D4.10 – i4Q Services for Data Analytics v2

WP4 – BUILD: Manufacturing Data Analytics for Manufacturing Quality Assurance

i4Q has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958205.

Document Information

GRANT AGREEMENT NUMBER	958205	ACRONYM	i4Q
FULL TITLE	Industrial Data Services for Quality Control in Smart Manufacturing		
START DATE	01-01-2021	DURATION	36 months
PROJECT URL	https://www.i4q-project.eu/		
DELIVERABLE	D4.10 – i4Q Services for Data Analytics v2		
WORK PACKAGE	WP4 – BUILD: Manufacturing Data Analytics for Manufacturing Quality Assurance		
DATE OF DELIVERY	CONTRACTUAL 31-Dec-2022	ACTUAL	30-Dec-2022
NATURE	Report	DISSEMINATION LEVEL	Public
LEAD BENEFICIARY	UNINOVA		
RESPONSIBLE AUTHOR	Ruben Costa (UNINOVA)		
CONTRIBUTIONS FROM	1-CERTH, 2-ENG, 5-KBZ, 7-IKER		
TARGET AUDIENCE	1) i4Q Project partners; 2) industrial community; 3) other H2020 funded projects; 4) scientific community		
DELIVERABLE CONTEXT/ DEPENDENCIES	This document describes the development of the final release of the i4Q Services for Data Analytics (i4Q^{DA}) solution. This is an updated deliverable with respect to its previous version D4.2 Services for Data Analytics.		
EXTERNAL ANNEXES/ SUPPORTING DOCUMENTS	None		
READING NOTES	None		
ABSTRACT	This deliverable describes the functionalities of the i4Q^{DA} solution, and the development of the final release of the i4Q^{DA} solution. This is an updated deliverable with respect to its previous version D4.2 Services for Data Analytics.		

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
0.1	15-Nov-2022	ToC	ToC of the v2 release is created	UNINOVA
0.2	25-Nov-2022	Draft	Final draft for internal review	UNINOVA
0.3	02-Dev-2022	Internal Review	Internal Review	UPV, FARPLAS
0.4	12-Dec-2022	Draft	Addressing comments from reviewers	UNINOVA
1.0	30-Dec-2022	Final document	Final quality check and issue of final document	CERTH

Disclaimer

Copyright message

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
CSV	Comma-separated values
DA	Data Analytics
DAG	Directed Acyclic Graph
DB	Database
DL	Deep Learning
DSS	Decision Support System
ETL	Extract, Transform, Load tasks
HTTP	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
LGBC	Light Gradient Boosting Classifier
ML	Machine Learning
PDF	Portable Document Format
RDBM	Relational DataBase Management (system)
SDK	Software Development Kit
SQL	Structured Query Language
UI	User Interface
WP	Work Package
XGBC	Extreme Gradient Boosting Classifier
XGBR	Extreme Gradient Boosting Regressor
YAML	Yet Another Markup Language

Executive summary

This document presents the final executive description of the **i4Q Services for Data Analytics** (i4Q^{DA}) solution providing the general description, the technical specifications and the implementation status. This is an updated deliverable with respect to its previous version D4.2 Services for Data Analytics. The deliverable **D4.10** is the Source Code of the final release of the i4Q^{DA} Solution in a private repository of Gitlab: https://gitlab.com/i4q/i4q-da.

The documentation associated with the i4Q^{DA} Solution is deployed on the website http://i4q.upv.es. This website contains the information of all the i4Q Solutions developed in the project "Industrial Data Services for Quality Control in Smart Manufacturing" (i4Q). The direct link to the i4Q^{DA} Solution documentation is http://i4q.upv.es/10_i4Q_DA/index.html.
Such documentation is structured according to:

- General description
- Features
- Images
- Authors
- Licensing
- Pricing
- Installation requirements
- Installation Instructions
- Technical specifications of the solution
- User manual

Document structure

Section 1: Contains a general description of the **i4Q Data Analytics**, providing an overview and the list of features. It is addressed to final users of the i4Q^{DA} Solution.

Section 2: Contains the technical specifications of the **i4Q Data Analytics**, providing an overview and its architecture diagram. It is addressed to software developers.

Section 3: Details the implementation status of the **i4Q Data Analytics**, explaining the final status, next steps and summarizing the implementation history.

Section 4: Provides the conclusions.

APPENDIX I: Provides the PDF version of the **i4Q < Data Analytics>** web documentation, which can be accessed online at: http://i4q.upv.es/10_i4Q_DA/index.html.

1. General Description

1.1 Overview

With the introduction of new fields and/or technologies, like Big Data and IoT, there is a growing need for fast and easy ways to grab all that data and use it to get information and/or to draw some conclusion about the origin of that data. That is where data analytics comes into play. To analyse any kind of data, the person making the analysis normally follows three steps, being first and foremost, making the data available, like extracting data from a CSV file; the second step is to prepare the data to be analysed like, for example, only selecting the fields necessary for the analysis, and, lastly, applying the analytical model to get some results, being a simple one like linear regression or more complicated ones like machine learning (neural networks, decision trees, etc.). When all these steps are connected, we obtain what is called a workflow. Having a workflow, facilitates the job of the person doing the data analysis, since there is not a need to be manually executing each step mentioned previously.

The i4Q Services for Data Analytics (**i4Q**^{DA}) solution aims to provide a Data Analytics experimentation environment that allows the creation of data analytics workflows in a dynamic way. This solution enables the creation of AI workflows in a simple and intuitive, code-free manner, building the workflow using a visual programming environment to place the components through drag-and-drop interface. The **i4Q**^{DA} solution is supported by Apache Airflow[1], a workflow orchestrator, which is used to create, schedule and monitor workflows, meaning it oversees the handling of execution, orchestration, management and storage of these workflows, and implements a number of operators and technologies that are used in the creation of these workflows, such as databases, machine learning libraries and platforms, among many other technologies.

1.2 Features

The **i4Q**^{DA} solution comprises four major components. The first one is the Apache Airflow platform, responsible for the orchestration of the workflows, meaning the scheduling, execution, monitoring and storage of the workflows. The second component is the set of supporting technologies, which encompasses all the necessary supporting technologies for workflow creation, as well as AI, Data Analytics and machine learning algorithms. The third component is the set of operators (connectors, transformers and analytics) used to create workflows, and finally, the last component is the **i4Q**^{DA} User Interface (UI), which guides users through the dynamic definition of workflows, by enabling the creation of workflows that can connect to data sources (data source configuration, connection and storage), perform data preparation and pre-processing (data filtering, aggregation, harmonisation and semantic enrichment) and apply AI methods for AI model training, updating and serving and Data Analytics (selection and configuration of AI methods).

This last component has the following specific features, that allow the dynamic creation of workflows:

[1] https://airflow.apache.org/

- Creation of Data Analytics Workflows - Visual programming of AI workflows, by using the drag and drop functionality to add operators and setting up dependencies between operators.
- Save and Load Workflows – Allows the saving of workflows in a database, so that it can be loaded and executed anytime, and allowing the user to make changes to previously done workflows.
- Creation and integration of new operators – The creation of user specific operators, through the help of guidelines, and integration of those specific on the workflow creation tool.
- Configuration of external tools and technologies – Full integration with tools and technologies, like Keras, Tensorflow, Sklearn, allowing an all-in-one box setting
- Execution and orchestration of workflows – Easy plug'n'play of user created workflows directly on the Airflow interface, allowing the execution and orchestration of multiple workflows.

2. Technical Specifications

2.1 Main User Interface

The main User Interface for the **i4Q^{DA}** is depicted in **Figure 1**, which highlights the main features of the interface (red boxes flagged with numbers):

- The main menu is placed on the top of the sidebar. This menu will provide access to the main options for each component, as well as for the initial interface. The components tree and a free-text search bar with autocomplete features, to look for specific components, are placed below the main menu, on the sidebar.
- The tabs bar is placed between the header and the main content area. It enables users to navigate between components, namely workflow/pipeline creation, addition of new operators, configuration of technologies and the Apache Airflow embedded user interface.
- The main content area is placed under the tabs bar and dynamically changes depending on the selected tab. Example interfaces and mock-ups will be presented in this section for the different components of the **i4Q^{DA}** solution.
- The header presents the logos and the user's information, notifications and settings menus.

All the information regarding the usage of this Main User Interface, its innerworkings, and integration/customization will be described in the following subsections, which divide the **i4Q^{DA}** into four components.

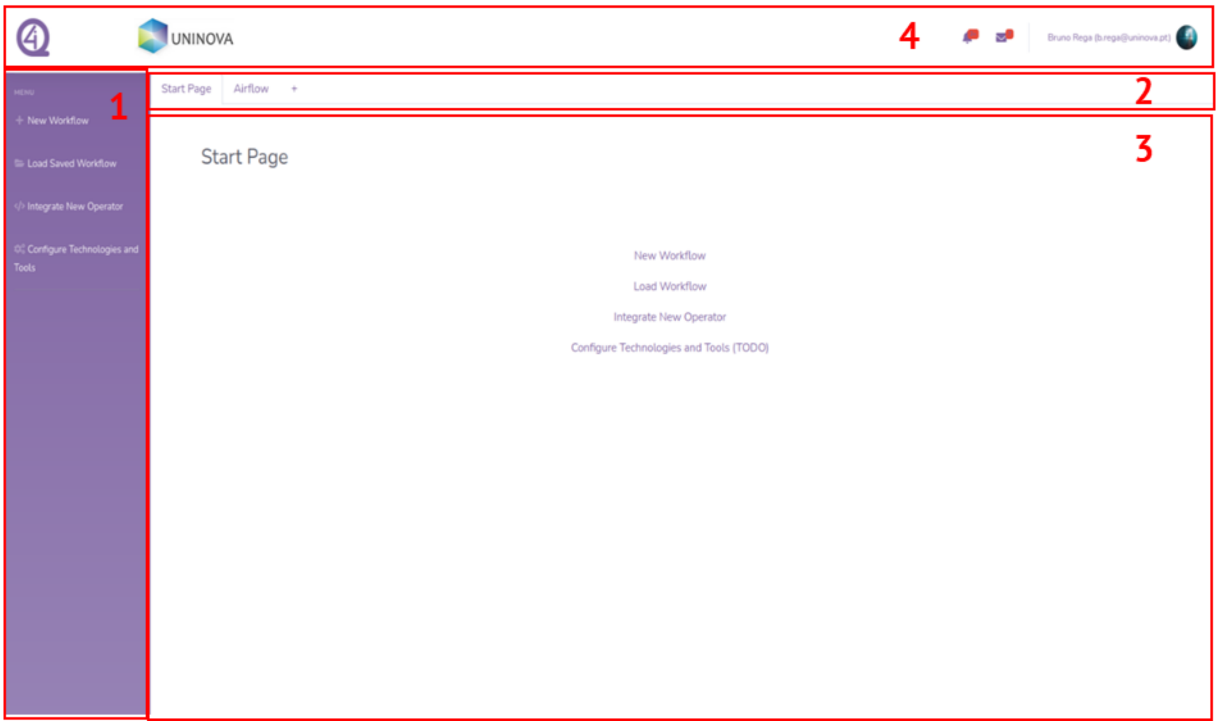


Figure 1. i4Q^{DA} Main User Interface

2.1.1 Workflow Creation Component

This component is responsible for the creation, definition and configuration of AI, ML and DL pipelines (i.e., workflows) to be executed in Apache Airflow[2]. To start creating a new workflow/pipeline, users must click the 'New Workflow' option on the Start Page or on the sidebar menu. After selecting the 'New Workflow' option, the window depicted in **Figure 2** is presented. This window allows users to select a unique name for their new workflow. The user inputs a unique name for the workflow and presses the 'Submit' button. Users can also cancel the creation of a new workflow, by selecting 'Cancel'.

