Stairway to AI: Ease the Engagement of Low-Tech users to the AI-on-Demand platform through AI, H2020

StairwAI AI Asset Management System - 2nd version

Deliverable information	
Deliverable number	D3.4
WP number and title	WP3 - Platform knowledge and community organisation
Lead beneficiary	UPC
Dissemination level	PU
Due date	31 December 2022
Actual date of delivery	30 December 2022
Author(s)	Miquel Buxons, Javier Vázquez-Salceda
Reviewer	Javier Farreres De La Morena

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101017142

Document Control Sheet

Version	Date	Summary of changes	Author(s)
0.1	30/11/2022	Updates to existing sections in the previous D3.3 deliverable	Miquel Buxons, Javier Vázquez-Salceda
0.2	01/12/2022	Installation procedure modification and format changes	Miquel Buxons
0.3	8/12/2022	Final version of the Executive Summary	Javier Vázquez-Salceda
0.4	12/12/2022	Final version of sections 1 and 2	Javier Vázquez-Salceda
0.5	24/12/2022	Internal Review Completed	Javier Farreras de la Morena
1.0	30/12/2022	Final version of the document	Miquel Buxons, Javier Vázquez-Salceda

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101017142

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Acronyms

Acronym	Explanation
ACID	Atomicity, Consistency, Isolation, and Durability (properties of data transactions)
ADMS	Asset Description Metadata Schema
AI	Artificial Intelligence
AMS	Asset Management System
API	Application Programming Interface
AQL	ArangoDB Query Language
DB	Database
DBMS	Database Management System
HW	Hardware
ISO	International Organization for Standardization
ML	Machine Learning
NMT	Neural Machine Translation
NLP	Natural Language Processing
NLU	Natural Language Understanding
OWL	Web Ontology Language W3C Specification Simple Knowledge Organization System
RDF	Resource Description Framework
SPARQL	W3C SPARQL Protocol and RDF Query Language
SQL	Standard (database) Query Language
StairwAI AMS	StairwAI Asset Management System
SW	Software
W3C	World Wide Web Consortium
WP	Work Package
XML	eXtensive Markup Language

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

1. Executive Summary

Deliverable D3.4 is produced within WP3 (Task 3.6) and corresponds to the release of the second prototype of the StairwAI AI Asset Management System (StairwAI AMS for short), one of the components of StairwAI's service platform. This StairwAI AMS should support the operation of the Horizontal Matchmaking in WP5, the Vertical Matchmaking in WP6, and the Multilingual interaction components in WP4.

The StairwAI AMS instantiates the *AI Assets Conceptual Semantic Model*, which was first reported in Deliverable D3.1 "Design of the knowledge representation in the StairwAI AI Asset Management System – 1st version" (released in M8) and then updated in Deliverable D3.2 "Design of the knowledge representation in the StairwAI AI Asset Management System – 2nd version" (released in M18).

This deliverable releases a second version of the StairwAI AI Asset Management System. It takes into account the data gathering made by WP3 in Task 3.3, Task 3.4, and Task 3.6 and the updated requirements gathered by WP2 at the end of the first Open Call (M18).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

2. Introduction

The aim of StairwAI is to enhance the AI-on-demand platform functionalities by means of a service layer that provides *Horizontal Matchmaking* (an AI-based automatic mapping between user requirements and assets of the AI-on-demand platform to meet end users' needs, developed in WP5) and *Vertical Matchmaking* (AI-based automatic mechanisms for hardware resource dimensioning and hardware resource provider discovery to satisfy end user needs, developed in WP6). The use of both services by users will be eased through a *Multilingual Virtual Assistant* (a chatbot based on Natural Language Processing techniques that will ease the interaction with the system in the native language of the user, developed in WP4). Figure 2.1 shows the planned interactions between the StairwAI services and the AI-on-demand platform.

