Federated decentralized trusted dAta Marketplace for Embedded finance

D3.5 - Federated Data Assets Catalogue II

Title	D3.5 - Federated Data Assets Catalogue II
Revision Number	1.0
Task reference	T3.3 T3.4
Lead Beneficiary	UNP
Responsible	Márcio Mateus
Partners	DAEM, IQB, MOH, NOVA, NOVO
Deliverable Type	DEM
Dissemination Level	PU
Due Date	2025-03-31 [Month 30]
Delivered Date	2025-07-31
Internal Reviewers	IDSA GFT
Quality Assurance	UPRC
Acceptance	Coordinator Accepted
Project Title	FAME - Federated decentralized trusted dAta Marketplace for Embedded finance
Grant Agreement No.	101092639
EC Project Officer	Stefano Bertolo
Programme	HORIZON-CL4-2022-DATA-01-04

This project has received funding from the European Union’s Horizon research and innovation programme under Grant Agreement no 101092639

Revision History

Version	Date	Partners	Description
0.1	2025-05-10	ENG	Table of Contents
0.2	2025-06-15	UNP NOVA	Integrated version with contributions
0.3	2025-07-30	UPRC	Version for peer review
0.4	2025-07-31	UNP	Update after peer review
1.0	2025-07-31	UNP	Version for submission

Definitions

Acronym	Definition
AAI	authentication authorization infrastructure
ACM	Association for Computing Machinery
AI	Artificial Intelligence
API	Application Programming Interface
APM	Assets Policy Management
AWS	Amazon Web Services
BERT	Bidirectional Encoder Representations from Transformers
CRUD	Create Retrieve Update Delete - Basic Operations in DBMS
DCAT	Data Catalog Vocabulary
FAME	Federated decentralized trusted dAta Marketplace for Embedded finance
FAQ	Frequently Asked Questions
FDAC	Federated Data Assets Catalogue
FIBO	Financial Industry Business Ontology
FIGI	Financial Instrument Global Identifier
GIS	Geographical Information System
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
ICT	Information Communication Technologies
ID	Identity
IEEE	Institute (of) Electrical (and) Electronic Engineers
JSON	JavaScript Object Notation
JWT	JSON Web Token
LLM	Large language model
OWL	Web Ontology Language (W3C)
RDF	Resource Description Framework
REST	Representational State Transfer
RPC	Remote Procedure Call
SA	Solution Architecture
SDK	Software Development Kit
SEO	Search Engine Optimization
SOTA	State of The Art
TR	Technical Requirement
UI	User Interface
UNP	Unparallel Innovation

URL	Uniform Resource Locator
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Other acronyms and abbreviations not present in the table, are introduced in the text along with their definitions.

Executive Summary

This deliverable is the second of two deliverables that report the work carried out in T3.3 – Federated Catalogue of Data Assets and T3.4 – Semantic Interoperability Middleware, both part of WP3 – "Secure, Interoperable, Federated Data Management".

The objective of this deliverable is to report the prototype of the Federated Data Assets Catalogue (FDAC) and the Semantic Interoperability Middleware. In this regard, this deliverable aims to describe the main functionalities and insights about the implementation details of those components. At the same time, it aims to present the interfaces that should be used in the functionalities of the presented prototypes.

With this purpose, this deliverable reports the development of the Federated Data Assets Catalog and the Semantic Interoperability Middleware, having identified the background technologies used in the development of those components, the functionalities provided by the background technologies and the functionalities added in the context of FAME.

This deliverable advances from the previous version by describing new functionalities developed in FDAC as well as integrations with Marketplace dashboard and the Asset Policy Manager. A new approach for the Semantic Middleware is also described, presenting a solution using LLMs and AI related tools to automate processes of processing Asset descriptions and add the

This deliverable also provides specific examples about how to use some of the described functionalities. These functionalities are extended by exposing asset edition capabilities on a REST API, enhanced with a search endpoint that aims to support the execution of search queries, and the implementation of a plugin system to extend assets' information and visualization information with resources provided by external systems.

Table of Contents

List of Figures

List of Tables

1 Introduction

This deliverable is part of WP3 – "Secure, Interoperable, Federated Data Management" whose main objectives are to:

- implement the project's AAI infrastructure for federated access to data providers, including data marketplaces, data spaces and other data sources;
- design and implement a unified approach for managing and enforcing data policies in a federated environment;
- design and offer a federated catalogue of data assets as part of the marketplace;
- design and implement a platform for semantic interoperability of diverse data assets across the federated data sources of the project; and
- design and implement a set of tools or regulatory compliance by design and regulatory compliance auditing of data-driven embedded finance applications over the FAME marketplace.

This deliverable is the second of two deliverables that report the work carried out in T3.3 – Federated Catalogue of Data Assets and T3.4 – Semantic Interoperability Middleware.

1.1 Objective of the Deliverable

The objective of this deliverable is to report the final implementation of the Federated Data Assets Catalogue and the Semantic Interoperability Middleware. On one hand, this deliverable aims to describe the main functionalities and provide some insights about the implementation details of those components. On the other hand, this deliverable aims at presenting the interfaces that should be used to use the functionalities of the presented prototypes.

1.2 Insights from other Tasks and Deliverables

This Deliverable relates to Deliverable 2.5 - "Requirements, Specifications and Co-Creation" where requirements from Pilot partners were collected and provided valuable information to guide the development of functionalities. Deliverable 2.6 - "Technical Specifications and Platform Architecture" also relates to this deliverable by identifying the FAME components that need to interact with the components described in this deliverable. At the same time, this deliverable can be used by other tasks from WP3, tasks from WP4 and Task 2.4 by describing the functionalities provided by the Federated Data Assets Catalogue and the Semantic Interoperability Middleware and how they can be used, providing valuable insights targeting their integration.

1.3 Structure

This document is divided into five (5) main chapters:

- Chapter 1 – Introduction: This chapter introduces the deliverable by highlighting its objective and its relation to other tasks and deliverables.
- Chapter 2 – Positioning in FAME SA: This chapter highlights the components described in this deliverable and how they relate with the remaining components of the FAME architecture.
- Chapter 3 – Components Specification: This chapter describes the Federated Data Assets Catalogue and the Semantic Interoperability Middleware, along with the identified background technologies that support their development to some extent and describes the functionalities provided.

- Chapter 4 – Components Demonstration: This chapter provides specific examples about how to use some of the functionalities described in this deliverable.
- Chapter 5 – Conclusion: This chapter concludes the deliverable and provides insights about the future steps that will be carried out in the development of the Federated Data Assets Catalogue and the Semantic Interoperability Middleware.

1.4 Summary of Changes

Table 1- D3.5 Updates from D3.2

Section	Status	Description of Update/Addition
		Section 1
1.2 - Insights from other Tasks and Deliverables	Updated	Alignment of D3.5 input regarding the technical activities and progress of the project.
1.4 - Summary of Changes	Added	Depiction of updates between D3.2 and D3.5.
		Section 3.1
3.1.3.1 – FDAC architecture	Updated	Updated the updated FDAC C4 diagrams and descriptions to match last version
3.1.3.3 - Integration with Asset Policy Manager	Added	Described the components developed, APIs provided and flows implements for integration of Asset policy manager with the dashboard and FDAC
3.1.3.4 - Asset Provenance Certification	Added	Described changes made to implement the support for Certification of assets
3.1.3.5 - Visual integration: Iframe enhancement	Added	Described the protocol to creates to apply filtering and sorting functions to iframes
		Section 3.2
3.2.1 - Description	Updated	Provided brief description of the new approach for semantic middleware
3.2.3 - New Approach: Usage of Large Language Models (LLMs)	Added	Added Description of new approach for the Semantic Middleware. Includes Related work where are identified related LLM and AI technologies and a description of the implementation. Is also identified how the plan for the improvement of the technology after FAME.
		Section 4
4.5- Asset Certification Endpoints	Added	Demonstration of how to use the REST API endpoints.
4.6 - Requesting Asset Iframes	Added	Demonstration of how to use the REST API endpoints.
4.7 - Iframe Post message API example	Added	Example of how to use the defined message structure to programmatically change the assets visualisation.
		Section 5
Conclusions	Updated	Update of the overall conclusions of the deliverable to reflect the final findings and outcomes of the deliverable.

2 Positioning into FAME SA

This deliverable focuses on the description, specification, and demonstration of two (2) components of the FAME architecture (presented in Figure 2-1). Those components are the Data Assets Catalogue, in this document, referred to as Federated Data Asset Catalogue, and the Semantic Interoperability, which will be referred to as Semantic Interoperability Middleware.

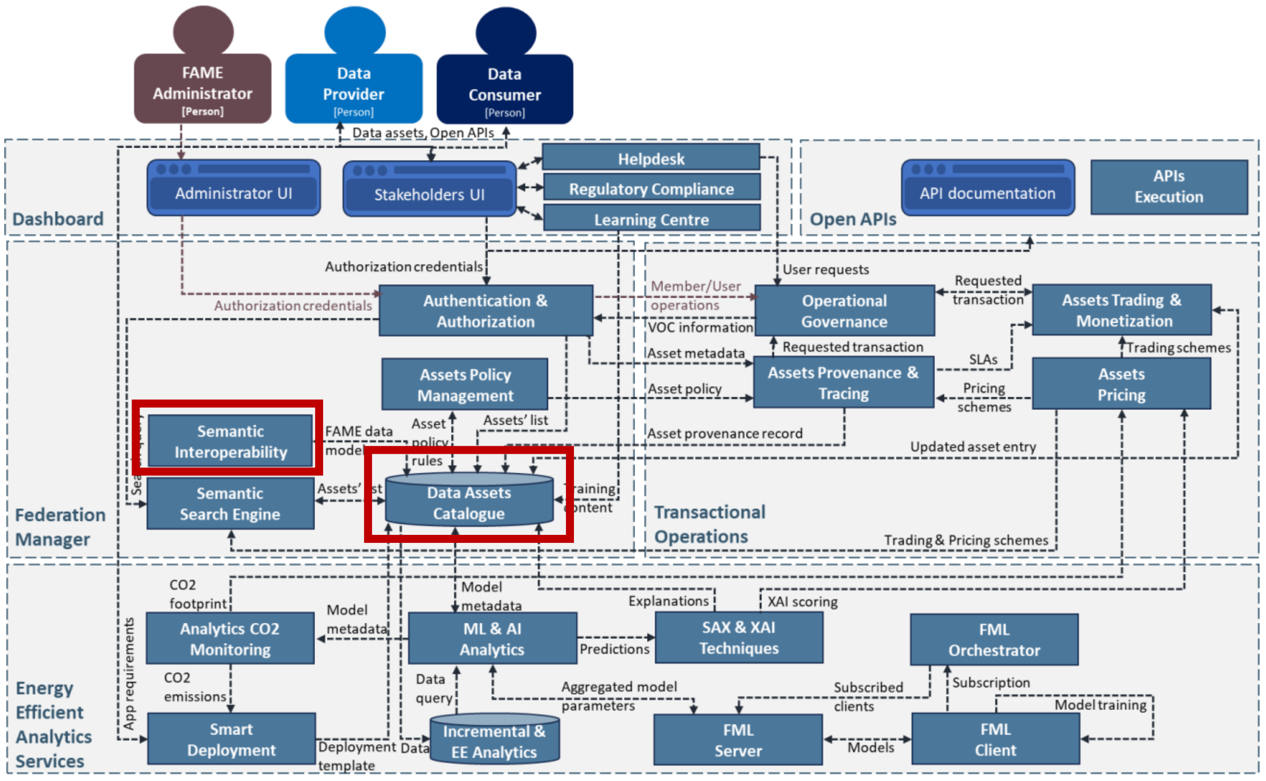


Figure 2-1 – FAME SA C4 container diagram

The *Federated Data Assets Catalogue (FDAC)* is responsible for storing and indexing all the information regarding the data assets of FAME, which represent multiple types of content, ranging from datasets to AI models, services, or relevant documentation. This component provides asset management functionalities that are exposed to the Dashboard using Open APIs and need to interact with the Authentication & Authorization component to receive information about the user interaction with the dashboard and attempting to perform actions in FDAC. The Asset Policy Management is evoked by FDAC to validate the read and write permissions for each asset.