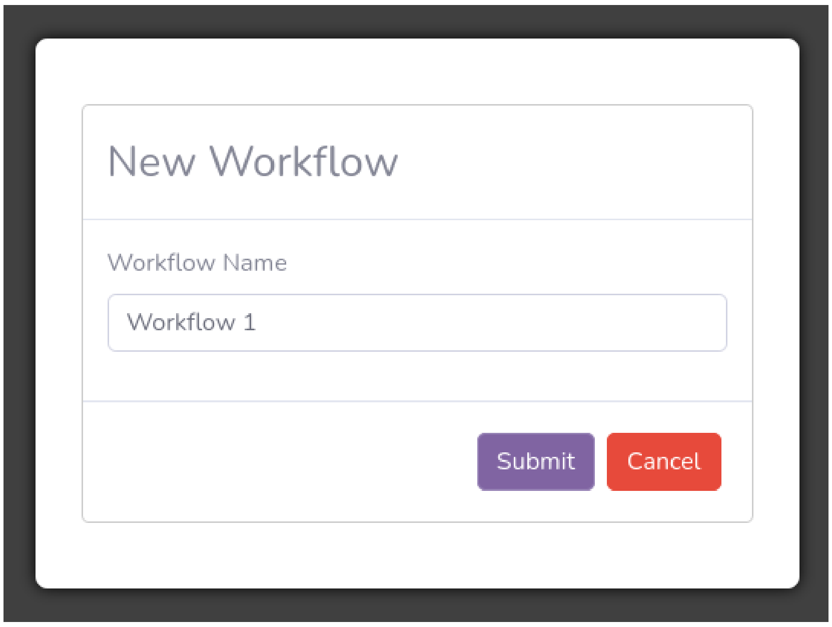


Figure 2. New Workflow Modal Window

After adding the name for the new workflow, a new tab for the workflow is added to the tabs bar (**Figure 1**-3) and the main content of that tab is also created (**Figure 1**-4), as depicted in **Figure 3**. The new workflow main content comprises four main components:

Bellow the tabs bar it is possible to see that the name of the workflow name is now visible inside the top-left hand side of the main content area. On the top-right hand side three new buttons were added. Those buttons are:

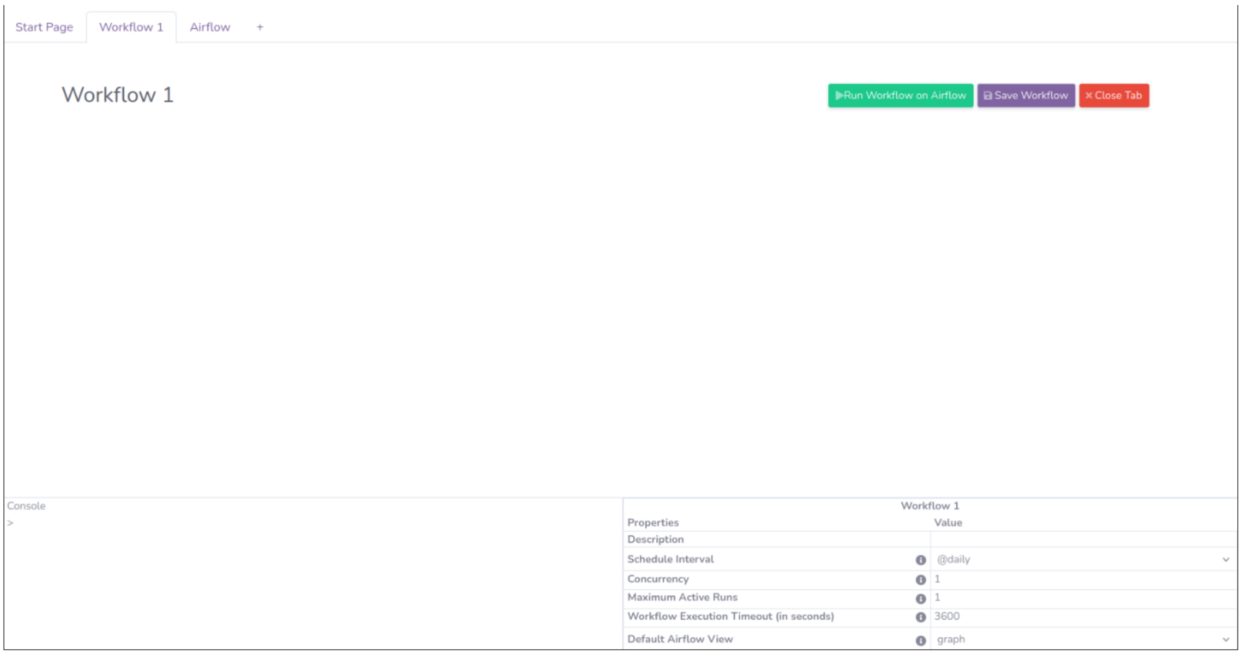
- "Run Workflow on Airflow" – "Compiles" the workflow and sends it to Airflow to be executed. This option is locked until the "Save Workflow" is pressed.
- "Save Workflow" – Saves the current workflow to the backend database.
- "Close Tab" – Closes the workflow tab.

The blank canvas in the middle of the main content area corresponds to the actual workflow creation space and will house the different operators that the user adds to the workflow. The user is free to arrange the various operators in any location inside this canvas since an operator can be dragged and dropped in any position on the canvas.

[2] https://airflow.apache.org/

The bottom right component is the Properties Table view, a dynamic table that enables users to edit workflow, operator, and technology parameters/attributes, depending on the type of tab that is opened in the main content. In the case of a workflow, users can edit specific characteristics of the workflow, such as adding a description or changing Airflow-specific attributes. On the other hand, when an operator is selected (on focus), the Properties Table will enable users to edit the operator's attributes (as presented in Figure 4).

The bottom left component is the Console view, which will present information about the communication with the backend and other useful information about the corresponding view in the main content.



Workflow 1

Properties	Value
Description	
Schedule Interval	@daily
Concurrency	1
Maximum Active Runs	1
Workflow Execution Timeout (in seconds)	3600
Default Airflow View	graph

Figure 3. Workflow Creation View

After creating a new workflow tab, users can start building the pipeline, by dragging operators from the 'Operators' sidebar menu and dropping them into the blank canvas of the main content area, as presented in **Figure 4**. It is also possible to see that the Properties Table View has some different content comparing with the Table in **Figure 3**. This happens when we select the operator, showing all the input parameters necessary for the operator, which in this case is the "File Type", the "File Path" and the "Data Provider". Connector Operators have a blue border, while the Transformer Operators have a green border. The Analytics Operators have a red border, as it is possible to see in **Figure 6**.

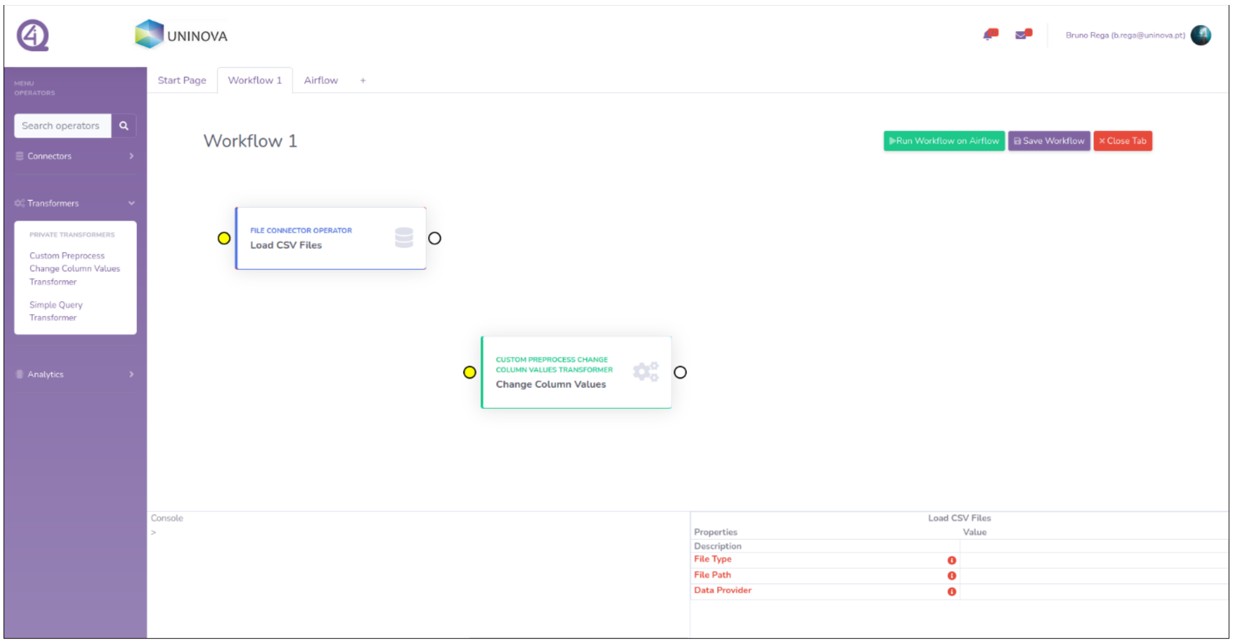


Figure 4. Adding and Editing New Operators

When a new operator is added, dragging an operator from the sidebar and dropping it in the canvas, the user is requested to input a unique name for the new operator, through the modal view presented in **Figure 5**.

