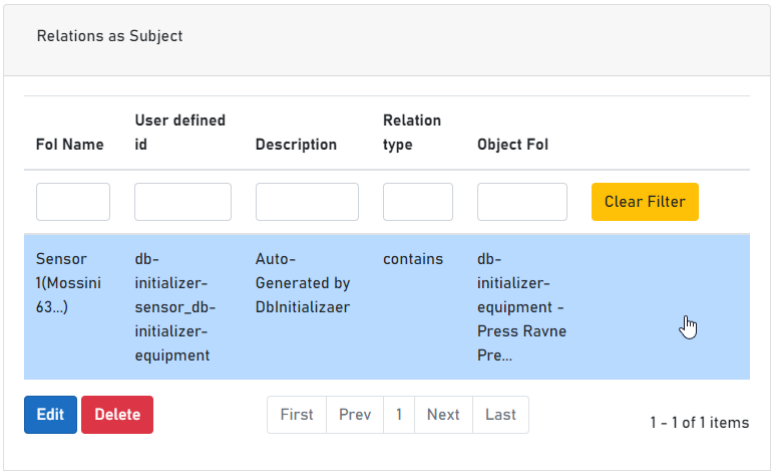

Figure 41. Edit or Delete Relations

3.6.3.4 Connecting Data Sources

A Data Source provides information coming in real-time from sensors or other sources inside the industrial environment. The Data is circulated through the internal elements of the DENiM Data Platform, providing the semantic entities with real-world information.

Press the Data Source button in any FoI's Edit modal (**Figure 34**) to start the process inside the Data Source Creation modal (**Figure 42**).

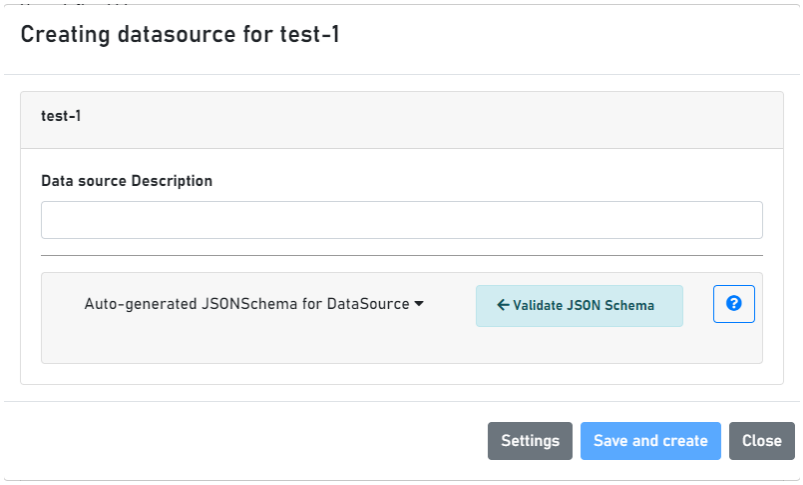

Figure 42. Create Data Source modal

A pre-defined JSON syntax specifies a FoI and a Data Source connection. The DENiM Platform will auto-generate the necessary JSON Schema for the selected FoI based on all its specified Properties (**Figure *43***), so the user only needs to validate it or modify the FoI. Once the FoI is correctly set, the JSON Schema can be created (clicking in the "Auto-Generated JSONSchema for DataSource" Section) and saved (by clicking the "Save and create button") to establish the connection. See additional details over the JSON Schema in the Exploring existing Data Sources section.