The *Semantic Interoperability Middleware* enhances the interoperability capabilities of the FDAC by extending the amount of data models supported by FDAC. This is achieved by providing mappings between the FDAC internal data model and the relevant data models and online model translation capabilities.

In the next sections, more details are provided about the design and implementation of these components.

3 Components Specification

3.1 Federated Data Asset Catalogue (FDAC)

3.1.1 Description

The FDAC operates as a registry within FAME, tasked with the methodical cataloguing and archival of assets. The spectrum of these assets spans, among others, datasets, AI models, services, and essential documentation, while ensuring a structured approach to data management.

This registry is able not only to represent assets originated from FAME activities but also to index assets deriving from external sources, such as relevant Data Spaces or Data Marketplaces. All the information about the assets will be made available to any FAME component that requires such information for its operation by the usage of well-defined interfaces.

3.1.2 Related Work

The implementation of the Federated Data Asset Catalogue is the setup of a mechanism to store/index asset information. This mechanism must be able to support the CRUD actions associated with asset management and must define the data structure used to describe an asset. This section describes the existing technologies that can contribute to the implementation of the Federated Data Asset Catalogue.

3.1.2.1 UNPARALLEL Web Catalogue Framework

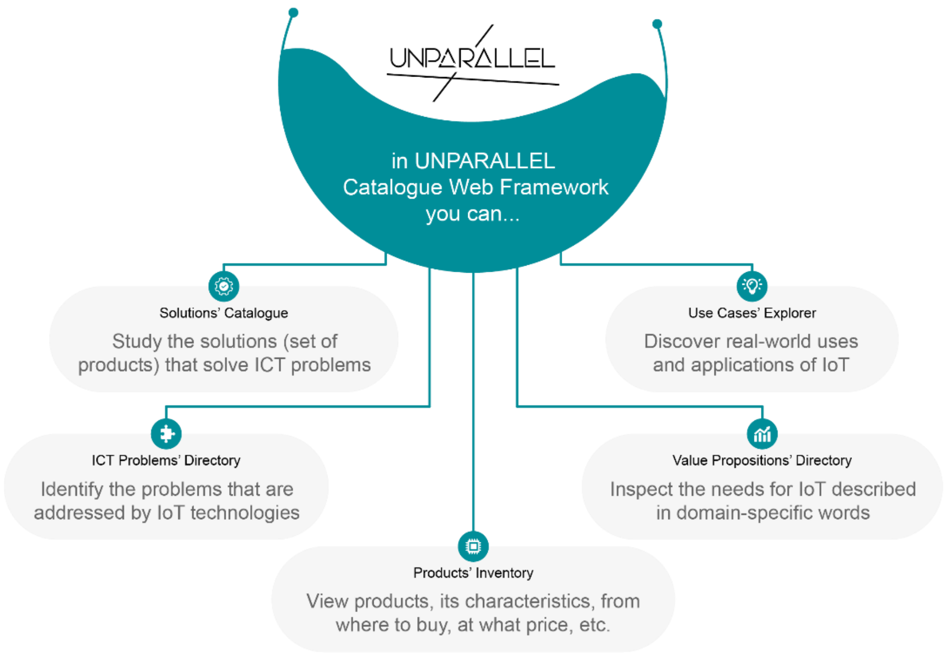

UNPARALLEL
in UNPARALLEL Catalogue Web Framework you can...
Solutions' Catalogue
Study the solutions (set of products) that solve ICT problems
Use Cases' Explorer
Discover real-world uses and applications of IoT
ICT Problems' Directory
Identify the problems that are addressed by IoT technologies
Value Propositions' Directory
Inspect the needs for IoT described in domain-specific words
Products' Inventory
View products, its characteristics, from where to buy, at what price, etc.

Figure 3-1 – UNPARALLEL Web Catalogue Framework

The UNPARALLEL Web Catalogue Framework is a well-established outcome of several Unparallel Innovation's developments during the past years. With its content structuring capabilities already being used in company products such as the IoT-Catalogue.com, it is perfectly suitable for leveraging the content of the Datasets Catalogue. The UNPARALLEL Web Catalogue Framework serves as a backend base of information, providing the Datasets Catalogue with data on components and datasets through the following functionalities:

i) **Product's Inventory**

It provides information about several components from simple sensors with common functions to more complex instrumentation delivering out-of-the-box technological solutions, with the possibility to apply filters when listing the products, by type, location, etc.

When opening a product, detailed information is displayed, including its characteristics, purchase options, price, name, description, etc. Several interactive bars are also provided, allowing for an intuitive navigation within the product page and through the other elements related to it.

ii) **ICT Problems and Value Propositions Directory**

This framework provides a way to describe several situations which could be described as a simple technological problem known as an ICT Problem. It usually describes a very simple situation (e.g.: Measure Temperature) that can make use of several products to provide a solution (e.g.: Temperature sensor).

A Value Proposition should be used when describing more complex problems, with usually reduced focus on the technological part and an increased focus on the type of problem to be solved. For instance: to describe the solution to a value proposition for a client or partner, we should break it into ICT Problems and then indicate which products should be employed respectively.

iii) **Use Case's Explorer and Solution's Catalogue**

A use case defines a real-world scenario which could be an experiment, process or other type of activity. The UNPARALLEL Web Catalogue Framework gives extensive support to define a use case, which will be characterized by Value Propositions, ICT Problems, Products and solutions. This framework is a powerful tool for anyone who wants to define a real-world complex scenario that can be solved by solutions relying on products available on the market.

3.1.2.1.1 Asset modulation

An asset is a generic element that represents something that can be described and/or indexed, such as a hardware component, a software component, a service, a dataset, a data model, etc. Table 2 shows a template of the information an asset is expected to have. This information serves to further characterize to assets presented to the user.

Table 2 - Asset Information Template

General information	
Brief information of the Component.	
Name	Name of the Component.
Summary	Short description of the Component (with a maximum of 280 char).
Description	Extended description of the Component (being able to highlight the text in different styles, bold, italic, bullet points, etc.).
Website	The website of the Component.
Manufacturer / Provider	Name of Component's manufacturer or provider entity.
Contact	Name and email of the contact(s) person.

Type	Indicate the type of the Component type. E.g., Component: Platform, Sensor, Gateway, Dataset, Machine Learning Model, Library, Extension, As a Service or Other Software.
Features / Capabilities	The Features focuses on the asset's abilities, i.e., what the component is able to do. Description, in the form [«action verb»] «direct object (the "what" that was acted upon)», examples: "[Measure] wind speed", "[Measure] air temperature", "[Measure] wind direction" for a Weather Station.
Standards	List of Standards supported by the Component.
License	If Open source specify the license[1] (e.g., Apache-2.0, MIT, etc.)
TRL[2]	If the component is in development, select the Technology Readiness Level (TRL) target of the Component according to the European Commission.
Reference	Provide useful documentation referred to the component: • Documentation: such as instructions manuals, datasheets, publications, API information related to the component, and so on. • Repository: Gitlab, GitHub URL and so on.
Linked Components	List the components and their relationship (e.g., uses, based on, composed by). E.g., Atos' Smart Fleet Framework based on FIWARE.
Media Gallery	Media gallery of the Component such as photos, images and videos. In the videos case, add the link in this field. Select the image of the Component to be the main image/logo.

Assets can be categorised as being of different types. Despite not being limited to them, usually, three types are used to classify most assets. Those types are:

- **Components:** Provides information about several components from simple sensors with common functions to more complex instrumentation delivering out-of-the-box technological solutions, with the possibility to apply filters when listing them by type, location, and so on. When opening a component, detailed information is displayed, including its characteristics, purchase options, price, name, description, etc. Several interactive bars are also provided, contributing to intuitive navigation within the component page and through the other elements related to it.
- **Data Model:** Data models represent the type of information that datasets can have, and are composed of three domains:
  - Data Concept – corresponds to an application domain for aggregation of data, where the different data measurements and properties are used together to better describe the state of something. A Data Concept can be represented by multiple Measurable Quantities.
  - Measurable Quantity – represents anything that can be measured, quantified or specific information distributed under different formats, such as images or text. Measurable Quantities may represent a property of something (e.g., mass, length, temperature, etc.), the count of an amount (just to represent some cases) or a property of a concept they are related to. For instance, when describing traffic

[1] https://www.iot-catalogue.com/openSourceLicences
[2] https://ec.europa.eu/research/participants/data/ref/h2020/other/wp/2016_2017/annexes/h2020-wp1617-annex-g-trl_en.pdf

volume in a particular model, instead of quantifying it, it may be described by an enumerated variable such as low, medium, and high volumes of traffic.
  - Unit – represents the magnitude of a specific Measurable Quantity according to a given system of units.
- **Dataset:** Displays information about large amounts of data in an organised and intuitive manner, including the Measurable Quantities represented in the dataset, the location in a map view, and a chart representation of the values for numeric datasets, amongst other types of data visualisation.