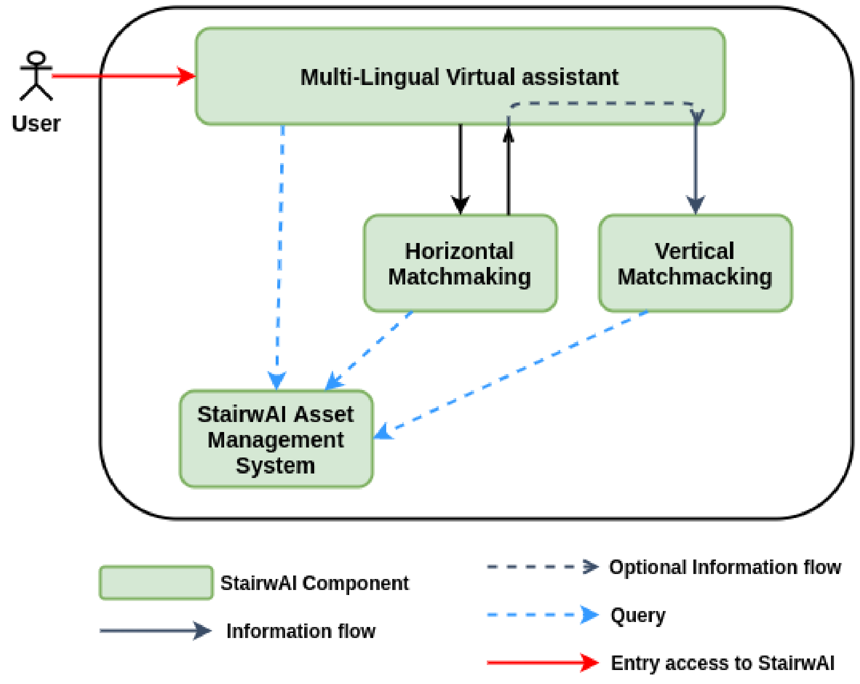

Figure 2.1 - Interactions between StairwAI's main components

The three core modules in the StairwAI system (Multilingual Virtual Assistant, the Horizontal Matchmaking module and the Vertical Matchmaking module) require the support of the *StairwAI Asset Management System* (StairwAI AMS for short). This StairwAI AMS is a semantically-enhanced information system to model all the assets (libraries, tools, algorithms, professionals, academic resources, hardware and cloud services, job positions, experts, etc) that are exchanged among them. It structures all the information about assets in the form of a knowledge graph, that is, a Semantic graph where entities are defined as class nodes fully interconnected by a main subclass property and that may be interconnected by extra domain-specific properties.

To ensure the semantic coherence of the knowledge graph database used by the StairwAI AMS and to ease the semantic alignment with other initiatives within the AI4EU ecosystem, this AMS follows the *AI Assets Conceptual Semantic Model* (defined in Deliverable D3.1 and extended in Deliverable D3.2).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

StairwAI AMS is built on the StairwAI ontology, which specifies the available classes, the relations between the classes and the attributes that each of these classes can have. Additionally, StairwAI ontology also specifies the instances for some non-instantiable classes, which can be considered as a close-set of options in which the instances of other classes can be related with.

The ontology establishes an immutable set of classes and relations. The user can declare instances of these classes. These instances must follow the structure of the ontology, that is, the instances must relate between them via instances of the relations specified in the ontology. Instances represent objects in the real world, and instanced relations describe their real world interactions.

For example, in figure 2.2 below, at the ontology level Class A relates with Class D via Relation 2. In the instance level, Instance A2 is an instance of Class A, Instance D1 is an instance of Class D, and both instances relate via an instanced Relation 2. The ontology describes a set of potentialities that must be instantiated in the instance level to represent the real world.

