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

StairwAI AI Asset Management System - 1st version

Deliverable information	
Deliverable number	D3.3
WP number and title	WP3 - Platform knowledge and community organisation
Lead beneficiary	UPC
Dissemination level	PU
Due date	31 December 2021
Actual date of delivery	29 December 2021
Author(s)	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

Document Control Sheet

Version	Date	Summary of changes	Author(s)
0.1	12/11/2021	First Table of Contents	Miquel Buxons, Javier Vázquez-Salceda
0.2	03/12/2021	First version of sections 3 and 4.	Miquel Buxons, Javier Vázquez-Salceda
0.3	10/12/2021	First version of Executive Summary and Annex A	Miquel Buxons, Javier Vázquez-Salceda
0.4	17/12/2021	First Version of Section 2. Section 3 and Annex A completed.	Miquel Buxons, Javier Vázquez-Salceda
0.5	21/12/2021	Executive Summary, Section 2 and Section 4 completed.	Miquel Buxons, Javier Vázquez-Salceda
	23/12/2021	Internal review completed	Cristian Barrué
1.0	28/12/2021	Final versión of the document	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

1. Executive Summary 5
2. Introduction 6
2.1. Purpose and Scope of the document 7
2.2. Structure of the document 7
3. Description of the StairwAI AMS prototype 8
3.1. Getters of concept instances 8
3.2. Insert, delete and modify functions 8
3.3. Queries 9
4. How to install the StairwAI Asset Management System 10
4.1. 10
5.2. Steps to install and run the AMS 10
Annex A: Analysis of candidate Graph Databases 11
A.1. Description of candidates 11
A.1.1. Arango DB 11
A.1.2. Neo4j 12
A.1.3. Graph DB 14
A.1.4. MarkLogic 15
A.2. Comparative analysis 16
A.2.1. Criteria for comparison 16
A.2.2. Comparison Table 17

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 D3.3 is produced within WP3 (Task 3.6) and corresponds to the release of the first 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 Multi-lingual interaction components in WP4.

The StairwAI AMS instantiates the *AI Assets Conceptual Semantic Model*, which has been reported in Deliverable D3.1 "Design of the knowledge representation in the StairwAI AI Asset Management System – 1st version" released in M8.

This deliverable releases a first version of the StairwAI AI Asset Management System, prior to the actual data gathering made by WP3 in Task 3.3, Task 3.4, and Task 3.6 and to the updated requirements gathered by WP2 at the end of the first Open Call (M18). An updated version of the knowledge representation, integrating the lessons learned from the data gathering and the interactions with related EU initiatives, will be provided in D3.2. "Design of the knowledge representation in the StairwAI AI Asset Management System - 2nd version", which will be delivered in M18, and will impact in the next version of the StairwAI AMS to be reported in D3.4 in M24.

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

2. Introduction

The main aim of StairwAI is to enhance the AI-on-demand platform services 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) and *Vertical Matchmaking* (AI-based automatic mechanisms for hardware resource dimensioning and hardware resource provider discovery to satisfy end user needs). The use of both services by users will be eased through a *Multi-Lingual 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). Figure 2.1 shows the planned interactions between the StairwAI services and the AI-on-demand platform.