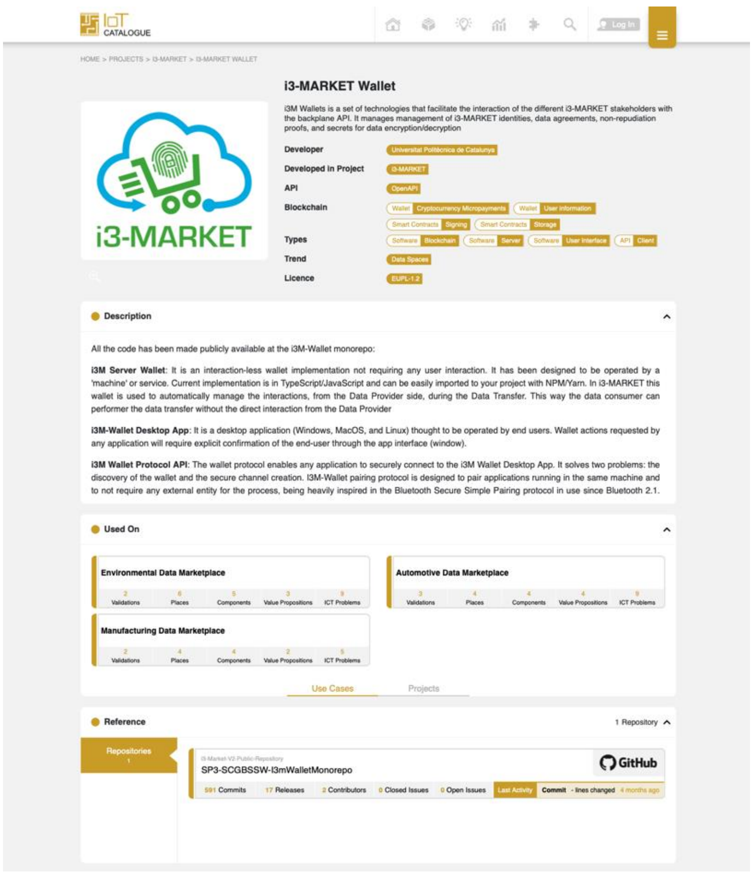
3.1.2.1.2 Asset Repository

The UNPARALLEL Web Catalogue Framework provides both a database to store and index the metadata of assets and a set of user interfaces that users can use to navigate across all the assets indexed. In to following subsections will be described the user interfaces to navigate and visualise the information of the different types of assets.

3.1.2.1.2.1 Assets Web Interfaces

Component Description

The Technological Asset page will contain relevant information about the technology, which includes general information, such as title, representative image, owner/developer, brief description, developed in Project, API, Blockchain, Types, Trend, and Licence. The description provides more detail about its functionalities. The information related to the technology will be displayed in the Components and Used-On sections. The page also has a references section (at the end of the page) for useful documentation (such as instruction manuals, datasheets, publications GitHub) about the technology (Figure 3-2).



i3-MARKET Wallet

i3M Wallets is a set of technologies that facilitate the interaction of the different i3-MARKET stakeholders with the backplane API. It manages management of i3-MARKET identities, data agreements, non-repudiation proofs, and secrets for data encryption/decryption

Developer	Universitat Politècnica de Catalunya
Developed in Project	i3-MARKET
API	OpenAPI
Blockchain	Wallet Cryptocurrency Micropayments; Wallet User information; Smart Contracts Signing; Smart Contracts Storage
Types	Software Blockchain; Software Server; Software User Interface; API Client
Trend	Data Spaces
Licence	EUPL-1.2

Description

All the code has been made publicly available at the i3M-Wallet monorepo:

i3M Server Wallet: It is an interaction-less wallet implementation not requiring any user interaction. It has been designed to be operated by a 'machine' or service. Current implementation is in TypeScript/JavaScript and can be easily imported to your project with NPM/Yarn. In i3-MARKET this wallet is used to automatically manage the interactions, from the Data Provider side, during the Data Transfer. This way the data consumer can performer the data transfer without the direct interaction from the Data Provider

i3M-Wallet Desktop App: It is a desktop application (Windows, MacOS, and Linux) thought to be operated by end users. Wallet actions requested by any application will require explicit confirmation of the end-user through the app interface (window).

i3M Wallet Protocol API: The wallet protocol enables any application to securely connect to the i3M Wallet Desktop App. It solves two problems: the discovery of the wallet and the secure channel creation. I3M-Wallet pairing protocol is designed to pair applications running in the same machine and to not require any external entity for the process, being heavily inspired in the Bluetooth Secure Simple Pairing protocol in use since Bluetooth 2.1.

Used On

	Validations	Places	Components	Value Propositions	ICT Problems
Environmental Data Marketplace	2	6	5	3	9
Automotive Data Marketplace	3	4	4	4	9
Manufacturing Data Marketplace	2	4	4	2	5

Use Cases Projects

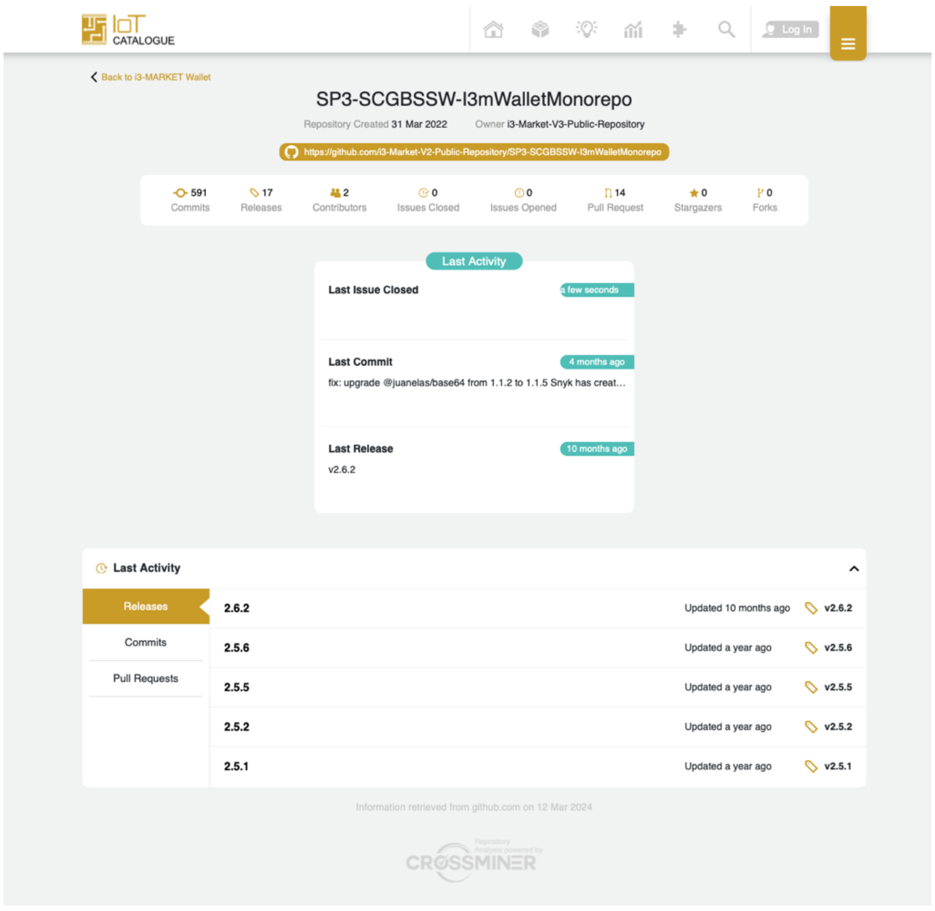
Reference 1 Repository

Repositories 1

i3-Market-V2-Public-Repository
SP3-SCGBSSW-I3mWalletMonorepo GitHub
591 Commits 17 Releases 2 Contributors 0 Closed Issues 0 Open Issues Last Activity Commit - lines changed 4 months ago

Figure 3-2 – Web interface for component visualization

As shown in Figure 3-3, in the references section, the user has access to the project repository link, GitHub information, a resume of the last activity (with Last Issued Closed, Last Commit and Last Release) and at the end of the page more details about last activity with the last five entrances of the Pull Requests, Commits and Releases.



SP3-SCGBSSW-I3mWalletMonorepo

Repository Created 31 Mar 2022   Owner i3-Market-V3-Public-Repository

https://github.com/i3-Market-V2-Public-Repository/SP3-SCGBSSW-I3mWalletMonorepo

591	17	2	0	0	14	0	0
Commits	Releases	Contributors	Issues Closed	Issues Opened	Pull Request	Stargazers	Forks

Last Activity

Last Issue Closed — a few seconds

Last Commit — 4 months ago
fix: upgrade @juanelas/base64 from 1.1.2 to 1.1.5 Snyk has creat...

Last Release — 10 months ago
v2.6.2

Last Activity

Releases | Commits | Pull Requests

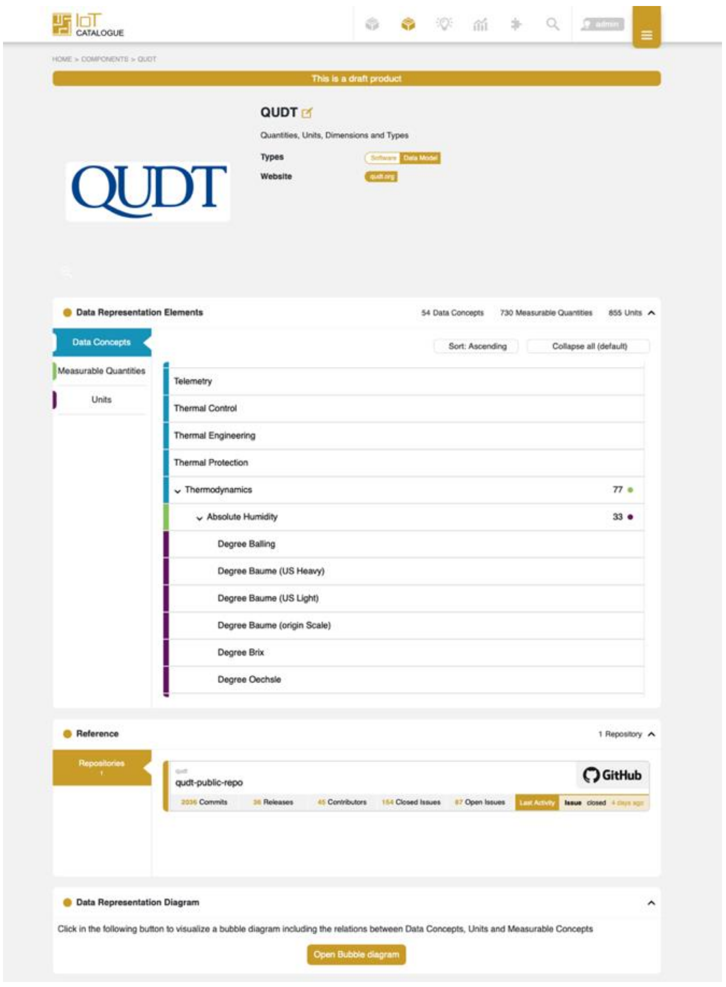
2.6.2	Updated 10 months ago	v2.6.2
2.5.6	Updated a year ago	v2.5.6
2.5.5	Updated a year ago	v2.5.5
2.5.2	Updated a year ago	v2.5.2
2.5.1	Updated a year ago	v2.5.1

Information retrieved from github.com on 12 Mar 2024

Figure 3-3 – Component repository details

Data Model

Figure 3-4 shows the Data model page displays the concepts, measurables and units that belong to the model in a navigable hierarchy. Data models are used by the dataset, describing the data they contain. These may be reused in full or in part by other datasets. The page allows for the visualisation of the model's concepts indicating the measurables each concept entails and the associated units.