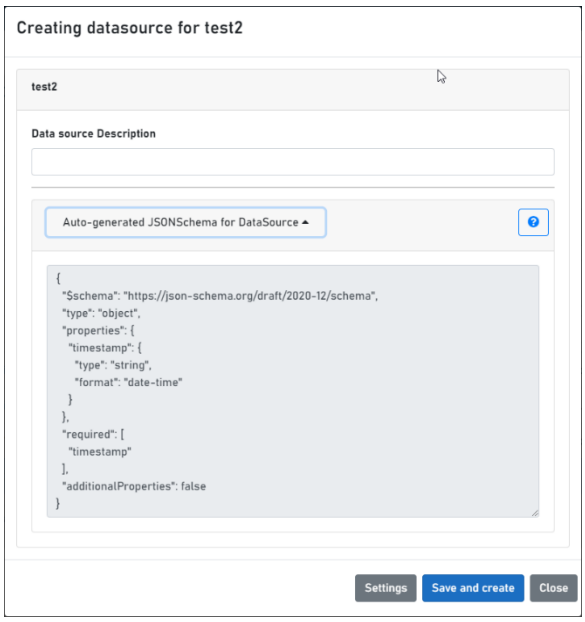

Figure 43. Proposed JSON Schema for Data Source connection

Additional advanced options may be configured before creating the connection by pressing the Settings button (**Figure 44**). These options configure internal parameters of the data platform behaviour and should be only modified by advanced users. The default values are optimal in most situations.

Figure 44. Advanced Settings for Data Source connection

REMINDER: Once a Feature of Interest is connected to a Data Source with ingested data, the FoI can't be deleted from the Data Space.

When a FoI is already connected to a Data Source, the Data Source button in its Edit FoI modal will display the Data Source Configuration modal (***Figure 45***). Here, information related to the Kafka Topic and Druid Table that define the Data Source is displayed. The Advanced Settings will still be available alongside a new Kafka Producer button. Pressing it will display a Python code template for creating a Kafka Producer for this Data Source (***Figure 46***). See additional details over the Kafka Producer field in the Exploring existing Data Sources section.