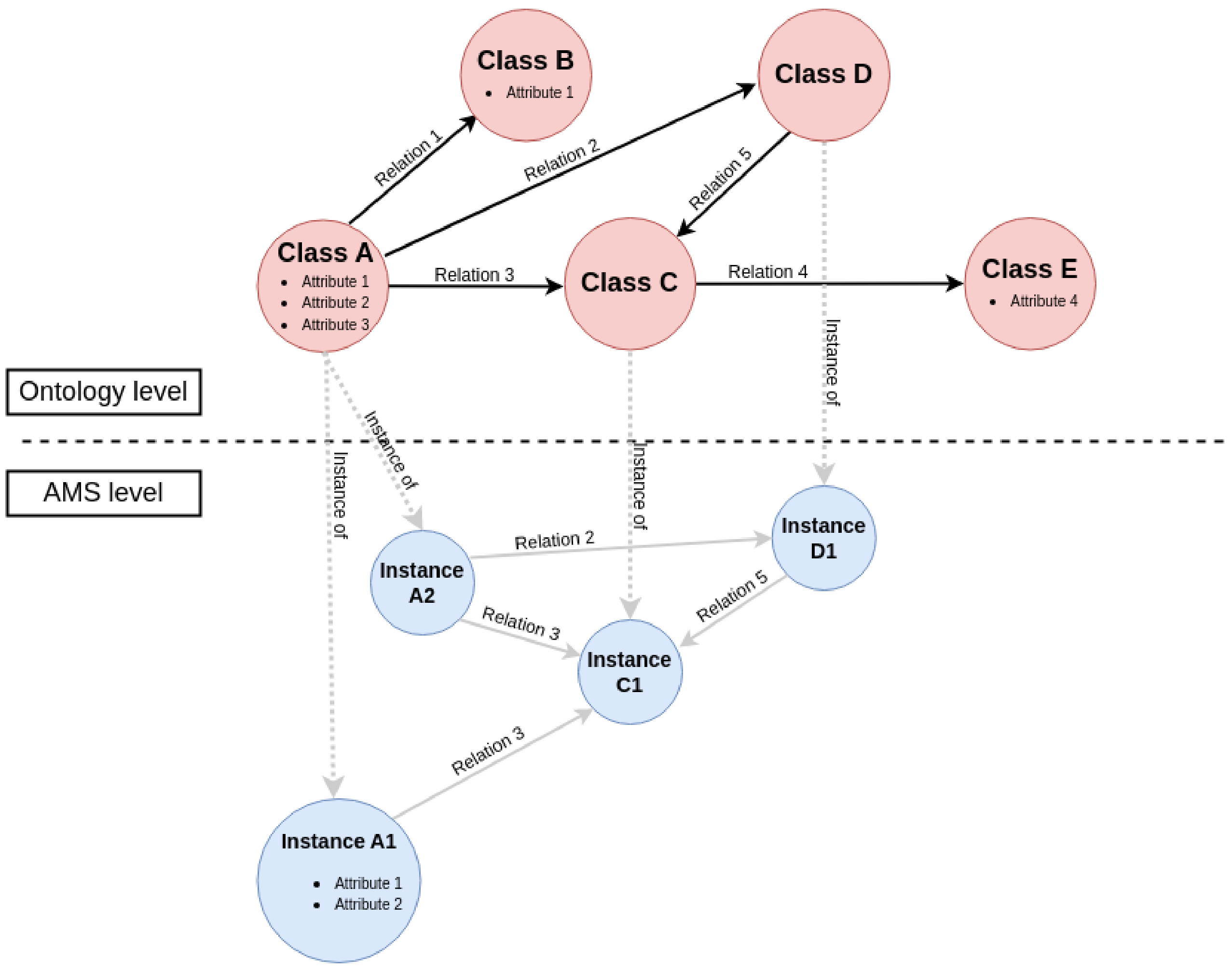

Figure 2.2 - StairwAI AMS conceptual idea.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

2.1. Purpose and Scope of the document

Deliverable D3.4 is a Demonstrator, in this case a set of software components that jointly form the second version of the *StairwAI Asset Management System*. These software components are available for download at StairwAI's GitLab (https://gitlab.com/stairwai/wp3/knowledge-representation). Instructions for download and installation are provided in the same repository.

This document is a companion to the Demonstrator. Its main purpose is to briefly describe the internal structure of the code. A copy of the installation instructions is also provided.

The second prototype of the *StairwAI Asset Management System* is based on the *AI Assets Conceptual Semantic Model* presented on Deliverable D3.2 "Design of the knowledge representation in the StairwAI AI Asset Management System – 2nd version" (published on June 30, 2022). This Conceptual Semantic Model is expected to evolve due to the ongoing interactions with other projects within the AI on Demand ecosystem as part of the AI Ontology Working Group initiative. The StairwAI AMS will be continuously updated with each change in the Conceptual Semantic Model.