IoT CATALOGUE

admin

HOME > COMPONENTS > QUDT

This is a draft product

QUDT

Quantities, Units, Dimensions and Types

Types: Software, Data Model

Website: qudt.org

Data Representation Elements — 54 Data Concepts, 730 Measurable Quantities, 855 Units

Data Concepts | Measurable Quantities | Units

Sort: Ascending | Collapse all (default)

- Telemetry
- Thermal Control
- Thermal Engineering
- Thermal Protection
- Thermodynamics 77
  - Absolute Humidity 33
    - Degree Balling
    - Degree Baume (US Heavy)
    - Degree Baume (US Light)
    - Degree Baume (origin Scale)
    - Degree Brix
    - Degree Oechsle

Reference — 1 Repository

Repositories 1

qudt-public-repo — GitHub

2036 Commits | 36 Releases | 45 Contributors | 154 Closed Issues | 87 Open Issues | Last Activity: Issue closed 4 days ago

Data Representation Diagram

Click in the following button to visualize a bubble diagram including the relations between Data Concepts, Units and Measurable Concepts

Open Bubble diagram

Figure 3-4 - Data Model page for QUDT (Quantities, Units, Dimensions and types)

Dataset

The dataset page displays the information provided by it. Different types of data require different visualisations. Currently supported types are times-series data (Figure 3-5) and geographic information system (GIS) data as shown in Figure 3-6. The page displays the measurable quantities used by the dataset and its corresponding units, along with the concepts it captures.

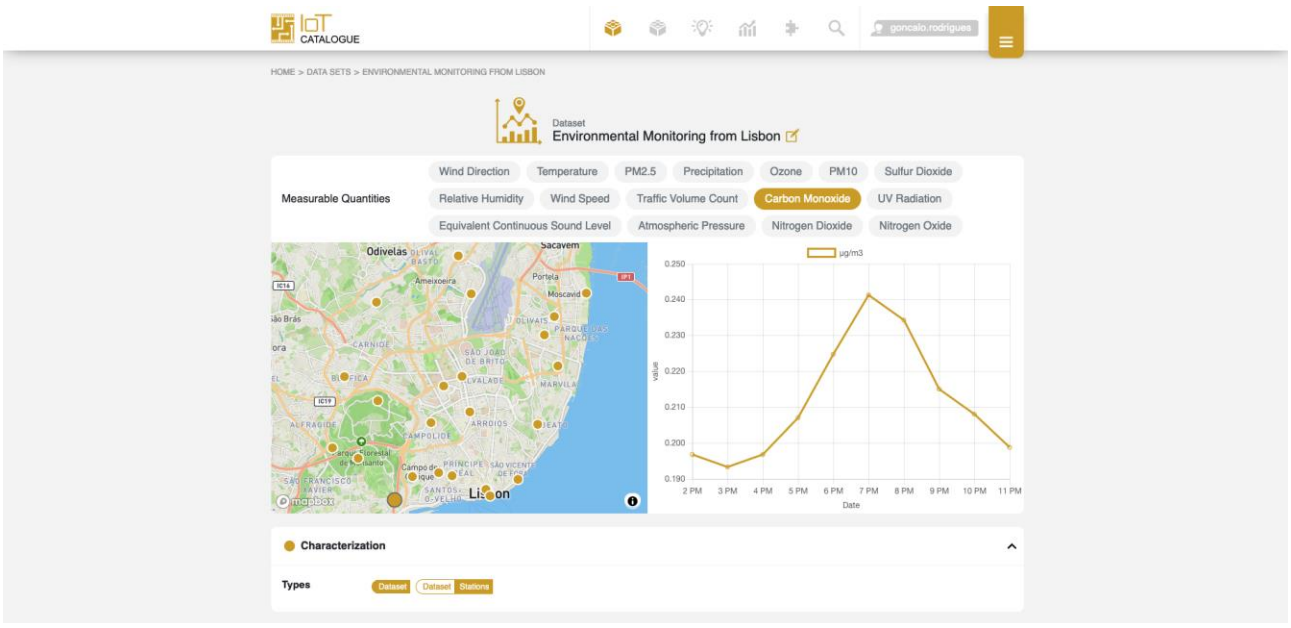


Figure 3-5 - Dataset page displaying information provided by Lisbon's environmental monitoring public API

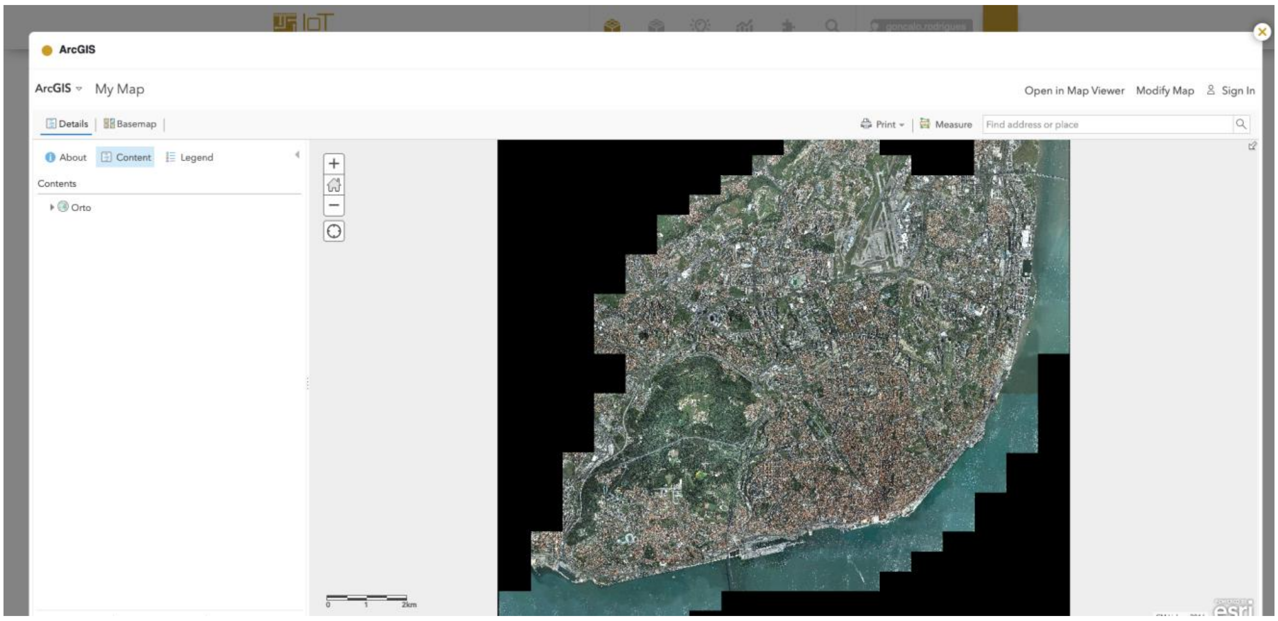


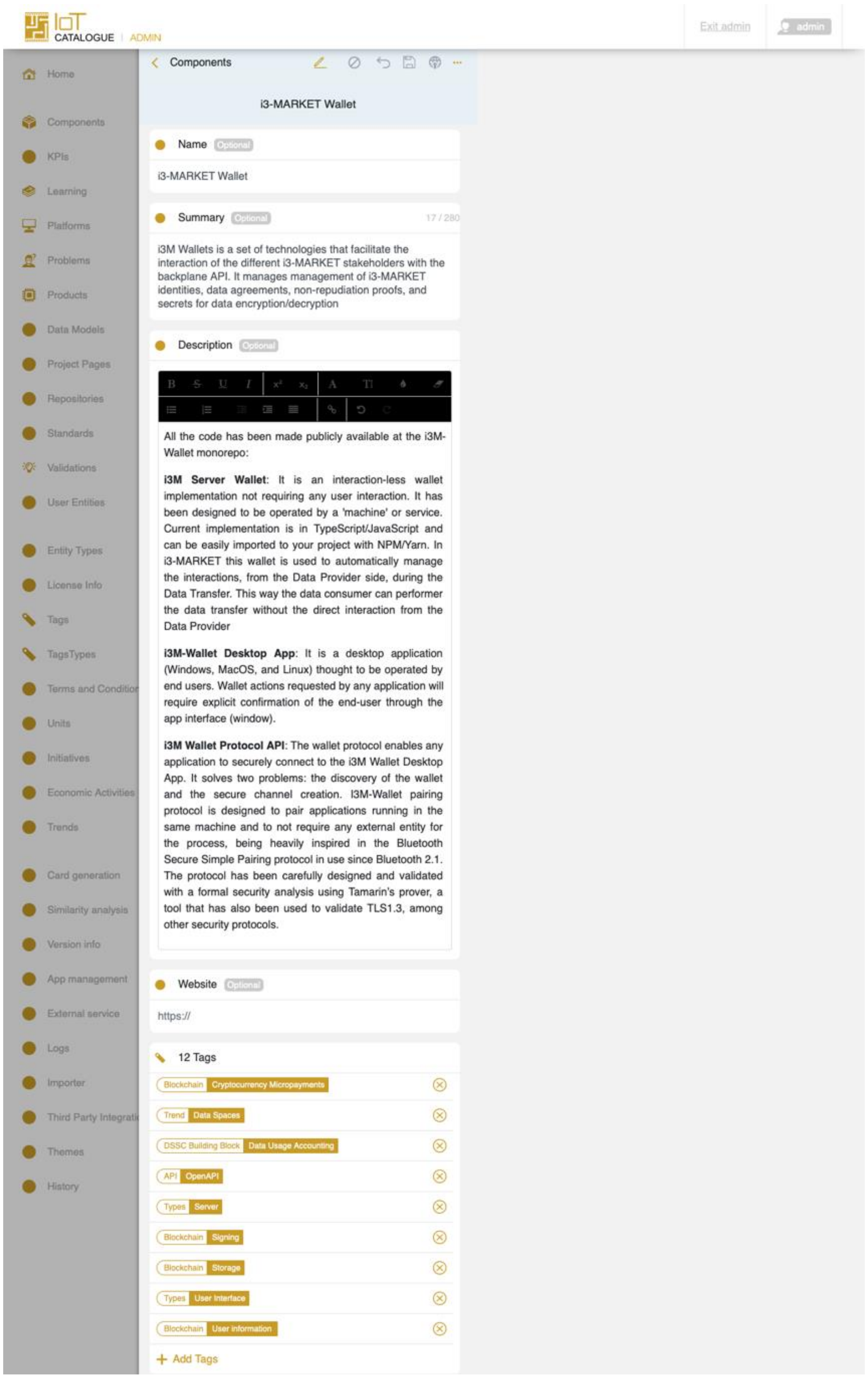
Figure 3-6 - GIS data visualization displaying an orthophoto map of Lisbon

Admin Description

The UNPARALLEL Web Catalogue Framework also contains an admin page with the Name, Summary, Description and Website of the project. Here the Asset Providers can add and/or modify assets (Figure 3-7).