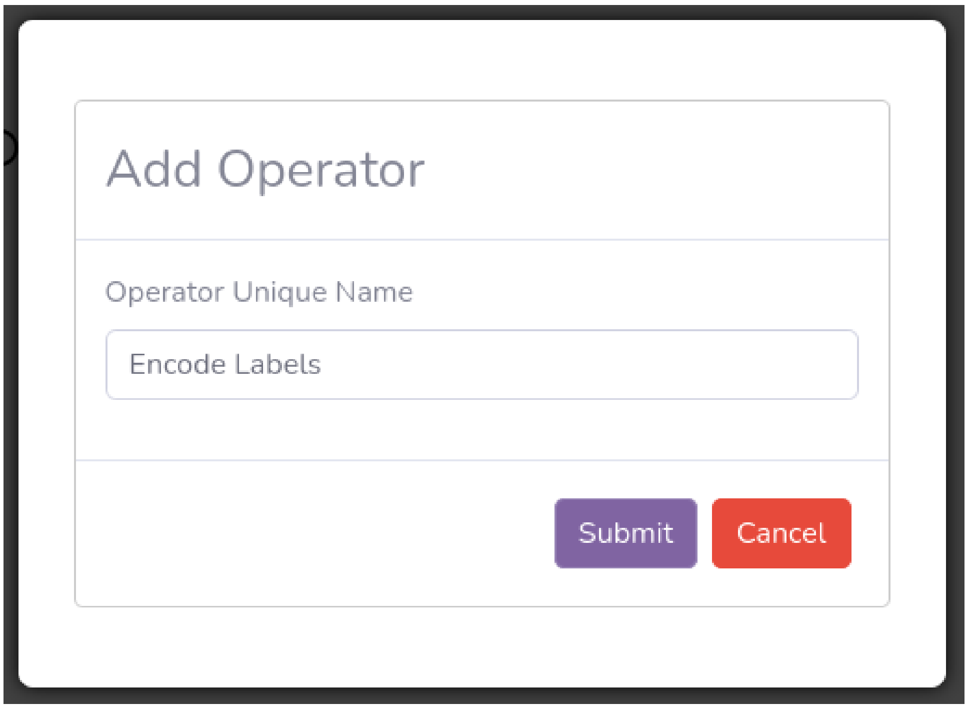
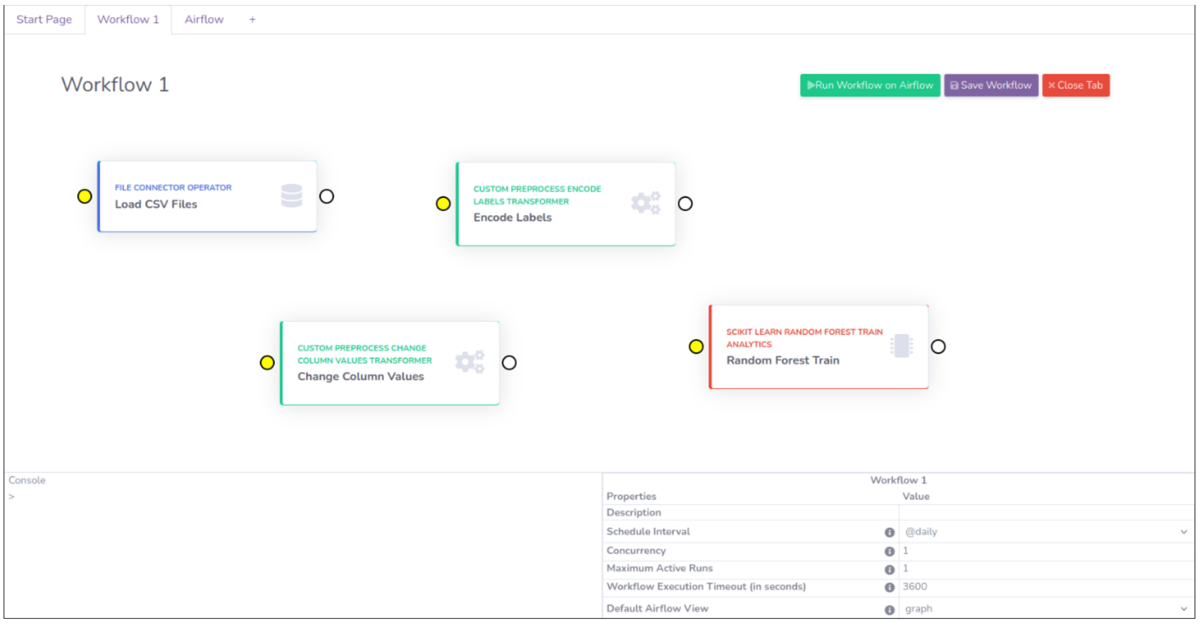


Figure 5. Add Operator Modal View

Figure 6 depicts a possible pipeline with one connector, two transformers and one analytic operator. Up to this point, users can create, save and delete workflows, configure workflow parameters (through the Properties Table), add operator rows, drag and drop operators and set up the operators (also through the Properties Table). The only step missing in workflow/pipeline creation is to add the dependencies between the different operators, to define the flow and order of tasks to be executed by Airflow.

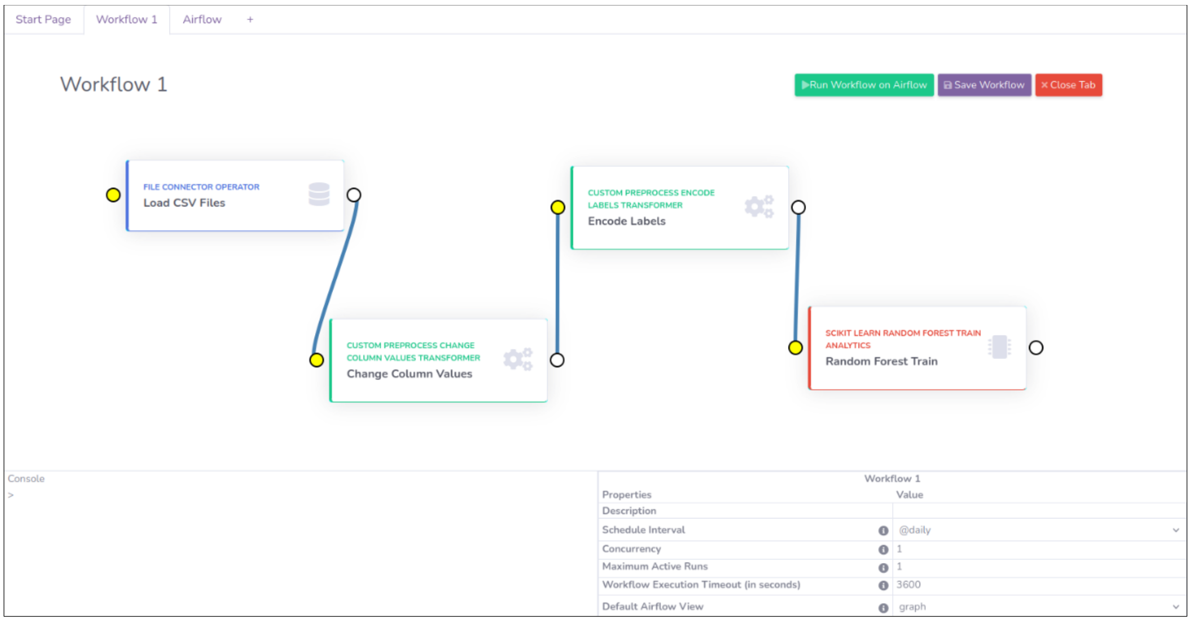


Workflow 1

Properties	Value
Description	
Schedule Interval	@daily
Concurrency	1
Maximum Active Runs	1
Workflow Execution Timeout (in seconds)	3600
Default Airflow View	graph

Figure 6. Workflow before Dependencies

To edit dependencies between operators, users must click and hold on a white circle in front of the operator's symbol and dragging the line to a yellow from another operator. A complete workflow with all the dependencies between all the operators can be seen in **Figure 7**.

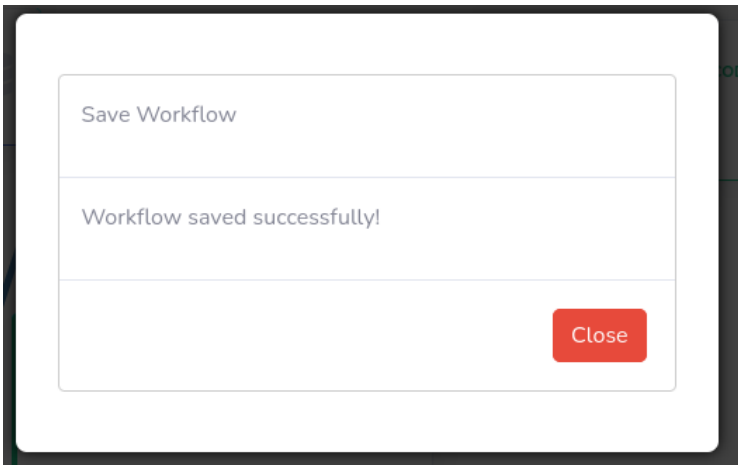


Workflow 1

Properties	Value
Description	
Schedule Interval	@daily
Concurrency	1
Maximum Active Runs	1
Workflow Execution Timeout (in seconds)	3600
Default Airflow View	graph

Figure 7. Final Sample Workflow

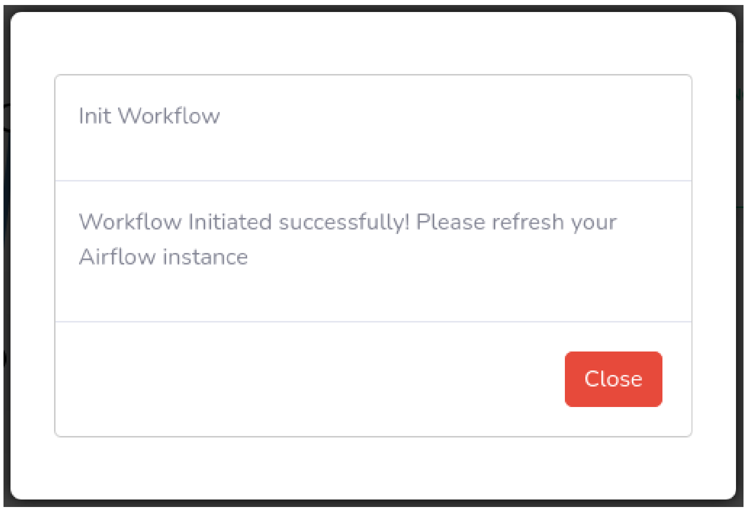
After completing a workflow, the user must save it to the backend database so that it can be accessed and edited at any other time. To do that, the user needs to press the "Save Workflow" button, which will then prompt the modal view shown in **Figure 8**.



Save Workflow

Workflow saved successfully!