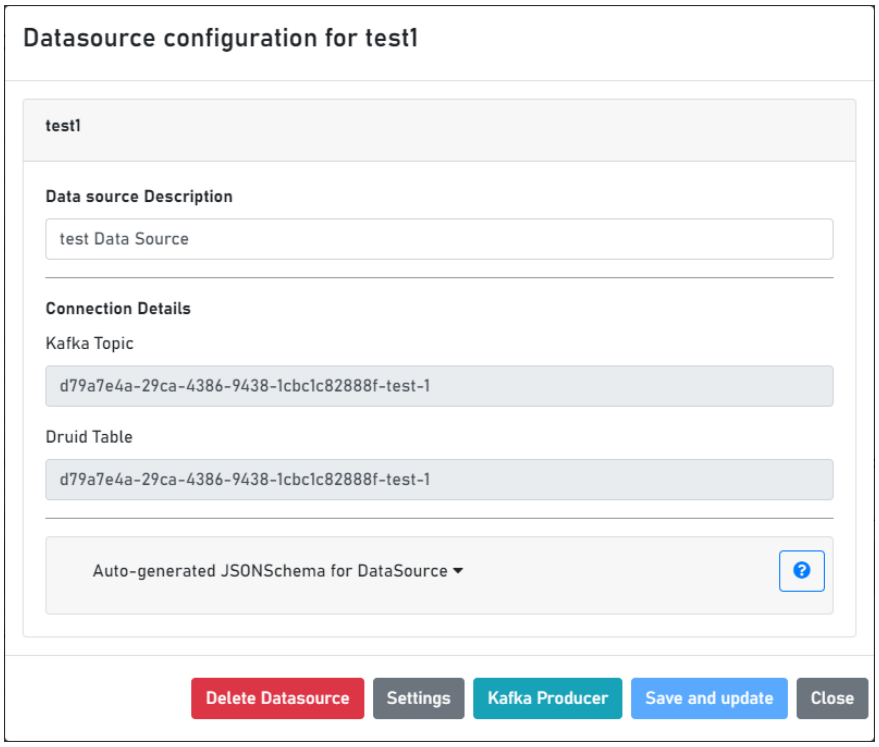

Figure 45. Data Source Configuration modal

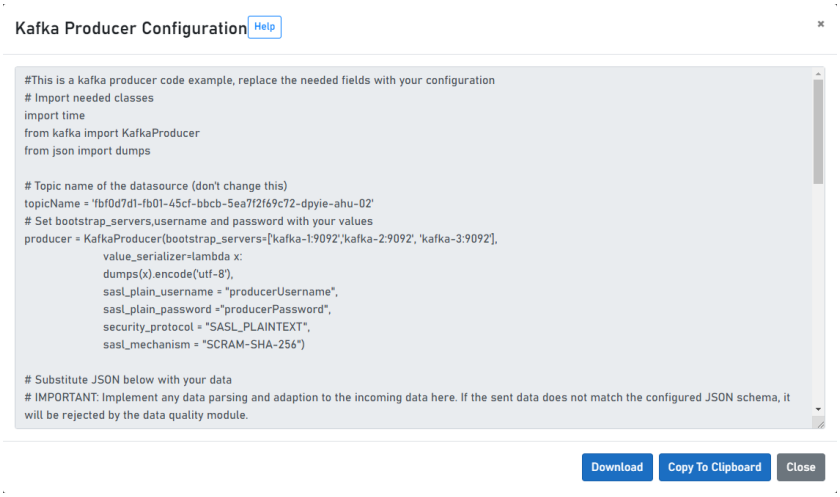

Figure 46. Kafka Producer code template

3.6.3.5 Deleting Data Sources

Data Sources' deletion falls under two possible conditions: those that haven't consumed data and those that have.

The deletion will be performed normally for Data Sources that haven't ingested data, and the Data Source will stop being available in the platform. All its support structures will also be deleted (Druid supervisor, Kafka topic, HDFS sink connector and HDFS folder)

However, deletion is not possible for Data Sources that contain data. Instead, the Data Source will be placed in an "Idle" state and remain connected to the related FoI. These kind of Data Sources will still be listed in the Data Source Manager view under "Disabled Data Sources" (**Figure *47***). All their dynamic support structures (Druid supervisor, Kafka topic and HDFS sink connector) will be removed, but no data loss will occur. FoIs whose Data Sources are set as obsolete can be connected to a new Data Source to continue ingesting data.

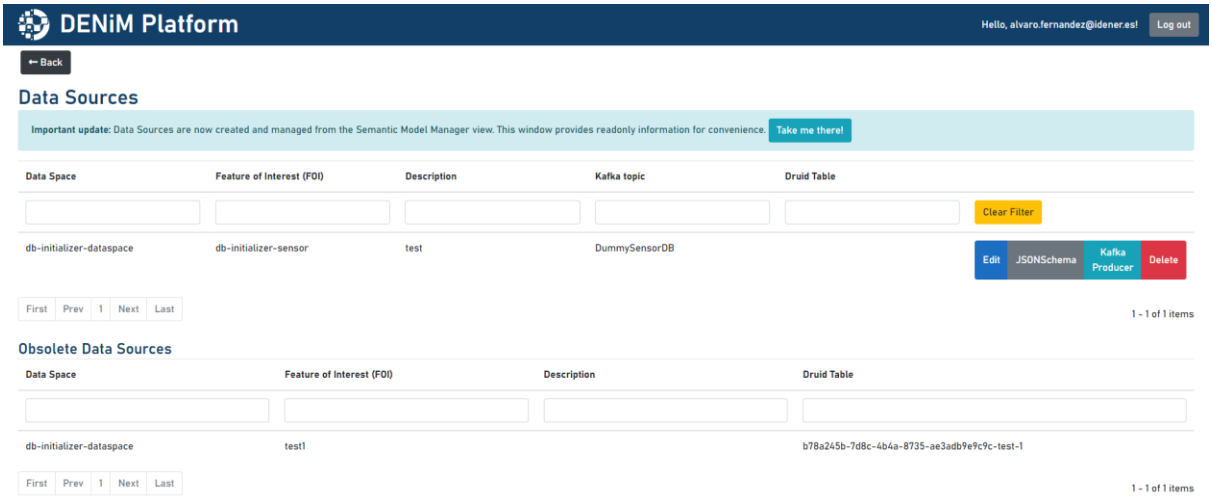

Figure 47. Obsolete Data Sources in the DSM view

To delete a Data Source, simply press the "Delete Data Source" button in their Data Source Configuration Modal (**Figure *45***), and the system will act according to the Data Source's state. After confirming the action, the Data Source will be removed from the FoI or set idle, but no data loss will be incurred due to this action.