Creating asset relations in the admin dashboard enhances user experience by establishing visible relations between them such as uses, used on, used by, composed by, or parent/child relations. Data models, for instance often have extensions of the base model. These extensions can relate to the base model via a parent/child relation on the Web Framework, while components may be made up of other smaller components and the composed by relation can be used to identify this link between FDAC components.

Adding tags through the dashboard is also critical because these not only provide further context for the asset but also help during the search being used for filtering results.



i3-MARKET Wallet

Name Optional

i3-MARKET Wallet

Summary Optional 17 / 280

i3M Wallets is a set of technologies that facilitate the interaction of the different i3-MARKET stakeholders with the backplane API. It manages management of i3-MARKET identities, data agreements, non-repudiation proofs, and secrets for data encryption/decryption

Description Optional

All the code has been made publicly available at the i3M-Wallet monorepo:

i3M Server Wallet: It is an interaction-less wallet implementation not requiring any user interaction. It has been designed to be operated by a 'machine' or service. Current implementation is in TypeScript/JavaScript and can be easily imported to your project with NPM/Yarn. In i3-MARKET this wallet is used to automatically manage the interactions, from the Data Provider side, during the Data Transfer. This way the data consumer can performer the data transfer without the direct interaction from the Data Provider

i3M-Wallet Desktop App: It is a desktop application (Windows, MacOS, and Linux) thought to be operated by end users. Wallet actions requested by any application will require explicit confirmation of the end-user through the app interface (window).

i3M Wallet Protocol API: The wallet protocol enables any application to securely connect to the i3M Wallet Desktop App. It solves two problems: the discovery of the wallet and the secure channel creation. I3M-Wallet pairing protocol is designed to pair applications running in the same machine and to not require any external entity for the process, being heavily inspired in the Bluetooth Secure Simple Pairing protocol in use since Bluetooth 2.1. The protocol has been carefully designed and validated with a formal security analysis using Tamarin's prover, a tool that has also been used to validate TLS1.3, among other security protocols.

Website Optional

https://

12 Tags

Blockchain	Cryptocurrency Micropayments
Trend	Data Spaces
DSSC Building Block	Data Usage Accounting
API	OpenAPI
Types	Server
Blockchain	Signing
Blockchain	Storage
Types	User Interface
Blockchain	User information

+ Add Tags

Figure 3-7 - Admin dashboard for asset creation

Advancements beyond the Related Work

Further developments beyond those previously discussed include a plugin system and a REST API. The plugin system allows for visual elements provided by the other tools and components of FAME to be displayed in the asset catalogue, enriching each asset page with its own visualisation tools and information. The developed REST API enables communication between FDAC and the UNPARALLEL Web Catalogue Framework. It currently provides endpoints to create and manipulate all the different assets as well as provide a search functionality.

3.1.3 Technical Specification

3.1.3.1 Component-level C4 Architecture

This subsection describes FDAC's main functionalities related to the capability to store and index asset metadata, providing interfaces to support the CRUD actions over such assets. All interactions with those functionalities will be processed by the FDAC API that provides an Open API description of its endpoints, identifying the functionalities provided and how to use them. CRUD functionalities are supported by the components "Asset Management" and Database represented on the diagram in Figure 3-8. Those functionalities are enhanced by the "Search" component that provides users with more discovery functionalities. Every request to list and an access assets are validated by checking within the Asset Policy Manager if the user has the required permissions to visualise the asset.

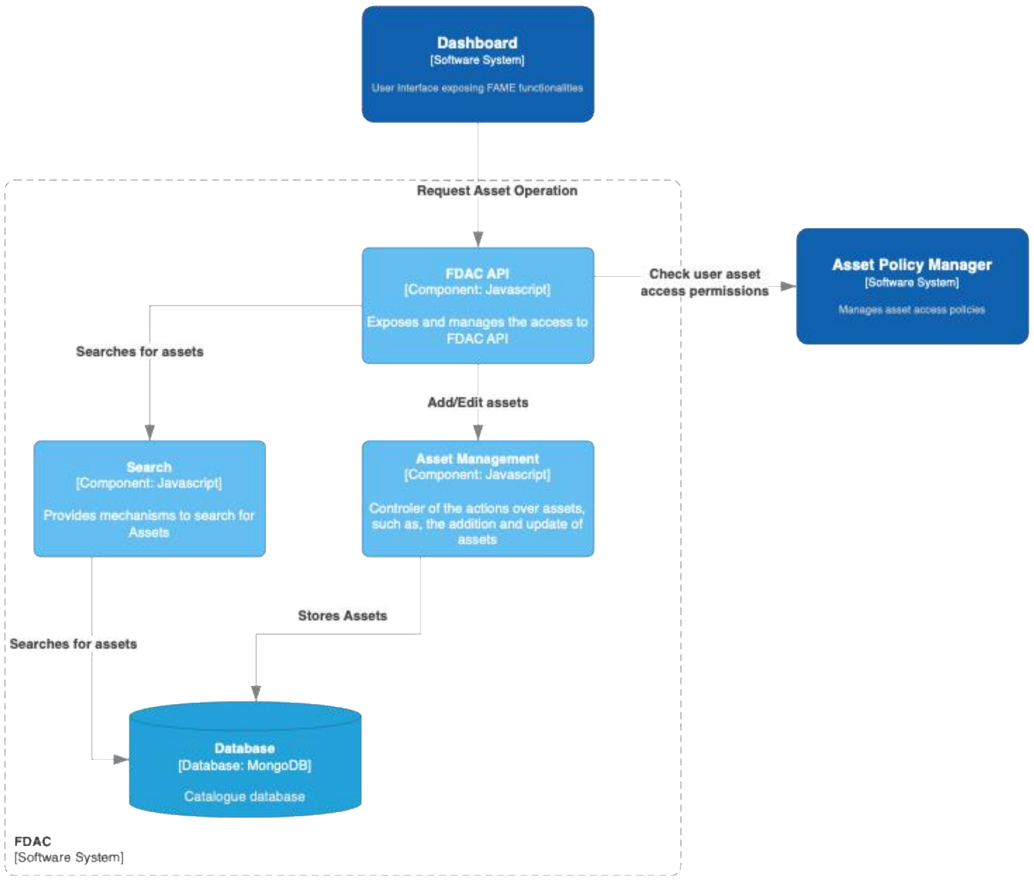


Figure 3-8 – FDAC C4 architecture

3.1.3.2 Asset Manager

The Asset Manager controls actions of creation, deletion, and modification of assets. The base functionalities of this component are provided by the UNPARALLEL Web Catalogue Framework. However, this component has been extended to support the usage of CRUD actions via REST API and the capability to extend the information associated with an asset with information produced by external tools.

3.1.3.2.1 REST API

The REST API, used to send and retrieve data from the UNPARALLEL Web Catalogue Framework, was developed by extending an existing library that was used internally, the IoT-Catalogue-Data-API . This initial implementation used DDP (Datagram Delivery Protocol) while the one used by FDAC is REST (Representational State Transfer) based. DDP is better suited for controlled environments, such as for internal development, while the REST standard with an OpenAPI specification, significantly facilitates its integration with heterogeneous components using different technologies, hence the change.

The FDAC REST API, shown in Figure 3-17, provides endpoints to manipulate all FDAC'S assets through GET, POST, PUT or DELETE requests. Thus, enabling users to retrieve a list of assets, get information on a specific asset, update an asset's information and so on. Users require an authentication token to use the API.

The search endpoint (**Figure 3-18**) provides a free text search with optional parameters. Results can be filtered per manufacturer, developer, owner, tags, or components. Other options include the expand parameter specifying if the results should include the object's content or only its ID. While the output filter option filters results based on the type of objects to return.

The REST API also provides an endpoint to certificate an asset where FAME can act as a certification authority, certifying the provenance of the Asset and assuring that it followed the full FAME process for its onboarding. The API also provide support to archive an asset, provide asset owners with means to archive an asset, hiding it from other users in the dashboard, this action can be undone by calling the "unarchive" endpoint.

3.1.3.2.2 Plugin Integration

The plugin integration allows the association of external information to data elements indexed in the FDAC instance. Each plugin has the freedom to define the structure of the information to be added.

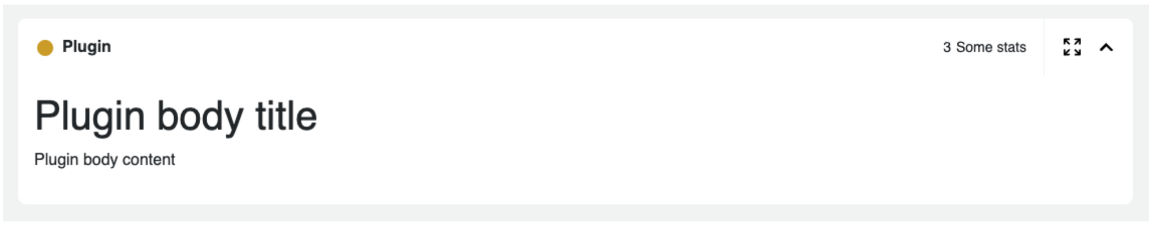
The plugin information can be added through a REST API endpoint, where the following elements will be sent:

- **Token:** Authenticates the endpoint and identifies the ID of the plugin information that will be added to the new element.
- **Element ID:** The ID of the element that will be associated with the content provided.
- **Content:** JSON of the content that will be associated with the component.

Each plugin may also provide dedicated visualisation capabilities to present the added content to the user which will be shown when the user accesses the pages of the indexed items. To define those visualisations plugins some React.JS code must be provided that will be integrated into the FDAC instance. In this code, a function "getData" must be provided to obtain plugin data of a specific data element from FDAC, making it available to the visualisation component.

The following UI components represent different types of visualization that can be used to display custom plugin data regarding each plugin depending on the intended visualisation purpose. Bootstrap V5 is used as a front-end framework meaning that visualisation must respect this library for different visualisation purposes.

That first component is the "Bar" (represented in Figure 3-9), which is used on FDAC to represent sections with information inside the FDAC asset page. This component includes a header with a title and a subtitle (on the left), and a body containing a custom render.