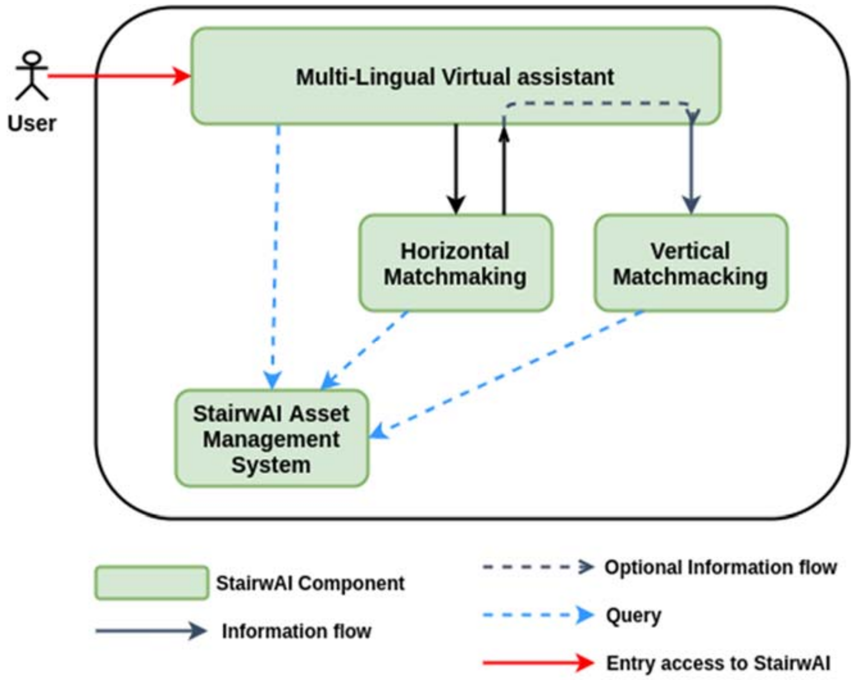

Figure 2.1 - Interactions between StairwAI's main components

The three core modules in the StairwAI system (Multi-Lingual 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 information about assets in the form of a knowledge graph, that is, a Semantic Web graph where entities are defined as class hierarchies 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).

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 D3.3 is a Demonstrator, in this case a set of software components that jointly form the first 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. Furthermore the interested reader can find in Annex A a summary of the analysis made to the candidate Graph Databases to select the one to be used as the core of the StairwAI AMS.

The first prototype of the *StairwAI Asset Management System* is based on the *AI Assets Conceptual Semantic Model* presented on Deliverable D3.1 "Design of the knowledge representation in the StairwAI AI Asset Management System – 1st version" (published on August 31, 2021). This Conceptual Semantic Model is expected to evolve due to the data gathering made by WP3 in Task 3.3, Task 3.4, and Task 3.6, to the updated requirements gathered by WP2 at the end of the first Open Call (M18) and to the interactions with other projects within the AI4EU ecosystem. The StairwAI AMS will be continuously updated with each change in the Conceptual Semantic Model. An updated version of this Semantic Model will be reported in Deliverable D3.2. "Design of the knowledge representation in the StairwAI AI Asset Management System - 2nd version", which will be delivered in M18. The next version of the StairwAI AMS will be released in Deliverable D3.4 "StairwAI AI Asset Management System - 2nd version", which is due in M24.

2.2. Structure of the document

Section 3 describes the internal elements of the initial set of requirements on 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.

Annex A reports the analysis made to the candidate Graph Databases to select the one to be used as the core of the *StairwAI AI Asset Management System.*

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 prototype

In this first prototype of the StairwAI AMS, we already implement the basic structure of it. The design chosen includes a general class Driver that hides the detail of the underlying graph database to the other StairwAI components. To do so the Driver is able to call the methods done by two subclasses. These subclasses will be the specific drivers for each one of the chosen graph database platforms (Neo4j and Graph DB, see Annex A). In this first version the Neo4j specific driver is implemented. Implementing the Graph DB driver only requires adapting the same Neo4j driver functions to the Graph DB IDE.

The main Driver is defined by three main capabilities: the *getters* which will extract all the different static data in the database (static in terms of the assets that the normal user is not able to insert, modify or delete), the *methods to insert, modify or delete* each one of the AI Assets of the platform, and finally, the implemented *queries*. In this first version, we implement a subset of each capability, in order to provide a prototype that let the developers of the other components in the StairwAI platform to start running tests with the AMS.

This version of the AMS has been wrapped in a docker container with all the dependences required. Therefore, it is quite easy to install in any machine following guidelines in the repository (also available in Section 4 of this document). In the same GitLab repository, where the AMS' docker is placed, some documents with extra information can be found in the wiki section, such as an Excel table file with the available relations and properties of each class.

3.1. Getters of concept instances

The getters of the concept instances have been already created for the classes that cannot be instantiated by a user in the AMS and general AMS key parts:

- Classes (current classes in the AMS).
- Properties available in the AMS, with its domain.
- Relations available in the AMS, with its domain and range.
- Business area (key field in AI4EU, used in AMS)
- Countries available in the AMS (e.g. the function to get the Countries is *get_countries()*)
- Organisation types
- Container instances
- Distribution instances

3.2. Insert, delete and modify functions

In this version, the class nodes that can be inserted, modified and deleted are:

- Agent's subclasses (Agent class is not instantiable):
    - Persons (e.g. *insert_person("Example")*)
    - Organisations

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

- AI Artifacts subclasses (Agent class is not instantiable):
    - Models
    - Libraries
    - Algorithms
    - Tools (e.g. *delete_tool("Example")*)
- Datasets

Each class has its allowed properties and relations to other class type nodes (the relations are the ones specified by the ontology):

- The list of the available properties and relations can be checked in the *class_attributes_and_relations.ods* file, placed in the wiki section of the AMS GitLab repository (GREEN for properties, BLUE for available relations and YELLOW if the class has inherited properties and/or relations).
- The domain classes for each property, and the domain and range classes for each relation, can be obtained using some Getters of the AMS.
- The current relations and properties are clearly biased by the concepts used to define Assets in AI4EU. These properties and relations can be modified, or extended, on demand in order to adapt to the concepts used to define Assets in other platforms.