3.6.3.6 Refreshing Data Sources

As it has been stated, the JSON Schema that governs the Data Sources is auto-generated from the Properties of the connected FoI. But, if the FoI properties can be edited, the Data Source will need to be updated too.

If changes are detected in the FoI properties, the Data Source Configuration Modal will display an additional button to Refresh the JSON Schema (***Figure 48***). Pressing this button will update the auto-generated JSON Schema, and users can apply these changes by pressing the Save and Update button.

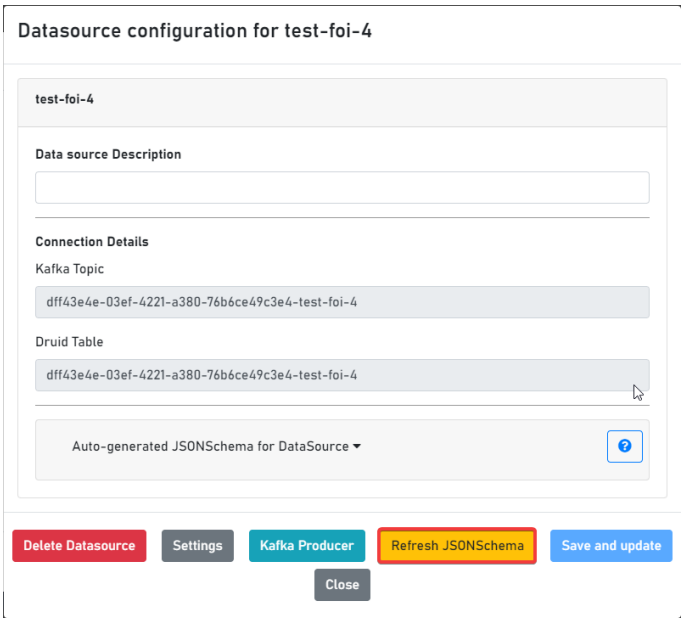

Figure 48. Refresh JSONSchema button available

The Data Source elements will be altered to account for the new (or removed) Properties, and certain policies are applied to maintain retro compatibility, as follows:

- For newly added Properties, additional columns are added to the corresponding Druid table. Their historic values up until their creation date will be filled with null or the given default value to maintain the timeseries.
- Removed Properties will see their related Druid table columns preserved as obsolete for consultation of already existing data.
- Property Type changes are not supported. A new Property (to be translated to a column) must be created instead.

3.7 DENiM Data Platform APIs

The DENiM Platform Frontend is the intended way for users to exploit the capabilities of the DENiM Data Platform. It provides a user-friendly approach to help users set up their Digital Twins and related applications and exploit them by connecting them to real-world data sources.

Nonetheless, the DENiM Platform also exposes an Application Programming Interface (API) for advanced users, which exposes all the main functionalities outlined in the frontend. The following sections will provide details on how to access these functionalities.

3.7.1 General API access and details

The DENiM Platform API is accessible through the regular Frontend deployment using the following URL:

https://denimplatform-develop.dtenabler.com/swagger

The API is presented through a Swagger Interface, which facilitates testing and executing operations with it and provides complete documentation of the APIs (**Figure *49***). Therefore, an extensive

documentation of the API endpoints will not be offered in this guide. The user is suggested to use the Swagger interface to learn about more details.

All the different APIs are accessed through the same Swagger interface, selecting them from the selector at the top-right side of the screen (1). The APIs are also subject to the same authorisation methods as the main Frontend interface. Users need to register with the DENiM Platform, as explained before, and enter their credentials in the Swagger interface, either in the general "Authorize" button (2) or individually for each method with their specific Auth button (3).