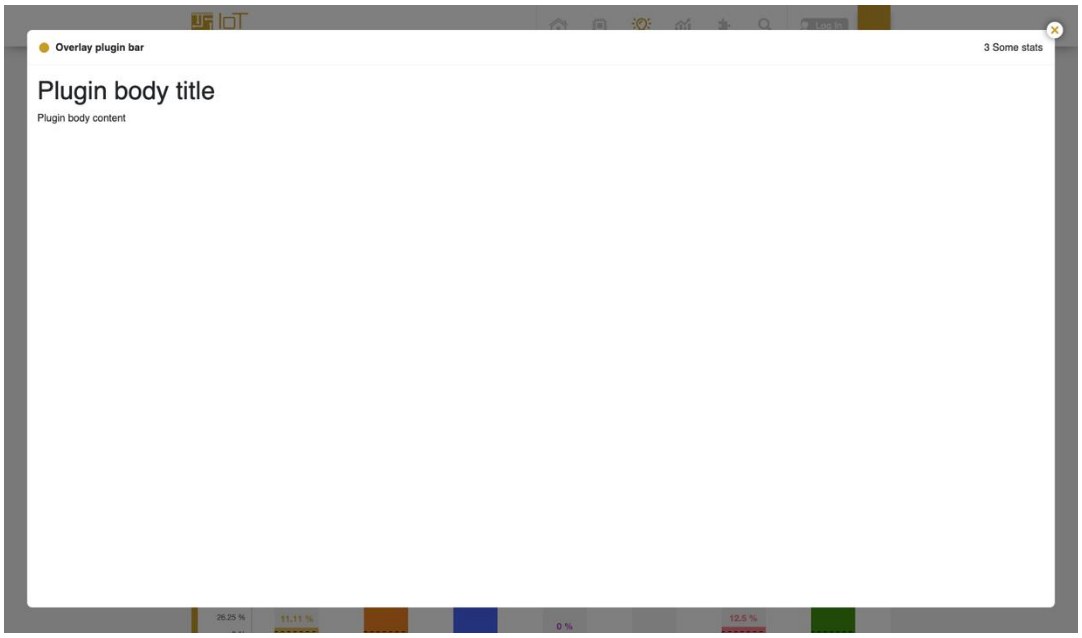
Plugin body title

Plugin body content

Figure 3-9: Plugin Bar available inside the asset page

When specifying the bar, it is possible to define the title, subtitle, and inner content, which will be rendered with the information of the plugin for the specific asset. This bar's visualisation will be rendered inside the asset page and its width will be limited to the max width allowed by the Bootstrap grid system.

The "Overlay" (represented in Figure 3-10) is a component that extends the behaviour of a bar by enabling a fullscreen view of a bar and is useful when a larger visualisation area is required. Like in the case of a regular bar view, this view includes a header and a body containing inner content that can be different from the shown in bar view. When available the overlay can be opened by clicking in the top right icon of the bar inside the asset page.

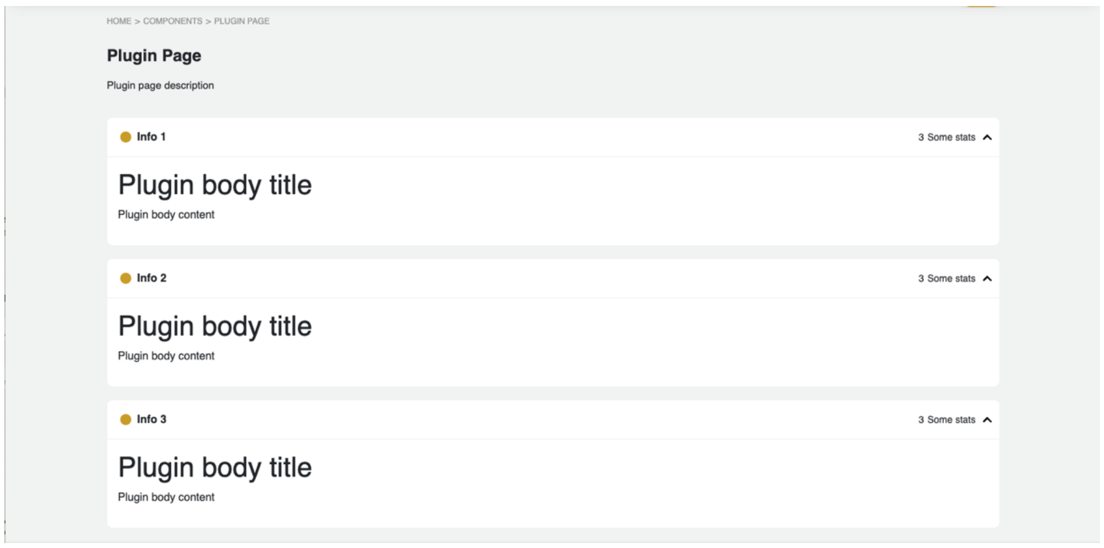


Plugin body title

Plugin body content

Figure 3-10: Plugin Overlay bar view

A full page (represented in Figure 3-11) can also be defined to show the information provided by a plugin regarding a specific asset. This could be useful for cases where there is more information to be displayed, and the space provided by a bar and the overlay view is not enough. It is also useful when is required to split the information along several bars. When defining the page, it is possible to define a header with a title and a description, and to include several bars. Each bar can also support an overlay view.


HOME > COMPONENTS > PLUGIN PAGE

Plugin Page

Plugin page description

Info 1 3 Some stats

Plugin body title

Plugin body content

Info 2 3 Some stats

Plugin body title

Plugin body content

Info 3 3 Some stats

Plugin body title

Plugin body content

Figure 3-11: Plugin Page

3.1.3.3 Integration with Asset Policy Manager (APM)

The implementation of the integration between the FDAC API and APM is presented in Figure 3-12. As the FDAC is an internal component, providing an API that should be protected, a "proxy" component was developed – FDAC Public API. This component exposes a subset of FDAC API containing the endpoints of interest for the FAME Dashboard and FAME REST API users, while being able of processing FAME User Tokens.

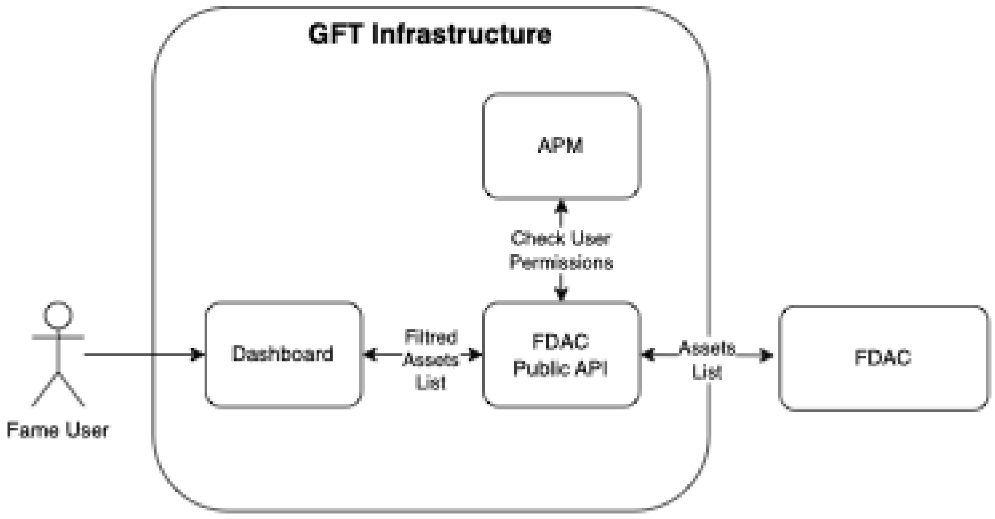


Figure 3-12: Diagram of components involved on integration of APM and FDAC

The FDAC Public API is also responsible for the implementation of integration that orchestrate the information flow between different FAME components. FDAC Public API also creates an authenticated communication with the FDAC using FDAC's internal authentication mechanisms, never exposing details of authentication information with FDAC to the Dashboard or REST users.

The presented diagram is detailed the integration with the Asset Policy Manager (APM). In the flow, when assets need to be rendered in the dashboard a call is made to the FDAC Public API. The flow for rendering assets that an authenticated user has access is the represent in Figure 3-13.

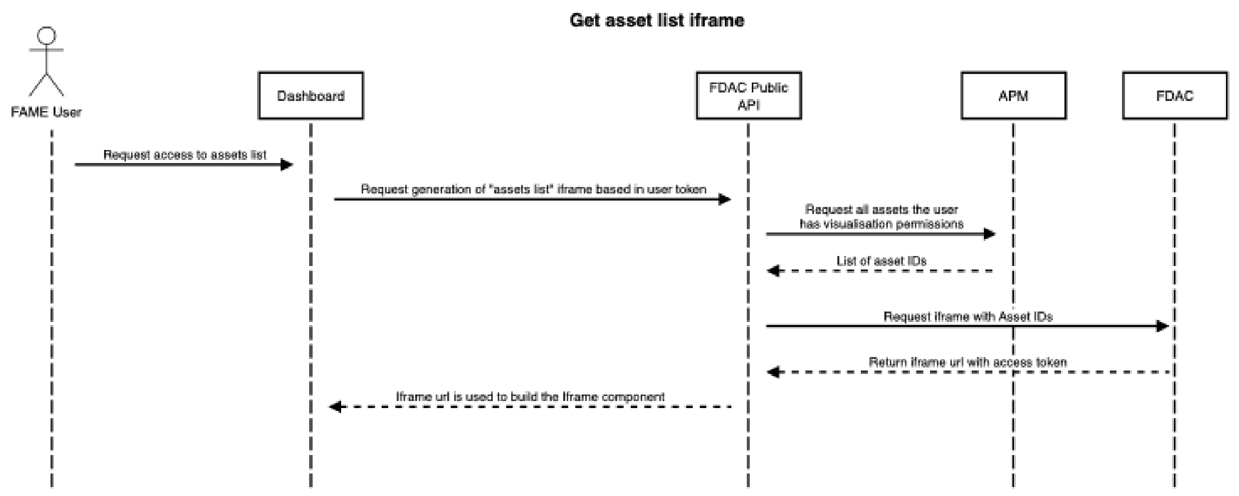

Get asset list iframe
FAME User
Dashboard
FDAC Public API
APM
FDAC
Request access to assets list
Request generation of "assets list" iframe based in user token
Request all assets the user has visualisation permissions
List of asset IDs
Request iframe with Asset IDs
Return iframe url with access token
Iframe url is used to build the Iframe component

Figure 3-13: Diagram representing the flow dashboard requesting an asset list

In the diagram is represented the following flow:

1. FAME user navigate to the asset list page on the dashboard
2. The dashboard sends a request with the JWT user token (if the user is logged in) to the FDAC Public API to generate the Iframe URL
3. The FDAC Public API will request from the APM a list of the authorised Assets IDs for the user associated with the JWT token, if a JWT token is not defined a list with the public asset IDs will be returned
4. Using the authorised asset IDs, the FDAC Public API will request from FDAC a session specific Iframe URL containing the page with the asset list based on the assets IDs
5. The dashboard will use the Iframe URL to show the page with a list of assets to the end user

Some actions can only be applied to Assets if the user has the ownership of the assets. For instance, only the user with ownership rights must be able to edit the visualisation policies associated to a given asset or be able to edit its offerings. The APM can also be used to verify if the user is owner of the asset, allowing to decide whether the button to edit the asset policy should be showed. The flow to identify it the authenticated user has ownership rights over a asset being visualised is shown in Figure 3-14.