3.3. Queries

In this first prototype 4 different basic queries are implemented, in order to allow the other modules' developers to make basic queries to the AMS.

- query_exist_instance: this query returns a Boolean set to True if a node exists, using its name and class of it.
- query_instances_class: this query gets all the instances of a class, and its subclasses if inference parameter is set to True.
- query_related_nodes: this query returns the nodes related to another one, using the node name, class and the specific relation.
- free_query: this query allows making a personalized query to the AMS (currently only in CYPHER language, the viability for a SPARQL adaptation will be analysed in the future).
- query_node_by_name_class: this query returns a node, with all its properties and relations, using its name and class.

This list will be extended, adapting it to the needs of the other StairwAI platform components developed in WP4, WP5 and WP6.

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

4. How to install the StairwAI Asset Management System

This section explains the requirements and steps to install the first version of the StairwAI Asset Management System. The AMS prototype has been wrapped into a docker container which handles all the dependences of the AMS internally. The current version of the AMS only supports connection with Neo4j databases.[1]

4.1. Requirements

Basic system requirements:

- PyCharm (or any other IDE that allows you to run and modify main.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, you have to follow these few steps:

1. Download the GitLab repository of the AMS link
2. Using a terminal inside the downloaded folder, apply the command ***docker-compose up*** (it might require admin permissions, then write ***sudo docker-compose up***)
   a. Now the docker is running, and the database is initialized, with all the required ontologies inserted and with the AI4EU old platform assets inserted
3. If you want to make queries, get some inserted information or insert new assets (the available ones) you have to edit the main.py file using a python IDE and run it.
   a. Some examples of possibilities of the current version can be found in sections 4.1, 4.2 and 4.3.
4. In the case that you want to close the docker, apply ***Ctrl + C*** to the running docker terminal and ***sudo docker-compose down*** in other terminal.

Consensus on the modelling of some AI Assets included in our model. As these consensuses are reached, we will include them in the *AI Assets Conceptual Semantic Model.*

[1] In the future, we will implement a version for GraphDB. For any of the DB types (Neo4j or GraphDB) the interaction user-AMS will be exactly the same, the connection will be done internally in the AMS, supporting the same queries and capabilities in both cases.

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

Annex A: Analysis of candidate Graph Databases

This annex reports the analysis made to the four candidate technologies that were considered to build the StairwAI AMS. Different aspects of the candidates' platforms were analysed, not only related to technical properties, but also to management and maintenance properties, that would become as important as technical ones in the future of the project.

A.1 Description of candidates

In this section, the four candidates that were analysed are briefly described (purpose, main characteristics), but no comparison is provided among them. The comparative analysis is provided in section A.2.

A.1.1 Arango DB

Arango DB is a native multimodel DBMS for graph, document, key/value and search. Arango DB allows to build high-performance applications using a SQL-like query language or JavaScript extensions over a NoSQL (i.e., Non-Relational) data model. This platform allows to use ACID transactions (i.e., transactions with the atomicity, consistency, isolation, and durability properties typical of SQL databases) when required. But it can scale databases horizontally and vertically easily.

Key features of Arango DB:

- Easy installation on a cluster.
- Arango DB provides a multimodel DBMS (a combination of key-value pairs, documents or graphs), the generated databases could be a combination of these schemas providing flexible data modelling to the final user.
- Usage of AQL (Arango DB Query Language).
- Arango DB can be used as an application server, and therefore the application and database can be fused together for maximal throughput.
- Applied to transactions, this platform permits optional transactional consistency and isolation.
- Set up the created database in a Leader/Follower configuration, or spread datasets across multiple servers.
- Configurable durability.
- Open-source platform.

To import an ontology to ArangoDB, it is necessary to translate RDF data to JSON (hints should be stored in another file if later the user wants to export the data from ArangoDB into RDF format). This database has community developed functions that allow the transformation between RDF and JSON format and, in addition, members of the community created an extension that allows to translate queries in the SPARQL standard to AQL query format and run them.