2.2. Structure of the document

Section 3 describes the internal elements of the set of requirements for the *StairwAI AI Asset Management System.*

Section 4 provides a copy of the User installation guidelines, also available in the WP3 repository within StairwAI's GitLab.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

3. Description of the StairwAI AMS

This second prototype follows a similar structure as the first one. AMS offers three main capabilities, namely:

- The *getters* which will extract all the different static data in the database; inside getters' category, we can distinguish between (1) non-instantiable class getters, (2) general data getters (such as, general AMS classes, attributes or relations getters) and, (3) specific node information getters.
- The *setters, which will insert, modify or delete* each one of the AI Asset instances of the platform.
- The implemented *queries*.

The StairwAI AMS has been wrapped in two different distributions (both available in the GitLab repository):

- One distribution has been wrapped in a docker container with all the required dependencies already included. Therefore, it is quite easy to install in any machine following the guidelines in the repository. This distribution was thought to be run locally, enabling local testing of AMS capabilities.
- The server-side distribution was designed to be deployed in a distributed environment. It is based on a Graph database docker container, which starts the knowledge graph and opens two different ports (for http and bolt protocols) to allow the users to connect with it. Apart from the database container, this distribution has two different API, one for common users and one with extra administrator-level functionalities.

For each capability, the set of functions available are described in the sections below.

3.1. Getters

- **get_classes**

Property	**Value**
Description	Getter of the ontology defined classes.
Parameter *(type: boolean)*	**only_labels**: specifies whether the user only wants the instantiable class labels (True) or if the user wants the class definitions (False).
Return *(type: string list)*	A list of the available classes.

- **get_instantiable_classes**

Property	**Value**
Description	Getter of the ontology instantiable classes with their definitions.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Parameter *(type: boolean)*	**only_labels**: specifies whether the user only wants the instantiable class labels (True) or if the user wants the class definitions (False).
Return *(type: string list or dictionary)*	A list, or dictionary, of the available instantiable classes

- **get_attributes**

Property	**Value**
Description	Getter of the ontology defined attributes.
Parameter *(type: boolean)*	**with_conditions**: whether the method must return the domain of each property.
Return *(type: Pandas DT or string list)*	If with_conditions is True, a pandas data frame with the properties and its domain. Otherwise, it will return the list of available attributes.

- **get_relations**

Property	**Value**
Description	Getter of the ontology defined relations.
Parameter *(type: boolean)*	**with_conditions**: whether the method must return the domain/range of relation.
Return *(type: Pandas DT or string list)*	If with_conditions is True, a pandas' data frame with the relations and its domain/range. Otherwise, it will return the list of available relations.

- **get_node_data_to_dict**

Property	**Value**
Description	This query returns the node data as a dictionary.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Parameter *(type: py2neo.Node)*	**node**: py2neo.Node instance.
Return *(type: dictionary)*	Node data as a dictionary, if node = None, returns None.

- **get_class_properties**

Property	**Value**
Description	This method returns a dictionary with the properties, and its definitions, of a specific type of class.
Parameter *(type: string)*	**class_type**: the name (label) of the class from which to extract the properties.
Return *type: (dictionary)*	A dictionary with the properties available for a specific class and its requirements.

- **get_business_categories**

Property	**Value**
Description	Available business categories' getter.
Parameter *(type: boolean)*	**with_descriptions**: if True it returns the descriptions, additionally to the labels of the business categories as a dictionary.
Return *(type: string list)*	List of the available Business category instances.

- **get_aitechniques**

Property	**Value**
Description	Available AI Techniques' getter.
Parameter *(type: boolean)*	**with_descriptions**: if True it returns the descriptions, additionally to the labels of the business categories as a dictionary.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Return *(type: string list)*	List of the available AI Technique instances.

- **get_countries**

Property	**Value**
Description	Available countries' getter.
Parameter	None
Return *(type: string list)*	List of the available Country instances.