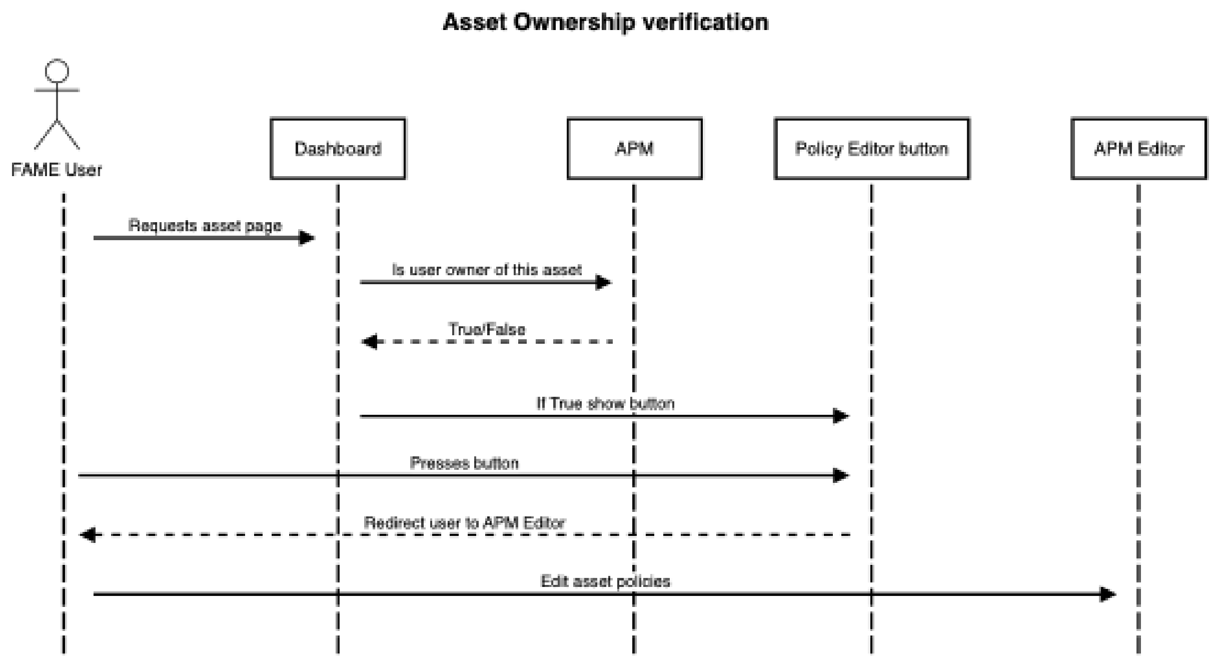


Figure 3-14: Flow to check if user has ownership rights over an asset

In the diagram is represented the following flow to show the policy edition button when the user visualizing the asset has ownership rights:

1. FAME user opens an asset detail page from the asset list page
2. The dashboard sends a request to the APM asking if the user is the owner of the asset
3. If the user is the owner, the policy editor button will be rendered
4. When the user presses the button, it will be redirected to the APM Editor page
5. The user will have access to a form allowing him to edit the policy

3.1.3.4 Asset Provenance Certification

Assets onboarded in the FAME marketplace using the full onboarding process on and registered in the Provenance component are marked as FAME certified, representing that the asset description is the same of the information registered in the Asset Provenance system. To support this functionality a certification API is exposed by FDAC, allowing to certificate an asset using the certificate authority associated to the user authenticated during the request. If the asset is edited the certification becomes irrelevant, in this case a new certification must be issued after deleting the older one.

On the list view all the certified assets are shown with a badge making it easy do differentiate between the certified and non-certified assets (Figure 3-15), when an asset is open information about the certificate authority is showed (Figure 3-15Figure 3-16).

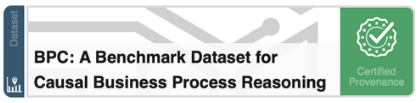
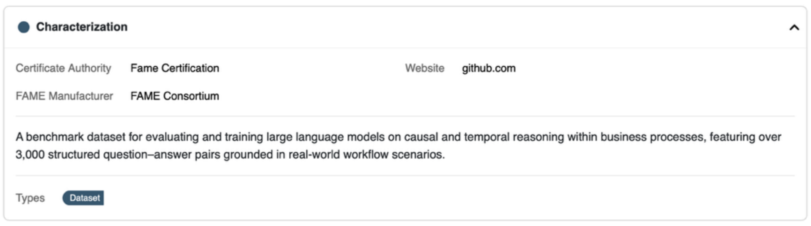

Dataset
BPC: A Benchmark Dataset for Causal Business Process Reasoning
Certified Provenance

Figure 3-15: Example of a certified asset card shown in a list view


Characterization

Certificate Authority Fame Certification
Website github.com
FAME Manufacturer FAME Consortium

A benchmark dataset for evaluating and training large language models on causal and temporal reasoning within business processes, featuring over 3,000 structured question–answer pairs grounded in real-world workflow scenarios.

Types Dataset

Figure 3-16: Example of showing the Certificate Authority that certified an asset

3.1.3.5 Visual integration: Iframe enhancement

After rendering the FDAC Iframe is necessary to establish a communication channel between the Iframe and the dashboard, this will allow to manipulate the asset list which can include filtering, sorting, etc, without reloading the entire Iframe content.

The messages are exchanged using the HTML postMessage API method which allows a secure communication between the dashboard (host) and the Iframe, the message is described by a JSON object containing the action and the value with the payload.

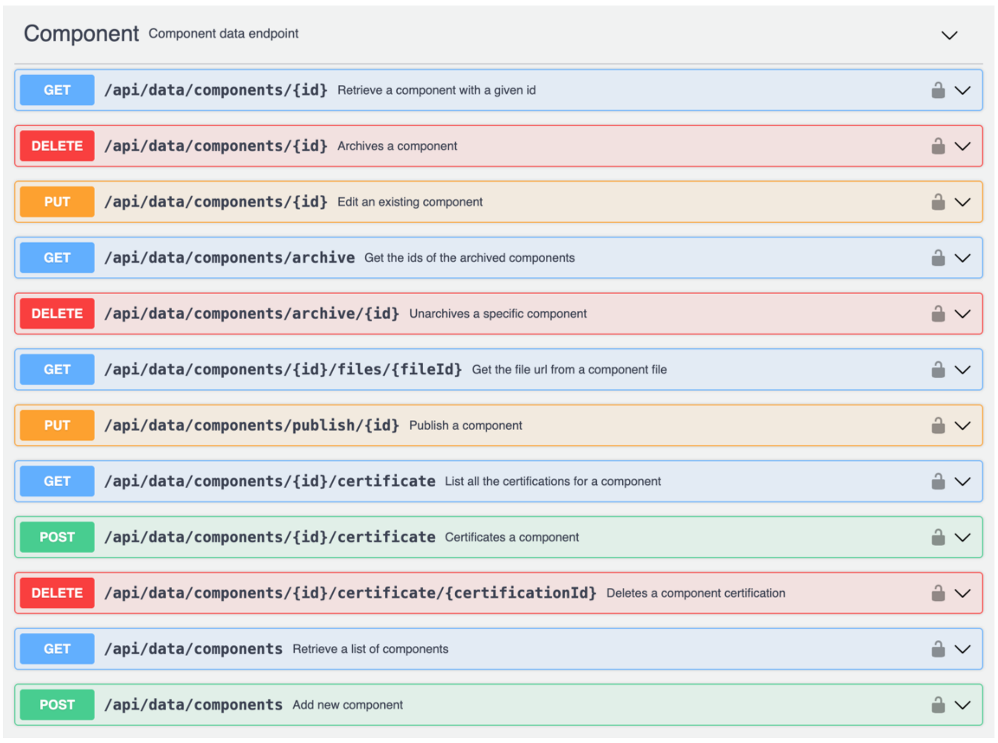
```
{
        "action":<ACTION_NAME>,
        "value":<PAYLOAD>
}
```

The following actions and respective values are currently supported:

- setFilter: Filters the listed components with a mongo filter
- setSort: Sorts the list elements using a mongo sort specifier
- tagFilter: Filters the listed elements using an array of tags
- search: It applies a search filter to the listed elements

3.1.4 Interfaces

FDAC's REST API Swagger displays a list of available endpoints as well as required and optional parameters and example responses. This API can be accessed here. This allows users to easily test out the API and see the available functionalities. The endpoints to edit the assets are listed in Figure 3-17.



Component Component data endpoint

GET /api/data/components/{id} Retrieve a component with a given id

DELETE /api/data/components/{id} Archives a component

PUT /api/data/components/{id} Edit an existing component

GET /api/data/components/archive Get the ids of the archived components

DELETE /api/data/components/archive/{id} Unarchives a specific component

GET /api/data/components/{id}/files/{fileId} Get the file url from a component file

PUT /api/data/components/publish/{id} Publish a component

GET /api/data/components/{id}/certificate List all the certifications for a component