Figure 8. Save Workflow Modal View

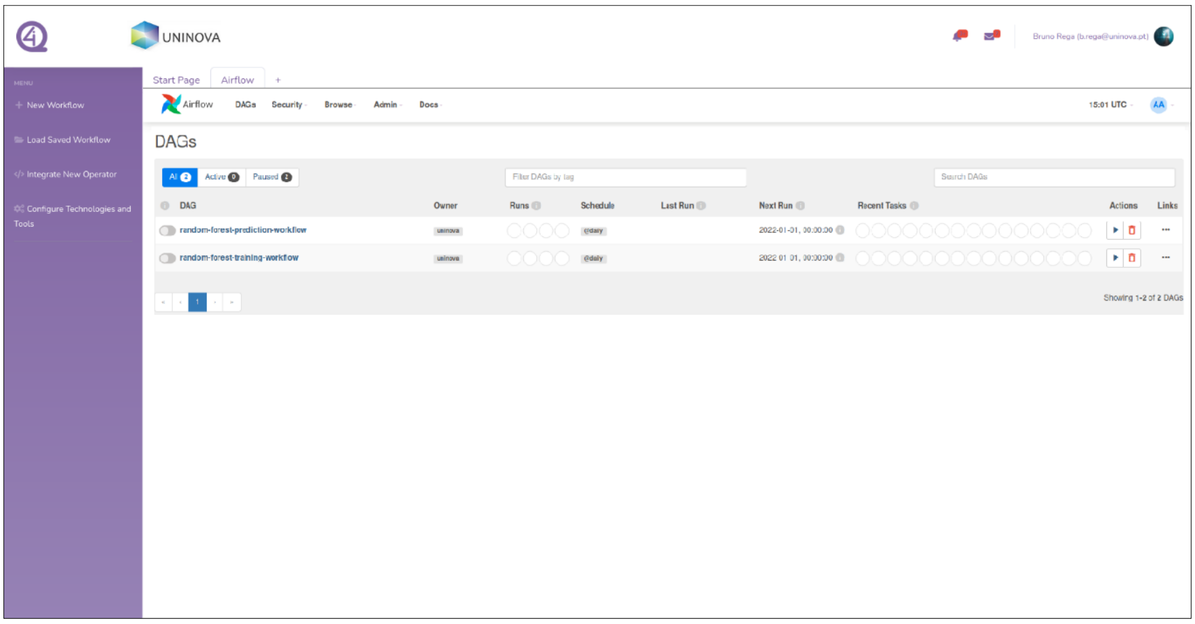


Init Workflow

Workflow Initiated successfully! Please refresh your Airflow instance

Figure 9. Init Workflow Modal View

The last step is to execute the created workflow on Airflow. For that, a specific button will be added with the name 'Initialise Workflow'. When users click the 'Initialise Workflow' button, the workflow is converted to an Airflow workflow and executed directly on Airflow. Further, users will have access to an embedded version of Airflow in a specific tab in the tabs bar, as presented in the mock-up of **Figure 10**. This embedded view will enable users to monitor, orchestrate and execute the pipelines created using the **i4Q**^{DA}.

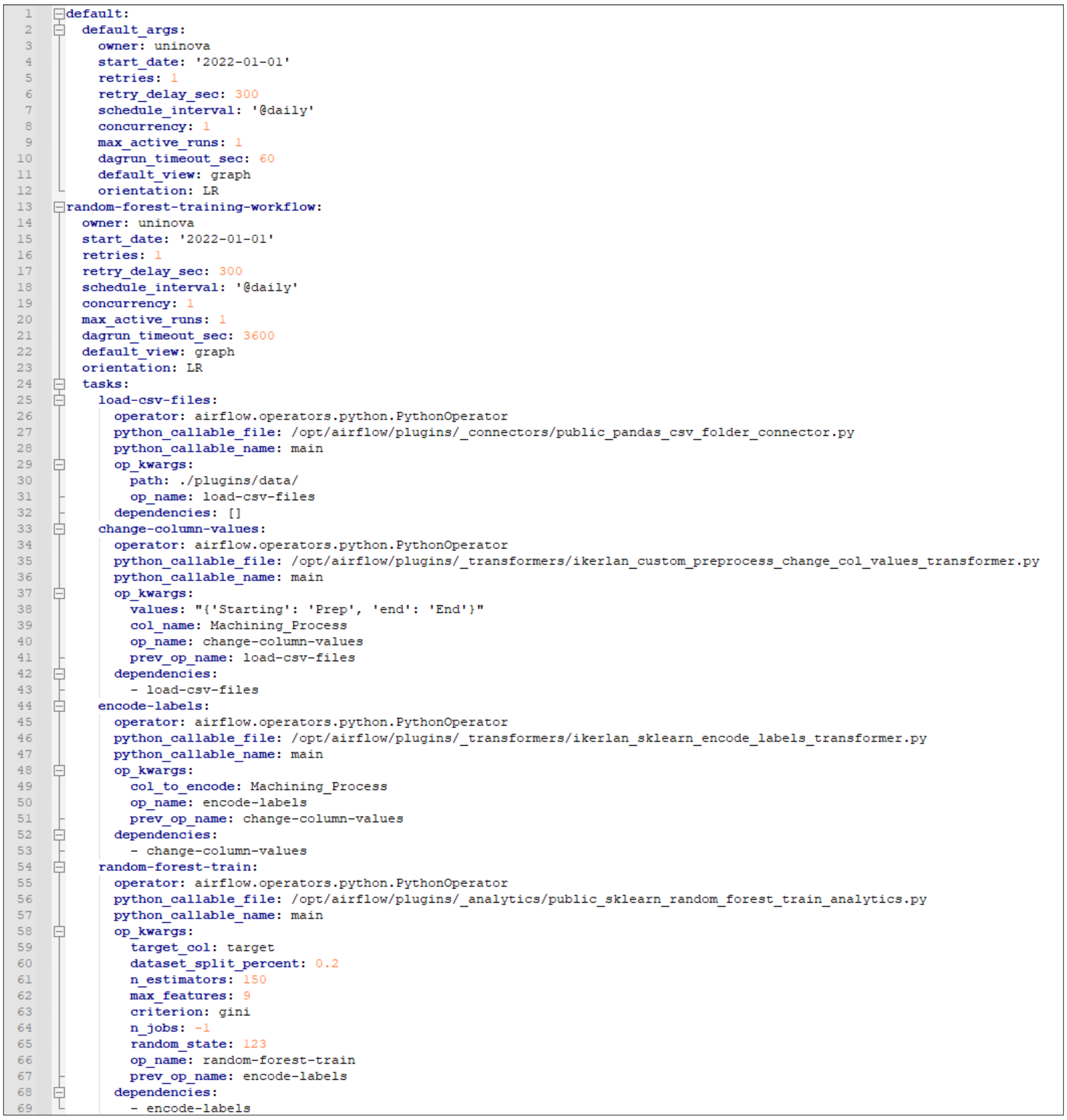


DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
random-forest-prediction-workflow	uninova		@daily		2022-01-01, 00:00:00			
random-forest-training-workflow	uninova		@daily		2022-01-01, 00:00:00			

Figure 10. Embedded Airflow Tab

2.1.2 Airflow Orchestration Component

This integration goes beyond the backend execution of AI pipelines, by embedding the Airflow user interface into the Data Analytics Environment. When users click the 'Initialise Workflow' button, the workflow is converted to an Airflow workflow and executed directly on Airflow. This is done with the usage of a DAG Factory library that enables the dynamic creation of workflows for Airflow. Since Apache Airflow requires that workflows are created programmatically, these workflows are static, and do not have an easy way to add, edit or remove operators, to edit the workflows. This library takes a YAML configuration file, with all the necessary configuration parameters for a workflow and its operators and converts it into a format that Airflow can interpret. This way, every time there is a need to change something in the workflow, like the type of the operator, there is only the need to change that component in the YAML file, and the DAG Factory will oversee the interpretation of the configuration file and subsequent creation/update of the workflow on Airflow. An example of a YAML file with a configuration of a simple workflow can be seen in **Figure 11**. In the image it is possible to observe many elements that represent some of the configurations necessary to build a workflow through DAG Factory, alongside the operators' respective configurations. This configuration file is divided into two groups. The first group, which has the name "default", represents some default parameters for the execution of the workflow, i.e., it contains the information that will be used to describe how the workflow will be executed. In the Figure, the fields that are present are the "schedule_interval", containing the information that relates to how often will the workflow be executed (e.g., '@daily' means the workflow will run once a day), the "concurrency", which represents how many tasks can be executed inside the workflow at the same time, the "max_active_runs", which is the number of instances of the workflow that can be executed simultaneously, and the "dagrun_timeout_sec", representing the amount of time, in seconds, that the workflow can be up and running, before timing out and/or failing. The remaining fields are the "default_view" and "orientation", which represent the configurations for the visualisation of the workflow in the Apache Airflow web application, the "owner" that has the name of the owner of the workflow, the "start_date" field with the starting date for its execution, the “retries” field with how many retries should the airflow make in case of failure, and finally the “retry_sec” field, with the information of how much time, in seconds, before each retry.



```yaml
default:
  default_args:
    owner: uninova
    start_date: '2022-01-01'
    retries: 1
    retry_delay_sec: 300
    schedule_interval: '@daily'
    concurrency: 1
    max_active_runs: 1
    dagrun_timeout_sec: 60
    default_view: graph
    orientation: LR
random-forest-training-workflow:
  owner: uninova
  start_date: '2022-01-01'
  retries: 1
  retry_delay_sec: 300
  schedule_interval: '@daily'
  concurrency: 1
  max_active_runs: 1
  dagrun_timeout_sec: 3600
  default_view: graph
  orientation: LR
  tasks:
    load-csv-files:
      operator: airflow.operators.python.PythonOperator
      python_callable_file: /opt/airflow/plugins/_connectors/public_pandas_csv_folder_connector.py
      python_callable_name: main
      op_kwargs:
        path: ./plugins/data/
        op_name: load-csv-files
      dependencies: []
    change-column-values:
      operator: airflow.operators.python.PythonOperator
      python_callable_file: /opt/airflow/plugins/_transformers/ikerlan_custom_preprocess_change_col_values_transformer.py
      python_callable_name: main
      op_kwargs:
        values: "{'Starting': 'Prep', 'end': 'End'}"
        col_name: Machining_Process
        op_name: change-column-values
        prev_op_name: load-csv-files
      dependencies:
        - load-csv-files
    encode-labels:
      operator: airflow.operators.python.PythonOperator
      python_callable_file: /opt/airflow/plugins/_transformers/ikerlan_sklearn_encode_labels_transformer.py
      python_callable_name: main
      op_kwargs:
        col_to_encode: Machining_Process
        op_name: encode-labels
        prev_op_name: change-column-values
      dependencies:
        - change-column-values
    random-forest-train:
      operator: airflow.operators.python.PythonOperator
      python_callable_file: /opt/airflow/plugins/_analytics/public_sklearn_random_forest_train_analytics.py
      python_callable_name: main
      op_kwargs:
        target_col: target
        dataset_split_percent: 0.2
        n_estimators: 150
        max_features: 9
        criterion: gini
        n_jobs: -1
        random_state: 123
        op_name: random-forest-train
        prev_op_name: encode-labels
      dependencies:
        - encode-labels
```

Figure 11. YAML example

The second group relates to the composition of the workflow, with all its tasks and the individual parameters of each task. In the example of **Figure 11**, the second group is called “random-forest-training-workflow”, which is the name given to the workflow. Inside this group it is possible to observe that some of the configurations from the previous group are present in this group as well, in case the user/owner wants to make some changes related to those fields, but also has two new parameters, which are the “description” field and the “tasks” field. The first field is self-explanatory, as it contains a brief description of the workflow, while the “tasks” field contains all the tasks that will be present in this workflow.