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

This database offers limited options to make inferences and validations into ontologies. For that the creators recommend the usage of an external tool called Foxx.

This company offers several extra functionalities that users can use in any situation, related to:

- Search (e.g. exists, in_range, ngram_match, levenshtein_match, etc.)
- Arrays (e.g. Jaccard distance, push, count, contains, sorted, union, etc.)
- Bit treatment (e.g. bit_and, bit_or, bit_xor, shift_left, bit_from_string, etc.)
- Dates (e.g. day, year, second, isoweek, timestamp, date_compare, etc.)
- Document (e.g. attributes, count, has, merge, translate, values, zip, etc.)
- Geo (e.g. distance, geo_distance, intersect, geo_point, near, within, etc.)
- Numeric (e.g. average, degrees, rand, stddev, tan, etc.)
- Strings (e.g. json_parse, count, find_last, split, to_hex, etc.)

It also offers utilities more specific to graph representations:

- Traversals (create a sub-graph using an initial node and a range of nodes that the algorithm is able to search)
- Shortest path
- K path

To end with the analysis of ArangoDB, this platform has a big community that creates new functionalities to the platform and solve the beginners' doubts. The company is located in US, and its implementation of a NoSQL database is well-considered overall the community.

A.1.2 Neo4j

Neo4j is a graph database management system developed by Neo4j, Inc. Described by its developers as an ACID-compliant transactional database with native graph storage and processing, Neo4j is available in a GPL3-licensed open-source "community edition", with online backup and high availability extensions licensed under a closed-source commercial licence. Neo4j's company also licences Neo4j with these extensions under closed-source commercial terms.

Neo4j is implemented in Java and accessible from software written in other languages using the Cypher query language through a transactional HTTP endpoint, or through the binary "bolt" protocol.

Neo4j neosemantics provides a set of 4 methods that allow the user to create their own inference, this inference is not automatic like in GraphDB and MarkLogic, but the user is able to run, and scale to which nodes apply, the following inference methods:

- nodesLabelled
- hasLabel
- nodesInCategory
- inCategory

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

Neo4j offers operations that can help in a semantic analysis of the graph, such as, similarity metrics as Node Similarity, Jaccard Similarity, Pearson Similarity. The Neo4j community has created some plugins that can be added to any Neo4j version that can help the user to semantic analysis, such as GraphAware Natural Language plugin[2]. In addition, the Neosemantics plugin has a function to import SHACL constrains files[3], in order to apply a validation in the full graph or only in a part of it. This possibility would be extremely useful in our case, in order to validate a new node created without having to validate the entire graph.

Neo4j offers a complete set of APIs that allow the usage of some programming language to interact with the graph database, in our case, Python is one of the most used programming language with plenty of extra plugins and resources. Neo4j offers a huge amount of free plugins that would allow us to be very efficient during the creation and maintenance phases of the project, with well documentation. The main three plugins would be:

- Neosemantics: basic, in our case, is a plugin that enables the use of RDF in Neo4j.
- Graph Data Science Library: provides extensive analytical capabilities centred around graph algorithms. The library includes algorithms for community detection, centrality, node similarity, path finding, and link prediction, as well as graph catalogue procedures designed to support data science workflows and machine learning tasks over your graphs.
- APOC: consists of several (about 450) procedures and functions to help with many tasks in areas like data integration, graph algorithms or data conversion.

Neo4j is developed by Neo4j, Inc., based in the San Francisco Bay Area, United States, and also in Malmö, Sweden. This platform has a huge and active community with several forums and discussions. The company offers mainly two products[4]:

- Aura: Neo4j Aura offers flexible plans for your fully managed Neo4j graph database service. This product has 3 different pricing plans.
  - Aura free
  - Aura professional
  - Aura enterprise
- Self-hosted Neo4j: run Neo4j on-premises, in your private cloud, or public cloud infrastructure. This product has 2 different plans:
  - Community edition
  - Enterprise edition

[2] graphaware/neo4j-nlp: NLP Capabilities in Neo4j

[3] SHACL (Shapes Constraint Language) is a language for validating RDF graphs against a set of conditions.

[4] Cloud & Self-Hosted Graph Database Platform Pricing

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

A.1.3 Graph DB

Ontotext GraphDB is a highly efficient and robust graph database with RDF and SPARQL support. This database supports a highly available replication cluster, which has been proven in a number of enterprise use cases that required resilience in data loading and query answering.

GraphDB uses RDF4J as a library, utilizing its APIs for storage and querying, as well as the support for a wide variety of query languages (e.g., SPARQL and SeRQL) and RDF syntaxes (e.g., RDF/XML, N3, Turtle).