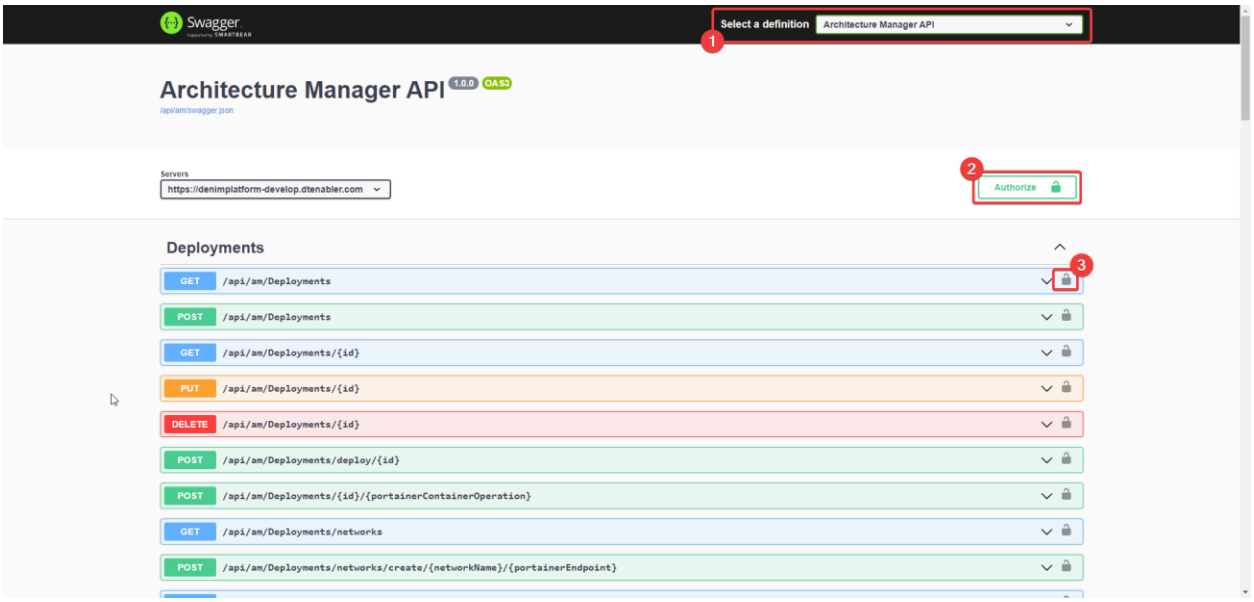

Figure 49. DENiM APIs Swagger Interface

Using either of the Authorization buttons will display the "Available authorizations" modal (***Figure 50***). Simply clicking the "Authorize" button (or a specific method's button, although this will only allow users to use said method) will take the user to the credentials page (the same as when accessing the Platform, as seen in **Figure *4***). The same user credentials are used.

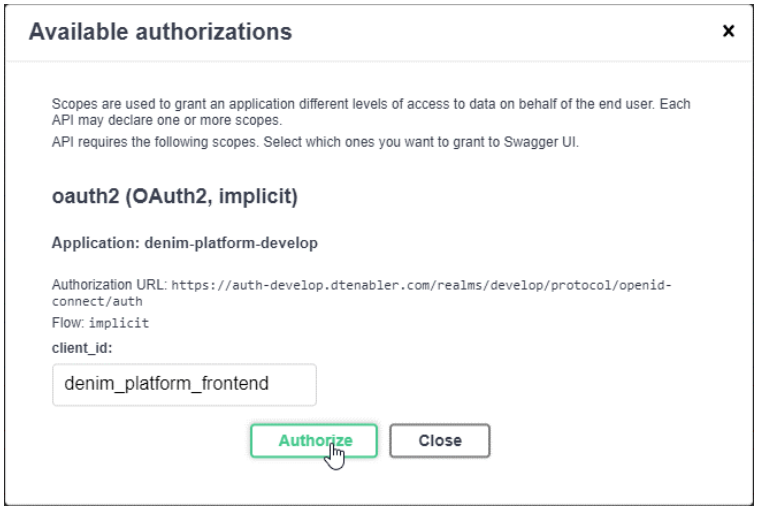

Figure 50. API Authorization

Each of the individual endpoints presented by each API gives details on how to call them correctly, including any parameters to be used directly in the URL and a complete JSON Schema of both the expected contents to be included as headers and the responses that will be provided (**Figure *51***). Use the "Try it out" button in the interface to test the method directly from the Swagger interface.