For this example, the workflow is composed of four operators, which are the "load-csv-files" operator, the "change-column-values" operator, the "encode-labels" operator, and the "random-forest-train" operator. The first operator is a connector type of operator, meaning it will connect to a data source to ingest some data, and in this example, this operator is in charge of ingesting data from multiple CSV files that are in a directory/folder. Inside this task it is possible to observe all the parameters that define a task, like the "operator", which refers to the Airflow operator mentioned above, and that in this case is a "PythonOperator", indicating that this task will be executed by running a Python script. Since Airflow also needs to know what script needs to be tor to execute this task, the second and third parameters, which are the "python_callable_file" and the "python_callable_name", contain the information about the directory of the script and the Python function that needs to be called, respectively. The "op_kwargs" field contain all the specific parameters that relate to the chosen Operator (load-csv-files) and so, for this example, a CSV folder files connector needs the "path" where the files are located. The "op_name" field is just the operator's name, which will be used inside the scripts. This field is not filled by the user but the Workflow Creation Component.

The following two tasks are from the transformers operator's category, namely the "change-column—values" and the "encode-labels". Both these operators will do some pre-processing on the data ingested in the previous operator. For the "change-column-values", this operator will change the values in a column, by other values chosen by the user. It is possible to see that this task has the same fields as the "load-csv-files", with the exception of the content of the "op_kwargs", which in this case has the "values" and the "col_name" fields. The first field contains all the pair values to be substituted in the column with the name "col_name". For this example, the column is the "Machining_Proccess", and the values are "Starting" to be replaced by "Prep" and "end" to be replaced by "End". The "op_name" is the same type of field as the previous task, and the "prev_op_name" is the same type of field but with the name of the last operator, and it is also not filled by the user. Another difference is that the Python script will not be the same as the previous task, and that this task will have its dependencies list with the name of the task previous to itself, meaning that the "load-csv-files" needs to finish before Airflow can start executing the "change-column-values" task. The other transformer operator is the "encode-labels". This operator will encode the values of a column, in this case the "Machining_Process", but only after the task "change-column-values" is completed.

The last task is from the analytics operators and is the "random-forest-train", which will be in charge of training and saving a random forest model. This task has the same structure relating to the other tasks mentioned above, with the "op_kwargs" field having the specific fields for this operator. The fields are the "target" with the name of the column for the training of the model, the "dataset_split_percent" with the percentage for the split of the data set (20% for the test and 80% for training in this case), the "n_estimators" with the number of estimators, the "max_features" with the maximum number of features, the "criterion" name, the "n_jobs" with the number of jobs and the "random_state" component. All of these are values specific for the training of a random forest model. The last two ("op_name" and "prev_op_name") follow the same rule as the previous tasks.

As mentioned above, after having this YAML file with all the operators for the desired workflow, the Airflow Orchestration Component will oversee all the scheduling and the orchestration of this workflow. Further, users will have access to an embedded version of Airflow in a specific tab in the tabs bar. This embedded view will enable users to monitor, orchestrate and execute the pipelines that were created using the Data Analytics Environment.

2.1.3 Operator Integration Interface

Although a set of built-in operators (i.e., atomic tasks within the pipeline) will be provided, the Data Analytics Environment will enable the integration of new, custom operators. This component is responsible for providing the necessary user interfaces and backend logic to add new operators to the environment.

It can be accessed by clicking on the link 'Add New Operator' in the Start Page menu. This view enables users to add their own operators to the Data Analytics Environment. Three inputs are needed:

- The new operator's name: A unique name for the new operator to be added/integrated.
- The operator's script: The script (python, bash, etc.) that executes the operator's task. This script comprises the code of the operator itself.
- The operator's JSON configuration file: This file contains all the parameters and information about the operator.

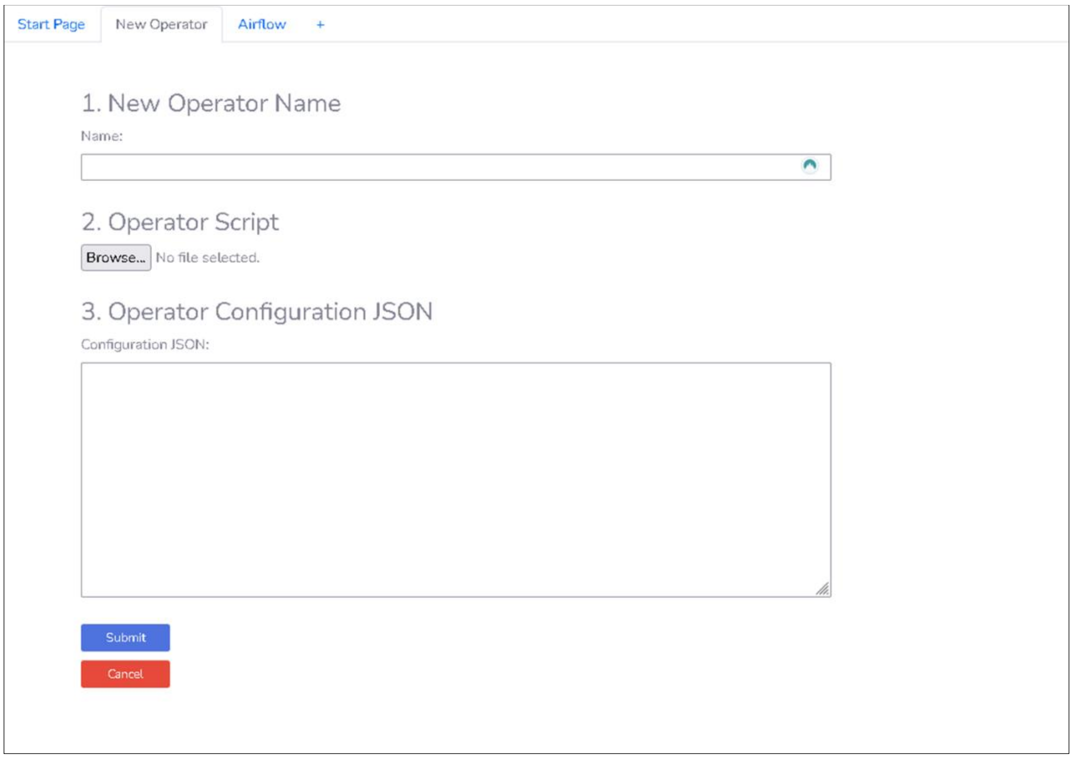


Figure 12. New Operator Interface

2.1.4 Software Development Kit

To integrate a new operator, users must account for the method to pass data between operators within Airflow. Since every operator is seen as an atomic micro-service, the passage of data between adjacent operators must be handled by a shared common database that can be used to store results from one operator and load these results to the subsequent operator. The database used internally for data transfer between adjacent operators is MongoDB[3].

Hence, a Software Development Kit (SDK) that supports data storage and loading to/from MongoDB was developed. Depending on the data structure used within each operator, the Software Development Kit provides specific storage and loading functions. For instance, if an operator uses a Pandas Dataframe as the main data structure to process data inside it, then data can be loaded from MongoDB to a Pandas Dataframe and stored from a Pandas Dataframe to MongoDB. The only exception is for connector operators, since these collect/store data from/to external sources, such as external databases and files or from an external Kafka stream. In **Figure 13**, green arrows represent the data path for transformer and analytics operators (Load – Process - Store) while the blue arrows depict the data path for connector operators (Load from External Source – Store; Load – Store to External Source).

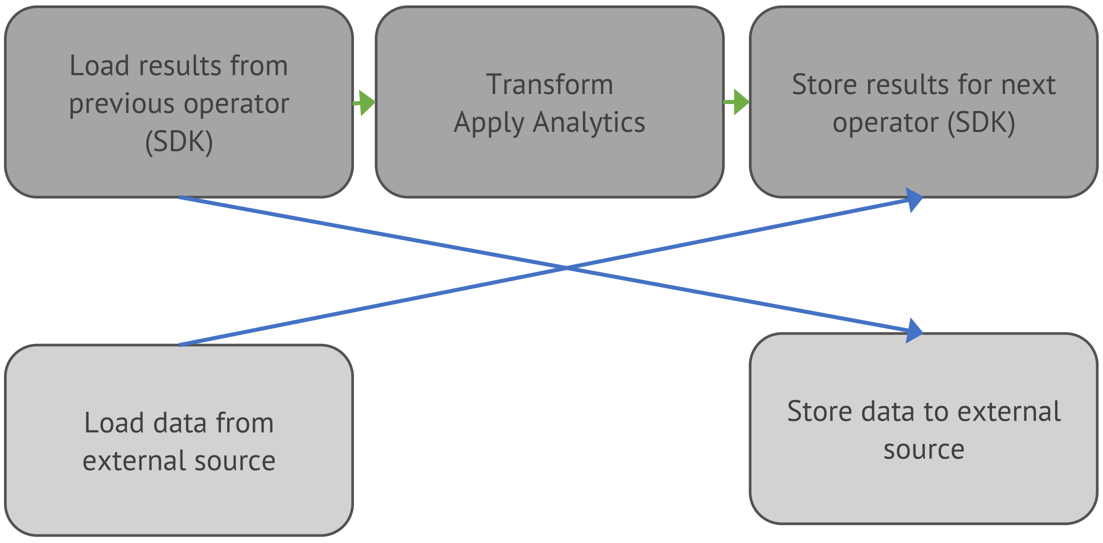

Load results from previous operator (SDK)
Transform Apply Analytics
Store results for next operator (SDK)
Load data from external source
Store data to external source

Figure 13. Data paths inside operators

Table 1 presents a list of data storage and loading functions provided by the Software Development Kit for a set of integrated technologies, such as Pandas and Apache Spark, just to name a few examples.