GraphDB supports inference out of the box and provides updates to inferred facts automatically. Facts change all the time, and the number of resources it would take to manually manage updates or rerun the inferencing process would be overwhelming without this capability.

Inference uncovers the full power of data modelling with RDF(S) and ontologies. GraphDB will use the data and the rules to infer more facts and thus produce a richer dataset than the one you started with. GraphDB can be configured via “rulesets” (sets of axiomatic triples and entailment rules) that determine the applied semantics. This platform provides a set of standard rulesets but, as a user, can customize our own ruleset.

GraphDB offers Plugins in order to increase its own functionalities, such as:

- Plugin API: a framework and a set of public classes and interfaces that allow developers to extend GraphDB in many ways.
- RDF Rank: an algorithm that identifies the more important or more popular entities in the repository by examining their interconnectedness.
- Semantic similarity searches: exploring and searching semantic similarity in RDF resources.
- JavaScript functions: defining and executing JavaScript code, further enhancing data manipulation with SPARQL.
- Change tracking: tracking changes within the context of a transaction, identified by a unique ID.
- Provenance: generation of inference closure from a specific named graph at query time.
- Proof plugin: finding out how a given statement has been derived by the inferencer.
- Graph path search: exploring complex relationships between resources.

Several of the plugins are user-defined indexes. They are created with SPARQL:

- Autocomplete index: suggestions for the IRIs` local names in the SPARQL editor and the View Resource page.
- GeoSPARQL support: the plugin allows the conversion of Well-Known Text from different coordinate reference systems (CRS) into the CRS84 format, which is the default CRS according to the OGC.
- Geospatial extensions: support of 2-dimensional geospatial data that uses the WGS84 Geo Positioning RDF vocabulary (World Geodetic System, 1984).
- Data history and versioning: accessing past states of your database through versioning of the RDF data model level.
- Text mining plugin: calling of text mining algorithms and generation of new relationships between entities.

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

GraphBD provides a complete SHACL validation procedure in which the user can validate the data inserted in the database using shapes graphs. The validation is a valuable tool for efficient data consistency checking, The protocol validates RDF graphs against a set of conditions. These conditions are provided as shapes and other constructs expressed in the form of an RDF graph. In SHACL, RDF graphs that are used in this manner are called shapes graphs, and the RDF graphs that are validated against a shapes graph are called data graphs.

This database is a product created by Ontotext, this company started in 2000 as a R&D lab within Sirma Group, a top 3 Bulgarian software company, Ontotext's technology and solutions are spread wide across the value chain of the most knowledge-intensive enterprises. It has 3 main products:

- GraphDB Free: it is a version free to use with limited to two queries in parallel.
- GraphDB Standard Edition: this version offers unlimited parallel queries.
- GraphDB Enterprise Edition: offers the same as GraphDB Standard Edition, adding more scalability and resilience; Elastic search and Solr connectors.

Ontotext has a small but active community behind it, It has a Stack Overflow forum with several discussions with doubts solved and open discussions.

A.1.4 MarkLogic

MarkLogic Server is a multimodel database that has both NoSQL and trusted enterprise data management capabilities. It is the most secure multimodel database, and it is deployable in any environment.

MarkLogic Server natively stores JSON, XML, text, geospatial, and semantic triples in one unified platform. This ability to store and query document data, location data, semantic graph data, and build relational views for SQL analytics results in unprecedented flexibility and agility when integrating data from silos.

MarkLogic provides support for document stores (using JSON/XML), Semantic Data storage (using RDF Triples and SPARQL) which is ideal for storing relationships, which is useful for managing metadata (e.g., ontologies), improving data integration, and building applications with highly connected data. Finally, MarkLogic supports Relational storage (using SQL and relational tables).

This platform provides support for inference based on rulesets predefined by them or customized by the user with the requirements. The databases can be managed using Python or any other programming language. RDF new triples cannot be validated using SHACL protocol, Marklogic provides functions in order to validate the queries. The platform provides several types of extra functionalities and methods that help the user to process the extracted DBMS data, connect with an interface, apply some machine learning algorithms, but during the documentation of this file, we could not find any reference to semantic operations directly to RDF schema. The platform offers semantic similarities between documents and words but is applied to only the document database side.

The company was created in the 2000s in the United States of America but it is present in the international market, having offices in North America, Europe and Asia. This platform has not a very active community in forums, but some of them could be found during the writing of this document.