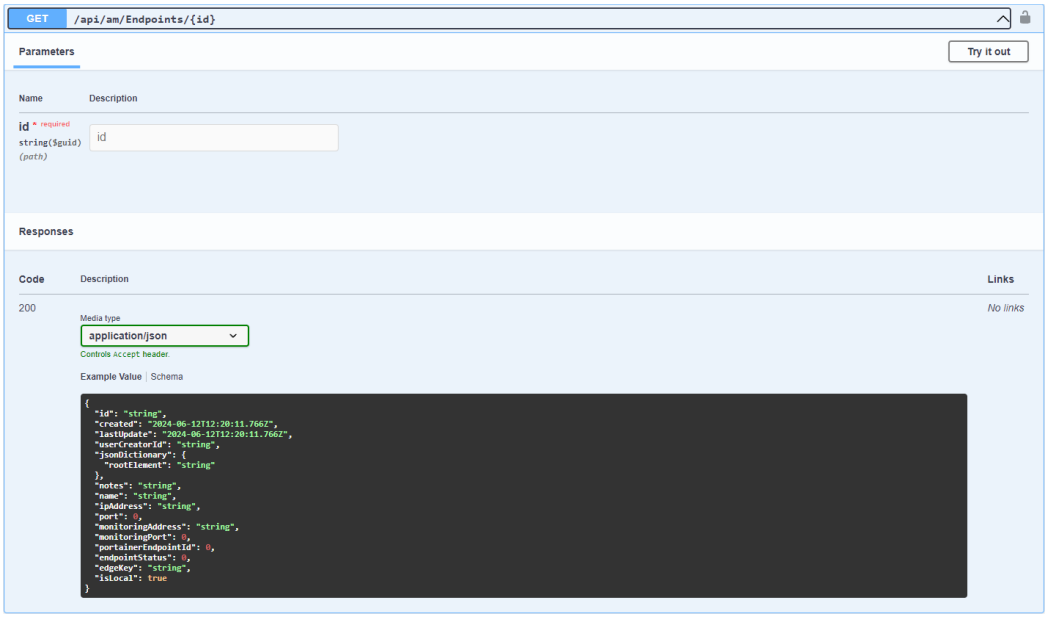

Figure 51. Swagger endpoint detailed

3.7.2 Architecture Manager API

The Architecture Manager API offers the complete CRUD (Create, Read, Update, Delete) operative over the structures the Architecture Manager controls, each with their own section in the Swagger Interface. These structures are, in order of appearance:

- Deployments: same concept as the Deployments explained in the Architecture Manager section. These represent an Application deployed to a certain Server.
- Endpoints: These are the different Servers to which we can make Deployments.
- Registries: The different Docker Registries for storing images used in Applications. Note that the API doesn't give access to the operations to build and upload Images to the default DENiM Registry. The operative is limited to the list of Registries available through the Platform.
- Stacks: These are equivalent to the Applications discussed in the Architecture Manager section. Represent one or various Images orchestrated through a Docker Compose, ready to be deployed to a Server.

3.7.3 Data Source Manager API

The Data Source Manager API implements complete CRUD operations over the Data Sources and their underlying support structures, including Druid tables, Kafka topics, and the HDFS Directory and Sink connector. Users don't need to concern themselves with these structures for any of the operations since their management is already accounted for when using any of the Data Sources endpoints.

3.7.4 Semantic Model Manager API

The Semantic Model Manager provides the complete CRUD over the elements that compose DENiM's Semantic Model, including:

- Data Spaces
- Features of interest
    - Their Properties
    - Their Relations
- System-level values:
    - Available Types for Features of Interest
    - Available Types of Properties
    - Available Types for Relations