Table 1. Software Development Kit functions

Library	SDK Function	Description	Parameters
Dask[4]	saveDaskDataframe	Saves a Dask DataFrame into a MongoDB collection	• **df:** the Dask dataframe object • **db_name:** the name of the MongoDB database • **coll:** the name of the MongoDB collection

[3] https://www.mongodb.com/
[4] https://www.dask.org/

	loadDaskDataframe	Loads a Dask DataFrame from a MongoDB collection	• **db_name:** the name of the MongoDB database • **coll:** the name of the MongoDB collection
	saveDaskBag	Saves a Dask Bag into a MongoDB collection	• **bag:** the Dask bag object • **db_name:** the name of the MongoDB database • **coll:** the name of the MongoDB collection
	loadDaskBag	Loads a Dask Bag from a MongoDB collection	• **db_name:** the name of the MongoDB database • **coll:** the name of the MongoDB collection
Pandas[5]	savePandasDataframe	Saves a Pandas DataFrame into a MongoDB collection	• **df:** the Pandas dataframe object • **db_name:** the name of the MongoDB database • **coll:** the name of the MongoDB collection
	loadPandasDataframe	Loads a Pandas DataFrame from a MongoDB collection	**db_name:** the name of the MongoDB database **coll:** the name of the MongoDB collection
Spark[6]	saveSparkDataframe	Saves a Spark DataFrame into a MongoDB collection	• **df:** the Spark dataframe object • **db_name:** the name of the MongoDB database • **coll:** the name of the MongoDB collection
	loadSparkDataframe	Loads a Spark DataFrame from a MongoDB collection	• **db_name:** the name of the MongoDB database • **coll:** the name of the MongoDB collection
MLFlow[7]	saveModelMLFlow	Saves model to MLFlow	• **model:** the model object • **identifier:** the unique identifier for the model • **flavour:** the type of model (e.g., scikit-learn, PyTorch, TensorFlow)

[5] https://pandas.pydata.org/
[6] https://spark.apache.org/
[7] https://mlflow.org/

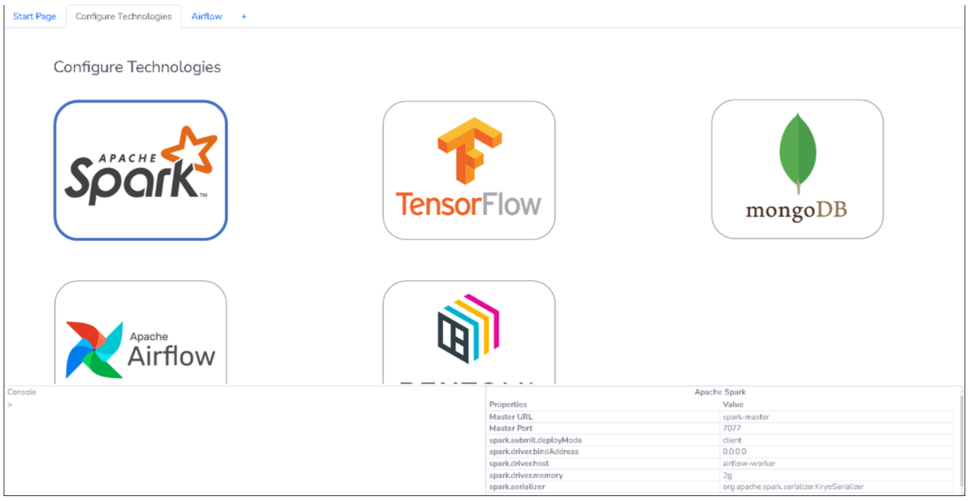
	loadModelMLFlow	Loads model from MLFlow	• **identifier:** the unique identifier for the model • **flavour:** the type of model (e.g., scikit-learn, PyTorch, TensorFlow)

2.1.5 Technology Configuration interface

This tool aims to simplify the process of configuring a data analytics environment with open-source tools that usually require multiple configurations and installations to run.

This component enables the selection and configuration of the desired technologies by the user via a user guided interface where the user selects the required functionalities and respective tools to meet those requirements. This tool will provide an environment automatically to meet the required functionalities or specifications required by the user.

This component also enables the use of external instances of integrated technologies. This means that users with private instances of the integrated technologies (e.g., Apache Spark, TensorFlow) can configure the Data analytics Environment to use these external instances, instead of the internal instances it contains.



Apache Spark

Properties	Value
Master URL	spark-master
Master Port	7077
spark.submit.deployMode	client
spark.driver.bindAddress	0.0.0.0
spark.driver.host	airflow-worker
spark.driver.memory	2g
spark.serializer	org.apache.spark.serializer.KryoSerializer

Figure 14. Technologies Configuration

The integrated technologies and libraries at the time of Release 2.0 are:

- Internal
  - **Apache Airflow:** Internal AI workflow orchestrator and executer
  - **MongoDB:** Internal data storage
  - **MLFlow:** Internal model storage and life-cycle manager
- Data Processing + Analytics
  - **Apache Spark:** Data connection and processing; AI algorithms
  - **Dask:** Data processing
  - **Pandas:** Data connection and processing

- **Scikit-Learn**[8]**:** AI algorithms
- TensorFlow[9] + Keras[10]: AI algorithms
- **PyTorch**[11]**:** AI algorithms

2.2 User Management

To approach the challenge of providing different users, with different needs, control over their own AI workflows and integrated operators, a User Management layer was added to the i4Q^{DA} solution, featuring User registration, login, user activation and password recovery. This layer enables users to store and load their own AI workflows, to integrate their own private or public operators. It also restrains access from users outside a particular company to access the company's AI workflows, when the i4Q^{DA} is deployed in a shared environment.

The login view is presented when a new user arrives at the solution's user interface (**Figure 15**). This view prompts users to provide their email and password to login into their account, enabling new users to register (through the 'Create an Account' link) and existing users to reset their password (through the 'Forgot Password?' link). Each user account will comprise the user's workflows and custom operators. After a successful login, users will be redirected to the main user interface (**Figure 1**).

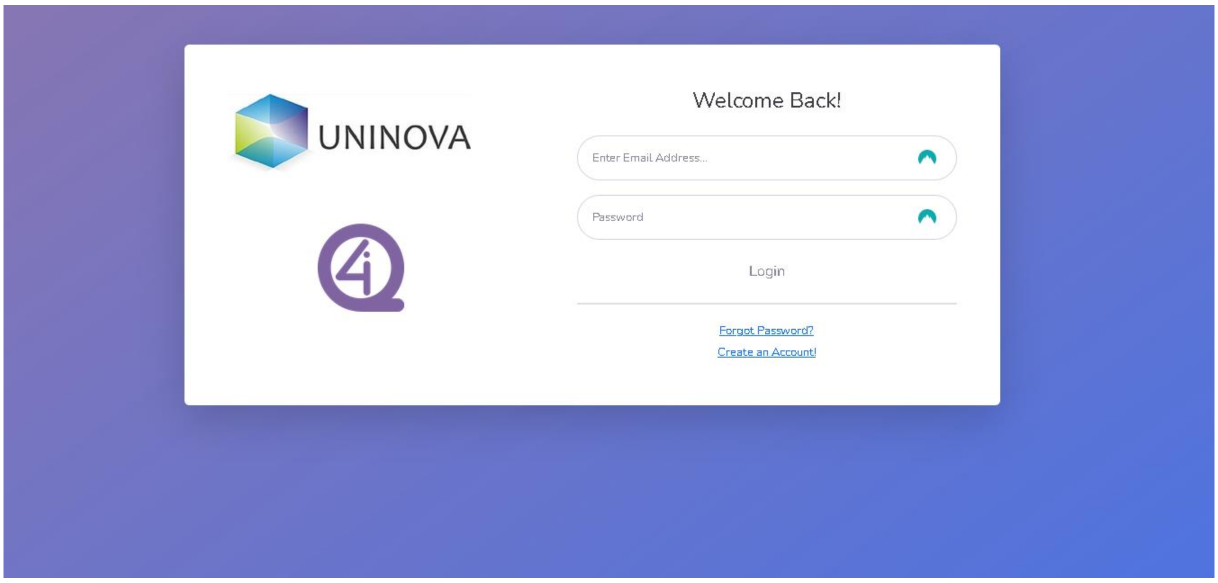


Figure 15. Login View

When a user does not have an account, it is possible to register a new one through the registration view (**Figure 16**). This view prompts users to input their first and last names, email address and password. When a new user registers an account, not only the account is created for i4Q^{DA}, but also the Airflow instance associated with the app. Hence, the user also registers a new account on Airflow.

[8] https://scikit-learn.org/
[9] https://www.tensorflow.org/
[10] https://keras.io/
[11] https://pytorch.org/

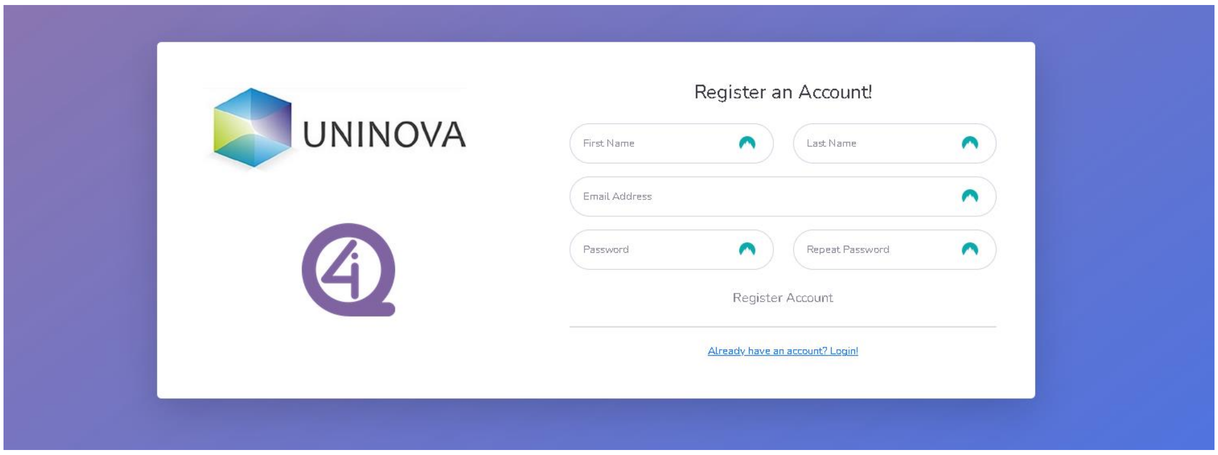
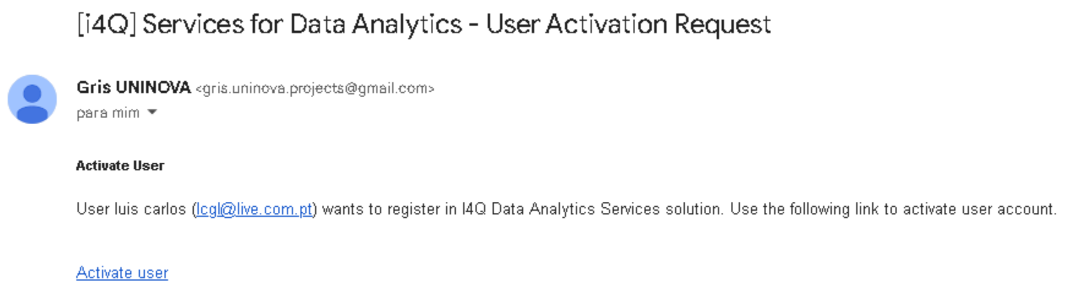


Figure 16. Registration View

For security purposes, new accounts need to be activated by an administrator user. This step provides extra security against unwanted registrations (e.g., users that do not belong to the company running the solution). When a new account is registered, an email is sent to the administrator user, so the new user account can be activated through a link sent in the email (**Figure 17**). By clicking on the link, the administrator activates the new user account. Accounts that an administrator does not activate cannot log in into the solution.