- **get_organisation_types**

Property	**Value**
Description	Available organisation types' getter.
Parameter	None
Return *(type: string list)*	List of the available Organisation type instances.

- **get_containers**

Property	**Value**
Description	Available containers' getter.
Parameter	None
Return *(type: string list)*	List of the available Container instances.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

- **get_distributions**

Property	**Value**
Description	Available distributions' getter.
Parameter	None
Return *(type: string list)*	List of the available Distribution instances.

- **get_paces**

Property	**Value**
Description	Available paces' getter.
Parameter	None
Return *(type: string list)*	List of the available Pace instances.

- **get_languages**

Property	**Value**
Description	Available languages' getter.
Parameter	None
Return *(type: string list)*	List of the available Language instances.

- **get_education_levels**

Property	**Value**
Description	Available education levels' getter.
Parameter	None

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Return *(type: string list)*	List of the available Education level instances.

- **get_education_types**

Property	**Value**
Description	Available education types' getter.
Parameter	None
Return *(type: string list)*	List of the available Education type instances.

- **get_skills**

Property	**Value**
Description	Available skills' getter.
Parameter	None
Return *(type: string list)*	List of the available Skill instances.

3.2. Setters

- **insert_instance**

Property	**Value**
Description	This function allows the insertion of new instances into the database. To check the set of available instantiable classes, please call *get_instantiable_classes()* function. Additionally, to know the available relations and attributes for the instances of a specific class, call *query_get_class_properties(class_type)* function.
Parameter *(type: string)*	**name**: the name of the instance that is going to be inserted.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Parameter *(type: string)*	**class_type**: The class of the inserted instance.
Parameter *(type: dictionary)*	****kwargs**: the instance specific parameters (relations or attributes) that are going to be modified in the AMS. In the name of the parameter (the key) white spaces ( ) must be replaced by underscores (_). The value should be the attribute content itself in the case of Attributes, and the ID of the related instance in the case of the Relations.

- **delete_instance**

Property	**Value**
Description	This function deletes the instance specified by its identifier.
Parameter *(type: integer)*	**id**: identifier of the instance that has to be deleted.
Return	None

- **modify_instance**

Property	**Value**
Description	This function allows the modification of any instance node in the database. To check the available instantiable classes, call *get_instantiable_classes()* function. Additionally, to check the available relations and attributes for the instances of a specific class, call *query_get_class_properties(class_type)* function.
Parameter *(type: integer)*	**id**: identifier of the instance which is to be modified.
Parameter *(type: boolean)*	**delete_property**: whether the attribute or a relation is to be deleted (True), or inserted (False).
Parameter *(type:dictionary)*	****kwargs**: the instance specific parameters (relations or attributes) that are to be modified in the AMS. The name of the parameter (the key) white spaces ( ) are substituted by underscores (_). The value should be the attribute content itself in the case of Attributes, and the ID of the related instance in the case of the Relations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Return *(type:dictionary)*	The dictionary with the node data modified.

- **modify_score**

Property	**Value**
Description	Insert, or delete, a score in the AMS.
Parameter *(type: integer)*	**start_node_id**: Identifier of the instance (node) from which the new score is to be set.
Parameter *(type: float)*	**score**: The score value (float number).
Parameter *(type: integer)*	**end_node_id**: identifier of the node to connect.
Parameter *(type: boolean)*	**delete_property**: whether to delete attributes or relations (True), or to insert new ones (False).
Return	None

- **score_list_modifier**

Property	**Value**
Description	Insert a list of scores in the AMS using a list of dictionaries.
Parameter *(type: dictionary list)*	**score_list**: list of dictionaries, where each dictionary is a score insertion, from each dictionary, it is required the fields of 'start_node_id' (with the ID of the relation starting instance), 'end_node_id' (with the ID of the relation ending instance) and 'score' (field that specify the score value).
Return	None

3.3. Queries

- **free_query**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Property	**Value**
Description	This query allows a personalized Cypher query construction.
Parameter *(type: string)*	**query**: string with the Cypher code command (query) that is going to be used.
Return *(type: unknown)*	The result of the query, without post-processing.