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

A.2 Comparative analysis

In this section we provide a comparison between the different proposed platforms to develop the knowledge graph using a set of criteria that is explained below.

A.2.1 Criteria for comparison

Ontology support

This criterion evaluates the capacity to handle an RDF file, storing the relationships exposed in the ontology, and the capacity to query in this generated schema.

Query language

The language used by a platform to make queries to the previously updated RDF schema.

Inference capacity

The capacity to automatically discover new facts based on a combination of data and rules for understanding that data. Inference with semantic triples means that automatic procedures can generate new relationships (new facts) from existing triples.

Semantic operations

The capacity to apply semantic knowledge and operations into the results obtained from a query and inside the query (semantic distance).

Python support

The ability to use an API to handle the queries and collect the results in Python.

SHACL validation

The capability to use W3C standard Shapes Constraint Language (SHACL) validation for efficient data consistency checking. The language validates RDF graphs against a set of conditions. These conditions are provided as shapes and other constructs expressed in the form of an RDF graph. In SHACL, RDF graphs that are used in this manner are called shapes graphs.

Lecture/writing time cost

This criterion evaluates the time cost that a platform needs in order to make a lecture in the database or write a new Triple in it.

Extra implemented methods

This criterion evaluates the proposed platforms in terms of extra libraries that this platform provides in order to speed up the development of the knowledge graph and the possible queries and semantic inferences that could be applied.

Community support

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

This criterion evaluates the community behind a platform and the documentation availability. When you start a project development in a platform, it is important that this one has good and deep documentation of the capabilities and bugs that this platform provides. In the same direction, if a platform has a big, active and kind community, could be crucial in order to solve problems that may appear during the development.

Company origin

This criterion describes the company origin and its servers' location. This criterion aims to promote the European initiative and avoid possible conflict of server location, in context that we are a European project.

Licences

This criterion aims to evaluate the licences that each platform offers, in order to avoid legal issues in the further development. The aspect is highly dependent with the following one.

Pricing

This criterion aims to evaluate each platform usage possibilities that StairwAI should consider. In order to find the plan that fits better with the project objectives and needs.

A.2.2 Comparison Table

The following table summarises the results of our comparative analysis

Criteria \ DB	*Arango DB*	*Neo4j*	*graph DB*	*MarkLogic*
Ontology support	No RDF support directly	Yes, RDF support using neosemantics	Yes, RDF stores	Yes, RDF stores
Query language	AQL	Cypher	SPARQL	SPARQL
Inference capacity	None	Yes, using methods	Yes, using rule sets	Yes, using rule sets
Semantic operations	No	Yes	Yes	Yes, but for documents
Python support	Yes	Yes	Yes	Yes
SHACL validation	No	Yes	Yes	No
Extra implemented	Yes	Yes	Yes	Yes

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

methods				
Community support	High	High	Medium	Medium
Company origin	US	Europe	Europe (Bulgaria)	US
Licences	Apache Licence 2.0	Neo4j community edition is fully open source (GPL v3) Enterprise edition (Commercial licence, developer licence and evaluation licence)	Depends on the product	For MarkLogic server: Developer Licence(Free) Enterprise Licence (Payment)
Pricing	Free community edition Enterprise edition (pay) Oasis (cloud) edition(pay for use)	Neo4j Aura (cloud) pay to use Community edition(free) Enterprise edition (Depending, developer or company)	GraphDB Free GraphDB Standard (payment) GraphDB Enterprise (payment)	Cloud server(Pay to use) MarkLogic server

After analysing all the previously mentioned aspects for each of the proposed platforms, we conclude that Neo4j and GraphDB excel the other two options because they ae the ones that satisfy the needs of the StairwAI project, sometimes adding interesting extra functionalities (e.g. Neo4j inference functions.

ArangoDB was discarded because it has not an easy integration with RDF (ontologies), no inference capacity, and in our tests we did not find the capability of executing semantic operations on it.

In the case of MarkLogic, it was discarded due to its lack of support to SHACL validation, its focus on commercial purposes, and weak community support.

In the case of Neo4j and GraphDB, apart from being the ones better matching our requirements, they both have good documentation available and good community support (especially in the case of Neo4j). Neo4j's

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

approach is based on a native graph database, in which we can install a library (provided by the company) that give us the opportunity to import and manage ontologies. Graph DB is directly oriented to ontologies, providing a native RDF store. Both of them provide interesting free options in their pricing, which we use in order to test them using real implemented approaches (our AMS prototype).

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