[i4Q] Services for Data Analytics - User Activation Request

Gris UNINOVA <gris.uninova.projects@gmail.com>
para mim

Activate User

User luis carlos (lcgl@live.com.pt) wants to register in I4Q Data Analytics Services solution. Use the following link to activate user account.

Activate user

Figure 17. Email - Activate New User

Finally, the final group of views corresponds to the process of resetting lost passwords. When a user lost the password, he/she can reset the password by selecting the 'Forgot password?' option in the login view. The user is redirected to the forgot password to view and prompted to input the email for the account (**Figure 18**).

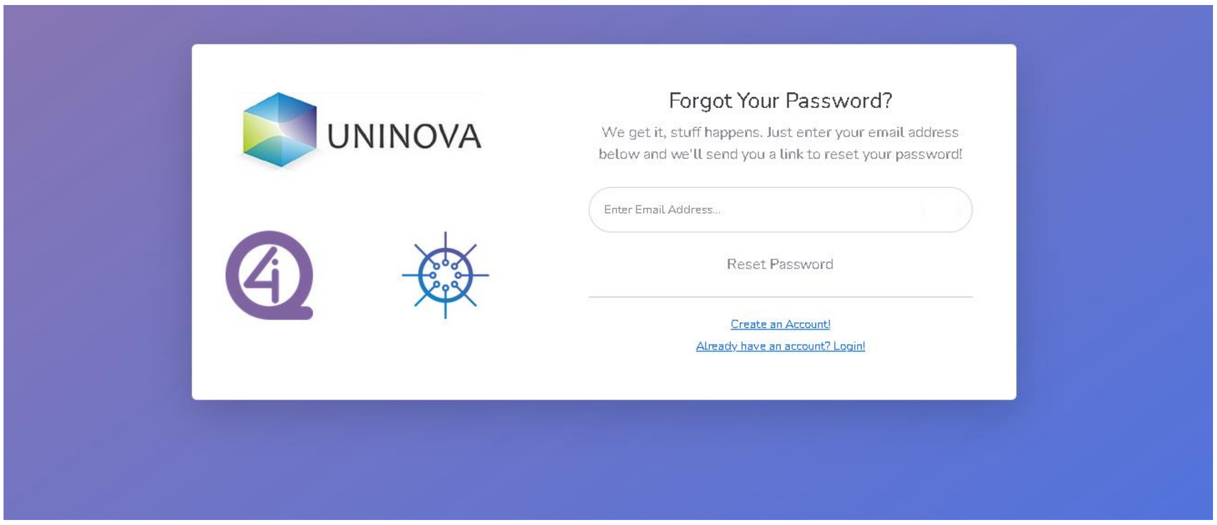
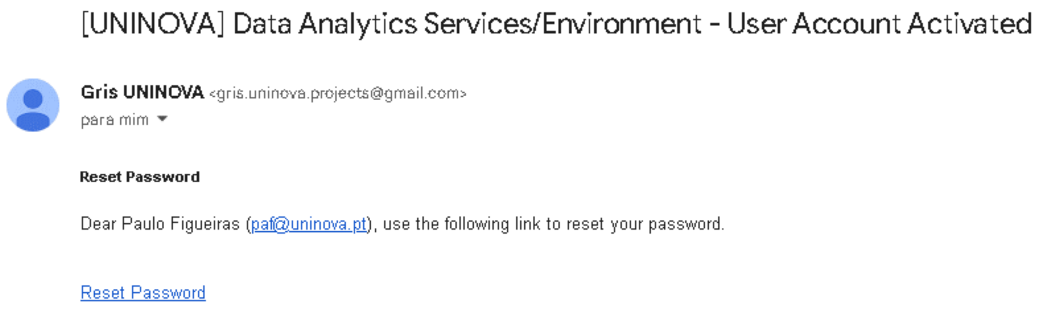

UNINOVA
Forgot Your Password?
We get it, stuff happens. Just enter your email address below and we'll send you a link to reset your password!
Enter Email Address...
Reset Password
Create an Account!
Already have an account? Login!

Figure 18. Forgot Password View

After inputting the email on the forgot password to view, the user will receive an email with a link to start the reset password process (**Figure 19**).



[UNINOVA] Data Analytics Services/Environment - User Account Activated

Gris UNINOVA <gris.uninova.projects@gmail.com>
para mim

Reset Password

Dear Paulo Figueiras (paf@uninova.pt), use the following link to reset your password.

Reset Password

Figure 19. Email - Reset Password

Lastly, the user is redirected to the Reset Password view by clicking on the Reset Password link in the email (**Figure 20**). In this view, users are prompted to input their new password and confirm the new password.

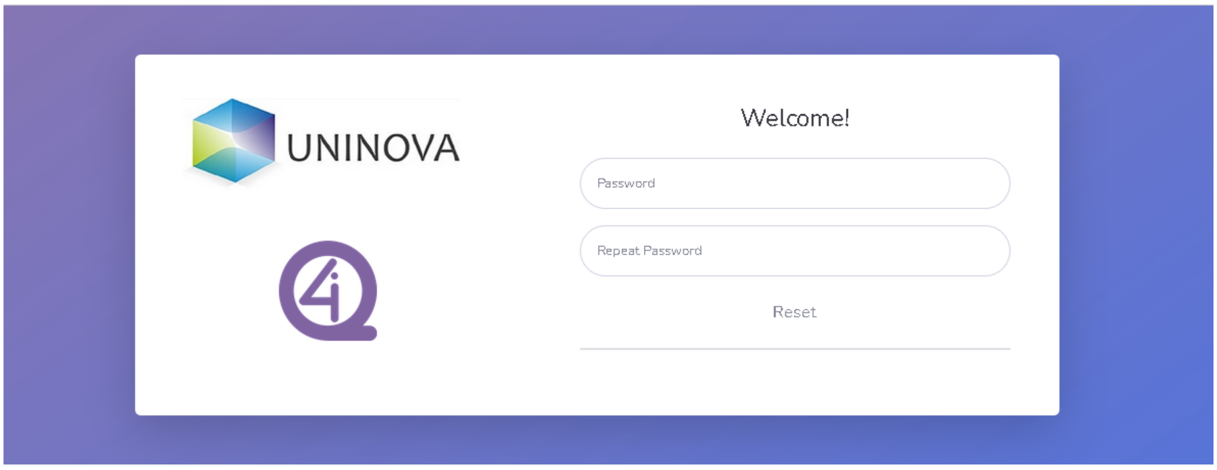


Figure 20. Reset Password View

2.3 Built-in Operators

Release 2.0 of the **i4Q^DA** solution comprises a set of built-in operators that can be used to build complete AI workflows and to test, evaluate and validate the solution's overall performance and usage experience. Table 2 describes the list of available built-in operators, already developed and integrated into **i4Q^DA**. With connector operators, data can also be stored into an external data source, although the description in Table 2 only accounts for loading external data.

Table 2. List of built-in operators

Type	Operator	Description	Techs
Connector	public_pandas_csv_folder_connector	Loads all the CSV files inside a folder to a single Pandas Dataframe	Pandas
	public_spark_csv_file_connector	Loads a CSV file into a Spark Dataframe	Spark
	public_spark_json_connector	Loads a JSON file into a Spark Dataframe	Spark
	public_spark_rdbms_connector	Loads the data from a RDBMS database into a Spark Dataframe	Spark
	public_webservice_connector	Performs an HTTP request to an API/Web service and the result is loaded into a Spark Dataframe	Spark
	public_kafka_connector	Loads a micro-batch of Kafka stream into a Pandas Dataframe	Pandas
Transformer	public_change_col_values_transformer	Changes the values of a column	Pandas
	public_sklearn_encode_labels_transformer	Encodes labels on a column, prior to training a model	Scikit-Learn Pandas
	public_aggregate_groupby_transformer	Performs an aggregation based on the groupby SQL clause	Spark
	public_convert_types_transformer	Converts the data type of the selected column to the type given by the user	Spark
	public_LabelEncoder_transformer	Encodes the selected column to values between 0 and the number of distinct classes minus 1	Scikit-Learn
	public_OneHotEncoder_transformer	Creates a new column for each category of a feature in the selected column, with binary encoding (0 or 1) to	Scikit-Learn

		indicate whether the row belongs to the category or not.	
	public_filter_dataset_transformer	Filters the dataset and returns the columns selected by the user	Spark
	public_filter_missing_values_transformer	Filters out the entries of a dataset where missing values exist in the selected columns	Spark
	public_normalize_columns_transformer	Normalizes the selected columns by mapping all the values of those columns to new values between 0 and 1 according to their original value	Spark
	public_remove_columns_transformer	Removes the selected columns	Spark
	public_remove_duplicates_transformer	Removes duplicate entries from the dataset	Spark
	public_rename_column_transformer	Renames the selected column	Spark
	public_replace_infinite_values_transformer	Replaces all the infinite values with Null values	Pandas Spark
	public_replace_missing_values_transformer	Replaces missing values for a new value selected by the user	Spark
	public_select_columns_transformer	Selects the desired columns and saves them to a new dataset containing only the selected columns	Spark
	public_spark_simple_query_transformer	Performs an SQL query provived by the user	Spark
Analytics	public_sklearn_random_forest_train_analytics	Trains a random forest classifier model using the scikit-learn random forest classifier library and saves the trained model to the MLFlow model library.	Scikit-Learn Pandas
	public_sklearn_random_forest_predict_analytics	Predicts results using a random forest classifier model created with the previous operator	Scikit-Learn Pandas
	public_spark_kmeans _analytics	Clusters results with a K-means clustering algorithm	Spark

	public_XGBClassifier_train_analytics	Trains an XGB classifier and saves the trained model to the MLFlow model library	MLFlo w
	public_XGBClassifier_predict_analytics	Applies an XGB classifier to the data and returns the resulting prediction in a new Spark Dataframe	Spark
	public_XGBRegressor_train_analytics	Trains an XGB regressor and saves the trained model to the MLFlow model library	MLFlo w
	public_XGBRegressor_predict_analytics	Applies an XGB regressor to the data and returns the resulting prediction in a new Spark Dataframe	Spark
	public_LGBClassifier_train_analytics	Trains an LGB classifier and saves the trained model to the MLFlow model library	MLFlo w
	public_LGBClassifier_predict_analytics	Applies an LGB classifier to the data and returns the resulting prediction in a new Spark Dataframe	Spark
	public_LGBRegressor_train_analytics	Trains an LGB regressor and saves the trained model to the MLFlow model library	MLFlo w
	public_LGBRegressor_predict_analytics	applies an LGB regressor to the data and returns the resulting prediction in a new Spark Dataframe	Spark

2.4 Architecture Diagram

The i4Q^{DA} solution includes processes and AI models which are mapped to the Platform and Edge Tiers of the i4Q Reference Architecture.