- **query_exist_instance**

Property	**Value**
Description	This query returns true if a specific instance exists in the database.
Parameter *(type: integer)*	**id**: ID of the instance searched.
Return *(type: boolean)*	True if the instance identified by id exists in the AMS, False otherwise.

- **query_if_relation_exists**

Property	**Value**
Description	This query returns if a specific relation exists in the AMS.
Parameter *(type: integer)*	**start_node_id**: identifier of the starting instance in the relation (all the relations in the AMS are directed).
Parameter *(type: string)*	**relation**: relation to be searched.
Parameter *(type: integer)*	**end_node_id**: identifier of the ending instance in the relation (all the relations in the AMS are directed).
Return *(type: boolean)*	A boolean with the result of the query.

- **query_node_by_name_class**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Property	**Value**
Description	This query returns an instance, if it exists, based on its name and the class.
Parameter *(type: string)*	**name**: name of the instance searched.
Parameter *(type: string)*	**ont_class**: class of the instance searched.
Parameter *(type: boolean)*	**to_dict**: whether the result must be returned as a dictionary (True) or apy2neo.Node instance (False).
Return *(type: dictionary or py2neo.Node)*	Instance data or instance node, if it exists.

- **query_node_by_name**

Property	**Value**
Description	This query returns an instance, if it exists, given its name. WARNING: DO NOT USE UNLESS IT IS ABSOLUTELY NECESSARY, very slow function with exponential time cost. Alternatively, use *query_node_by_name_class* that uses the instance class also.
Parameter *(type: string)*	**name**: name of the instance searched.
Parameter *(type: boolean)*	**to_dict**: whether to return the node data as a dictionary (True) or a py2neo.Node instance (False).
Return *(type: dictionary or py2neo.Node)*	Instance data or instance node, if it exists.

- **query_node_by_id**

Property	**Value**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Description	This query returns the node, if it exists, given its identifier.
Parameter *(type: integer)*	**id**: id of the instance searched.
Parameter *(type: boolean)*	**to_dict**: whether to return the node data as a dictionary (True) or a py2neo.Node instance (False).
Return *(type: dictionary or py2neo.Node)*	Instance data or instance node, if it exists.

- **query_instances_by_class**

Property	**Value**
Description	This query returns all the instances of a specific class.
Parameter *(type: string)*	**class_label**: class name.
Parameter *(type: boolean)*	**with_inference**: in the case of True value in this parameter, the query will retrieve the instances of the class specified as a parameter and all the instances of its subclasses also.
Parameter *(type: boolean)*	**to_dict**: whether to return the node data as a dictionary (True) or a py2neo.Node instance (False).
Return *(type: dictionary list or py2neo.Node list)*	List of the instance data or instance nodes, if they exist.

- **query_class_by_id**

Property	**Value**
Description	This query returns the class label of an instance, given an identifier.
Parameter *(type: integer)*	**id**: id of the instance searched.
Return	The class label of the instance with the id.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

(type: string)	

- **query_related_nodes_by_relation**

Property	**Value**
Description	This query returns the data of the instances related to the given identifier and relation.
Parameter *(type: integer)*	**id**: id of the instance searched.
Parameter *(type: string)*	**relation**: relation to be searched.
Parameter *(type: boolean)*	**to_dict**: whether to return the node data as a dictionary (True) or a py2neo.Node instance (False).
Return *(type: dictionary list or py2neo.Node list)*	List of the instance data, or instance nodes, if they exist.

- **query_get_assets_by_tag**

Property	**Value**
Description	This query returns a list of AI Artifacts that belongs to a specific AI Technique.
Parameter *(type: string list)*	**categories**: list of AI Techniques from which to retrieve the instances.
Parameter *(type: integer)*	**limit**: limit of nodes to be retrieved by the query.
Parameter *(type: boolean)*	**to_dict**: whether to return the node data as a dictionary (True) or a py2neo.Node instance (False).
Parameter *(type: boolean)*	**inference_cat**: whether to infer the AI Techniques subcategories of the ones in categories parameter.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Return *(type: dictionary list)*	A list of dictionaries where, for each category searched, a dictionary is retrieved with the artifact and the category that belongs to.