- **Platform Tier**: The services that are being used in this tier are the "Distribution Services", "Data Brokering" and "Storage, Data Transformation" and the "Data Analytics". Amongst these services, the most important for this solution is the "Data Analytics and services", which is in charge of analysing big amounts of data that can come in streams or as in a batch while providing many types of data analytic techniques aided by machine learning, that can be adapted based on the data type and the type of analysis. This customization, which can be done by the end-user, makes this a versatile analytical tool.
- **Edge Tier**: The i4Q^{DA} solution is comprised of the "Data Collecting" and "Data Management" services, which will be present in the tool itself and are fundamental for the data analytics, and the "Distributed Computing" services, which will allow this tool to be executed in any type of computer/device.

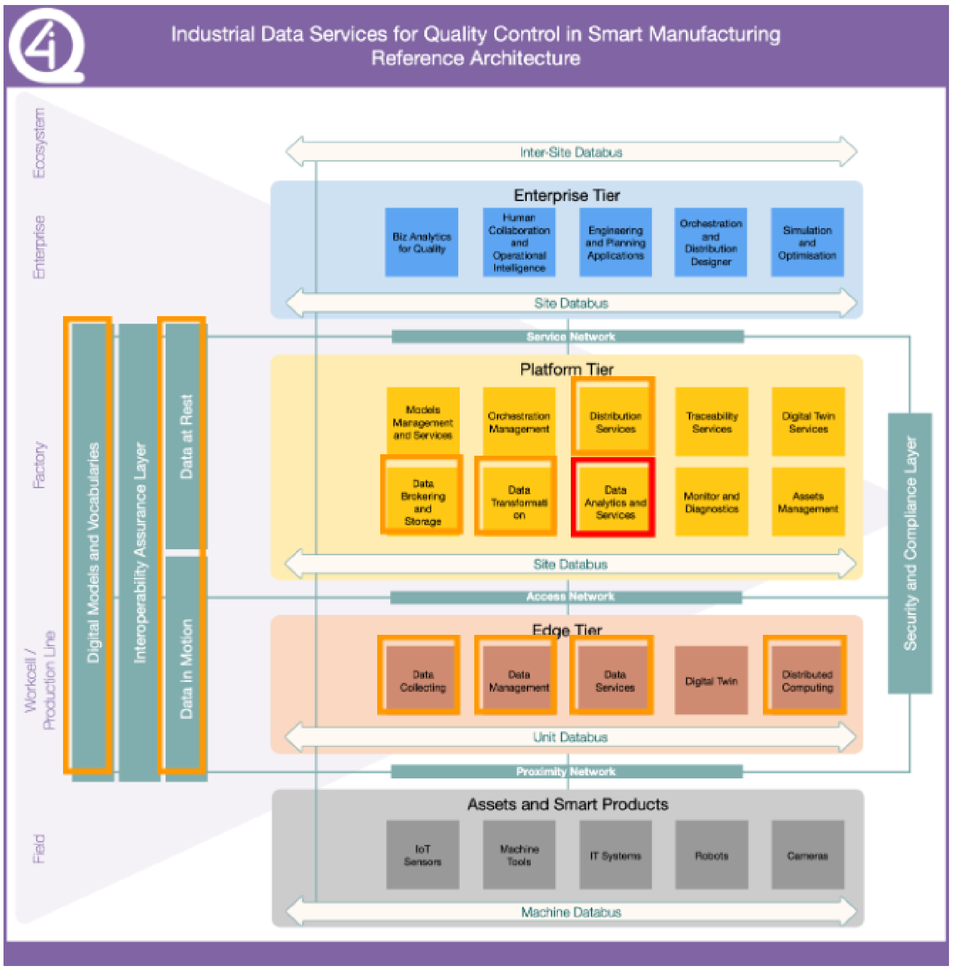


Figure 21. Reference Architecture

3. Implementation Status

3.1 Final Release implementation

The i4Q^{DA} is comprised of four major components, Apache Airflow, supporting technologies, a complete set of operators and the i4Q^{DA} dynamic workflow creation tool UI.

Regarding Apache Airflow the first developments were regarding the creation and configuration of workflows to be executed as well as the ability to save, edit and schedule their execution. This release introduces the creation of Airflow users that match the users registered on the i4Q^{DA} solution and the addition of several new operators, already described in section 2.3.

In order to work properly, the workflow creation tool needed some base tools and operators to allow the creation of the first workflows. Some of these operators consisted in connectors to multiple databases both SQL (PostgresSQL) and NoSQL (MongoDB), Apache Spark to perform ETL tasks in a distributed way, scikit-learn to perform ML preparation and models definition and MLflow to manage the execution of ML models. The final release integrated even more technologies, such as TensorFlow and Keras, PyTorch and Dask. This integration enables users to select from a range of libraries, technologies and data structures to be used together in the creation of the AI workflows. Users can integrate new operators that use these technologies, and regarding data structures, users can now handle their data with Spark, Pandas and Dask dataframes, and Dask bags. MLFlow was also integrated for storing and loading AI models that are trained within the i4Q^{DA} solution.

For the workflow creation tools User Interface and respective connection to the backend, the current iteration can be summarized into the following list:

- Create AI Workflows – Allows users to create AI workflows, from a data connection to pre-processing, training and prediction of results, through a visual block coding user interface. The main idea is that users lacking coding skills can create their own no-code AI workflows.
- Integrate new operator – Allows the incorporation of user created operators, which then can be used in that user's workflows. This feature gives a layer of customization, allowing the user to build more case specific workflows.
- Save and Load workflow – Allows the user to save the current progress of any workflow being constructed in the user interface, and to load any previous saved workflow. These functionalities give the user the capacity not needing to finish the workflow in one go, and/or to be able to edit any previous finished workflows.
- Initialization of workflow (batch) – This allows the initialization of the user-created workflow in the UI tool into Apache Airflow, which a will take care of the execution and orchestration of that initialized workflow. The user also has access to an embedded instance of the Airflow in the UI.
- Consume Kafka messages (stream) - Allows the user to consume and process messages from the i4Q message broker or any other Kafka cluster.

3.1.1 Solution features analysed and mapping with user requirements

A set of features has already been developed for i4Q^{DA}, based on the set of user requirements referring to i4Q^{DA} (Deliverable 1.9) and in line with the functional viewpoints (Deliverable 2.6). Similar requirements have been assigned into common categories of tasks based on an extensive technical study conducted on user requirements, available datasets, etc., introduced to ensure the generalization abilities of the i4Q^{DA} solution.

- [PC4r2.1.1] and [PC2r7], which are requirements from the model data block, are covered by the analytics operators and MLFlow, which is the main framework used to store and load models to be used.
- [PC3r1.1.1], [PC3r1.1.2], [PC5r4.2] and [PC4r2.1.2], which are requirements from the execute models block, are covered by the feature of the analytics operators, in the UI, that will have operators for model execution, which then will be orchestrated and executed by Airflow.
- [PC4r6.1.1.1] and [PC3r2.2], which are requirements from the train model block, are covered by the feature of the analytics operators, in the UI, that will have operators for model training and saving.
- [PC2r11] relates to the connect to data block, which is covered by the connector operators, from the UI.
- [PC3r2.1] relates to the system of the solution, which is covered by all the features presented, such as the creation of workflows through the UI, or the orchestration and execution of workflows through Airflow.
- [PC2r14] is related to the deliver results block, which is done through the saving of the results gathered by the execution of the analytical models, using the connector operators to save those results into a database or file.
- The [PC2r10], which corresponds to streaming data, is also accomplished, through the addition of a KafkaStreamOperator that enables the consumption of Kafka streams and the creation of streaming-based AI workflows.

3.2 History

Version	Release date	New features
V 0.1	29/11/2022	Initial Airflow integration
V 0.2	07/01/2022	Backend logic
V 0.3	21/01/2022	First connectors
V 0.4	07/02/2022	Big data technologies integration
V 0.5	02/03/2022	ML operators
V 0.6	05/04/2022	Containerization of the components
V 1.0	02/05/2022	UI integration
V 1.1	09/05/2022	Bug fixes
V1.2	30/09/2022	User Management (Registration, Login, Password recovery)
V1.3	31/10/2022	React-based UI Mock-up ready

Version	Release date	New features
V1.4	15/11/2022	Several operators integrated (connectors, transformers, analytics), including streaming operators (Kafka Operator)
V 2.0	13/12/2022	Streaming workflows, React based UI, User Management

4. Conclusions

Deliverable D4.10 Services for Data Analytics is a technical specification document, providing an in-depth technical overview of the final release of the i4Q^{DA} solution. It describes in detail the role, the functionalities, and the conceptual architecture of i4Q^{DA}. The major features of the solution are detailed in order to clarify the fundamental functionalities and objectives of i4Q^{DA}. An extensive technical overview is presented, describing its architecture diagram with respect to i4Q Reference Architecture.

The final implementation status of i4Q^{DA} is detailed thoroughly, presenting the significant progress of its overall development.

Appendix I

The PDF version of the **i4Q Data Analytics** (i4Q^{DA}) web documentation can be accessed online at: **http://i4q.upv.es/10_i4Q_DA/index.html**