- **query_get_experts_by_tag**

Property	**Value**
Description	This query returns a list of AI Experts that belongs to a specific AI Technique.
Parameter *(type: string list)*	**categories**: list of AI Techniques from which to retrieve the instances.
Parameter *(type: integer)*	**limit**: limit of nodes to be retrieved by the query.
Parameter *(type: boolean)*	**to_dict**: whether to return the node data as a dictionary (True) or a py2neo.Node instance (False).
Parameter *(type: boolean)*	**inference_cat**: whether to infer the AI Techniques subcategories of the ones in categories parameter.
Return *(type: dictionary list)*	A list of dictionaries where, for each category searched, a dictionary is retrieved with the expert and the category that belongs to.

- **query_topk_assets**

Property	**Value**
Description	This query returns an ordered asset list with, at most, K elements based on the scored assets in AI Technique.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

Parameter *(type: string)*	**category**: AI Technique from which the assets will be retrieved.
Parameter *(type: integer)*	**k**: the maximum number of elements retrieved.
Parameter *(type: boolean)*	**to_dict**: whether to return the node data as a dictionary (True) or a py2neo.Node instance (False).
Return *(type: dictionary list or py2neo.Node list)*	An ordered list of assets (dictionaries{score, instance}).

3.4. Administrator extra functionalities

- **var_file_init**

Property	**Value**
Description	This method creates from scratch the model data-structures of the AMS using the StairwAI Ontology structure imported. Finally, store it in a JSON file.
Parameter *(type: boolean)*	**mod_file**: whether the modification in the fixed data structures will be written in the JSON file.
Parameter *(type: string)*	**path**: path to the JSON file where the fixed data structures are written.
Return	None

- **create_backup**

Property	**Value**
Description	This method creates a StairwAI AMS backup, with the information extracted from the nodes and relations it creates two pickle files where these data are stored. Use 'read_backup()' function in order to store again the files' information to the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

	database.
Parameter	None
Return	None

- **read_backup**

Property	**Value**
Description	This method read the pickle files with the AMS instances backup generated by the method create_backup(), and load the instances in the AMS. Additionally, in deletes the pickle files.
Parameter	None
Return	None

- **hard_db_reset**

Property	**Value**
Description	Hard reset function, this functionality allows the possibility to reset the AMS, e.g. in the case of StairwAI ontology update. This function automatically stores all the inserted instances in a backup file, restarts the AMS, imports the latest StairwAI ontology available, makes the changes to the *swai_ams.json* file and finally incorporates the instances available in the backup. WARNING: currently, an AMS reset will change all the node identifiers!
Parameter *(type: boolean)*	**with_backup**: Boolean parameter, if it is set to true a backup of the AMS instances is done before the reset, after it the instances are loaded to the AMS again. If it is set to False, the AMS INSTANCES WILL BE COMPLETELY AND PERMANENTLY DELETED.
Return	None

4. How to install the StairwAI Asset Management System

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142

This section explains the requirements and steps to install the local version of the StairwAI Asset Management System. This version of the AMS has been wrapped into a docker container, which handles all the dependences of the AMS internally.

4.1. Requirements

Basic system requirements:

- PyCharm (or any other IDE with the ability to run and modify usage_example.py python code).
- Docker and docker-compose installed (version 2.2.2 or greater).

4.2. Steps to install and run the AMS

In order to run the current prototype of the AMS, please follow these few steps:

1. Download the GitLab repository of the AMS link
2. Go to the localhost-version folder
3. Using a terminal inside the folder, run command ***docker-compose up*** (if it requires admin permissions, then run ***sudo docker-compose up***)
    a. Now the docker is running, and the database is initialized, with all the required ontologies inserted and with some testing AI Artifact instances inserted (extracted from AI4EU old platform assets)
4. To make queries, get some inserted information or insert new assets (the available ones) python functions with the desired capability must be called.
    a. Some usage examples are found in the usage_example.py script.
5. To close the docker, apply ***Ctrl + C*** to the running docker terminal and ***docker-compose down*** in another terminal.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017142