POST /api/data/components/{id}/certificate Certificates a component

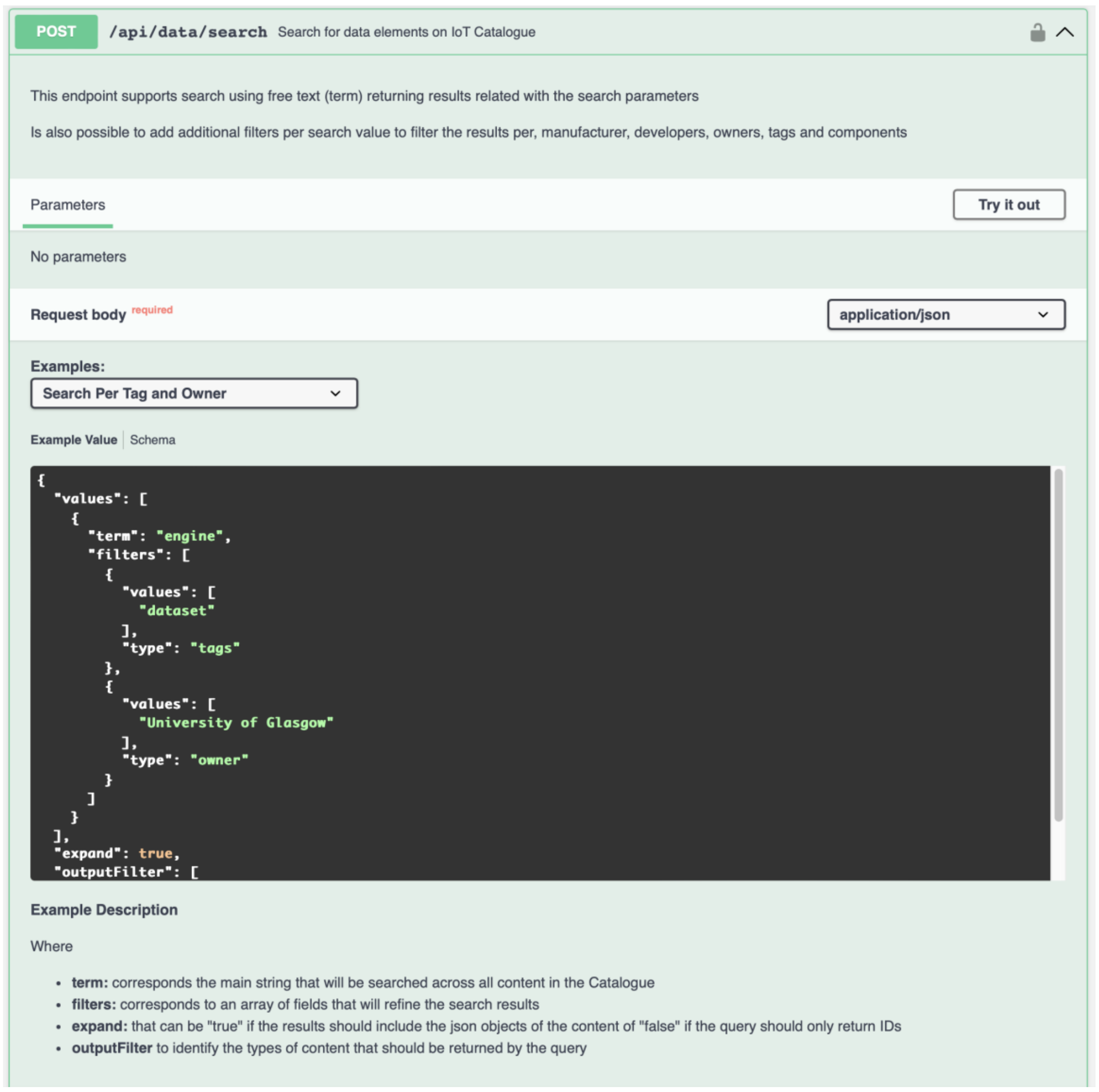
DELETE /api/data/components/{id}/certificate/{certificationId} Deletes a component certification

GET /api/data/components Retrieve a list of components

POST /api/data/components Add new component

Figure 3-17 – FDAC REST API Swagger

The search endpoint, which allows users to submit queries to the FDAC for components that match specific criteria, is shown in Figure 3-18. This endpoint allows users to describe a query by defining a set of terms that describe the intended goal and a set of filters that allow the refinement of the results. A query example is included in the OpenAPI description of the endpoint, shown on the example section of the figure bellow.



This endpoint supports search using free text (term) returning results related with the search parameters

Is also possible to add additional filters per search value to filter the results per, manufacturer, developers, owners, tags and components

No parameters

```
{
  "values": [
    {
      "term": "engine",
      "filters": [
        {
          "values": [
            "dataset"
          ],
          "type": "tags"
        },
        {
          "values": [
            "University of Glasgow"
          ],
          "type": "owner"
        }
      ]
    }
  ],
  "expand": true,
  "outputFilter": [
```

Example Description

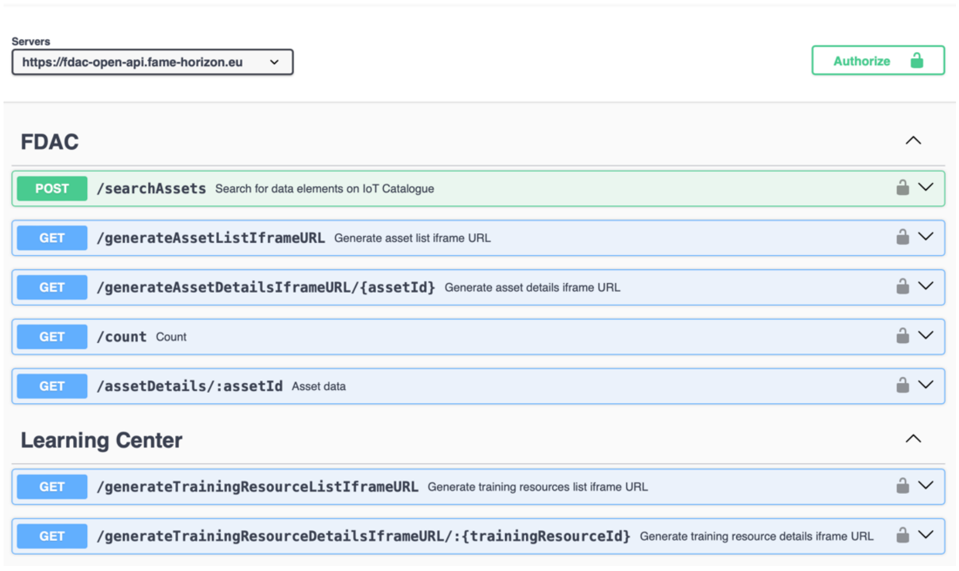
Where

- **term:** corresponds the main string that will be searched across all content in the Catalogue
- **filters:** corresponds to an array of fields that will refine the search results
- **expand:** that can be "true" if the results should include the json objects of the content of "false" if the query should only return IDs
- **outputFilter** to identify the types of content that should be returned by the query

Figure 3-18 - FDAC REST API Search response example

The communication between the FDAC Public API and the dashboard is done through a rest API exposed on the FDAC Public API, during the communication an optional JWT token can be defined to obtain data specific to an user otherwise only public data will be returned, the response of the API is used to build the Iframe which will display FDAC data on the dashboard.

The swagger on image below (Figure 3-19) contains the endpoints that are used by the dashboard when interacting with the open API. The responses of this API serves two purposes, to generate URL’s in string format which will be used when the Iframe is being launched in the dashboard and to generate JSON data (for example the counting) which will be used by dashboard.



POST /searchAssets Search for data elements on IoT Catalogue
GET /generateAssetListIframeURL Generate asset list iframe URL
GET /generateAssetDetailsIframeURL/{assetId} Generate asset details iframe URL
GET /count Count
GET /assetDetails/:assetId Asset data
GET /generateTrainingResourceListIframeURL Generate training resources list iframe URL
GET /generateTrainingResourceDetailsIframeURL/:{trainingResourceId} Generate training resource details iframe URL

Figure 3-19: Open API description of FDAC Public API

3.2 Semantic Interoperability Middleware

3.2.1 Description

The *Semantic Interoperability* component consists of a set of functionalities that enhance the *FDAC* capabilities with semantic functionalities, aiding the process adding new assets when the information is described on other formats than the one internally supported by FDAC.

In an initial approach these functionalities are mainly applied to the asset's metadata, referring to the support of multiple ontologies to describe the concepts used by the *FDAC* API. This allows to expose of API methods that support the description of the information in other formats than the ones used internally in *FDAC*.

In a second approach are explored LLM related technologies to collect and interpret human readable information, translate it to the data model used by FDAC and automatically index it in FDAC database.

3.2.2 Initial approach: Semantic Similarity

3.2.2.1 Related Work

The Semantic Interoperability Middleware aims at providing semantic translation functionalities that allow users from different semantic contexts to use FAME services while using their own semantic concepts. To support such behaviour the Semantic Middleware must be able to support a collection of Semantic Ontologies and technologies that support the execution of semantic translations on-demand, based on the user needs. In this section are presented the background technologies and relevant Ontologies that can contribute to the implementation of the Semantic Middleware.

3.2.2.1.1 Relevant Ontologies

3.2.2.1.1.1 INFINITECH Datapack

The INFINITECH project[3] identified a set of reference ontologies for both the Financial and Legal domain and organized them into the so-called Data Pack. The Data Pack provided the fundamental

[3] https://www.infinitech-h2020.eu/

baseline to allow data modelling while ensuring semantic interoperability between the considered pilots. In some cases, the Data Pack needed to be extended (i.e. newer concepts have been included) to accommodate the pilots' specificities. It contains the metadata in .ttl format and also contains the metadata in two different formats, .json-ld and .owl to ensure the Data Pack is accessible to different communities. The complete Data Pack can be found at http://graph-data-model.infinitech-h2020.eu/content/datapack/. The identified ontologies are:

- **INFINITECH Core Ontology** – The INFINITECH Core Ontology compiles the common vocabularies that are used across different fintech and finance domains in the INFINITECH Project. The objective of the INFINITECH Core Ontology is to summarise the common terms or vocabularies and establish their relationship based on similarities or equivalences.
- **Traffik Analysis Hub Ontology (TAHO)** – The Traffik Analysis Hub Ontology (TAHO) is an ontology whose aim is to represent the Traffik Analysis Hubs data as a semantic model.
- **Financial Industry Global Instrument Identifiers (FIGI) Ontology** – The Financial Instrument Global Identifier (FIGI) is a unique, persistent twelve-character string that serves to identify financial instruments. Along with the identifier, several related data points are identified and defined to provide clear context and differentiation of the financial instruments specified by the identifiers. This ontology provides the 'schema' for the identifier and related constructs. Individuals representing the corresponding security types and pricing sources are provided in separate RDF/XML serialized OWL files.
- **LKIF Core** – The LKIF Core legal ontology is a library of ontologies relevant to the legal domain. It consists of fifteen modules, each of which describes a set of closely related concepts from both legal and common-sense domains. The most abstract concepts are defined in five closely related modules: top, place, mereology, time and spacetime.
- **The Financial Industry Business Ontology (FIBO)** – The Financial Industry Business Ontology (FIBO) defines the sets of things that are of interest in financial business applications and the ways that those things can relate to one another. In this way, FIBO can give meaning to any data (e.g., spreadsheets, relational databases, XML documents) that describe the business of finance. FIBO is hosted and sponsored by the Enterprise Data Management Council (EDMC) and is published in several formats for operating use and business definitions. FIBO is a trademark of EDM Council, Inc. It is also standardized by the Object Management Group (OMG). FIBO is developed as an ontology in the Web Ontology Language (OWL). The language is codified by the World Wide Web Consortium (W3C), and it is based on Description Logic. The use of logic ensures that each FIBO concept is framed in a way that is unambiguous and that is readable both by humans and machines.

3.2.2.1.1.2 Data Catalogue Vocabulary (DCAT)

Beyond these Ontologies, there is another one that is important to refer to, the **Data Catalogue Vocabulary (DCAT)**. DCAT is an RDF vocabulary designed to facilitate interoperability between data catalogues published on the Web. Data described in a catalogue can come in many formats, ranging from spreadsheets, XML and RDF to various specialized formats. DCAT does not make any assumptions about the serialization formats of the datasets, but it does distinguish between the metadata of the dataset and its different manifestations or distributions. Data is often provided through a service which supports the selection of an extract, sub-set, or combination of existing data, or new data generated by some data processing function. DCAT allows the description of a data access service to be included in a catalogue. Complementary vocabularies can be used together with DCAT to provide more detailed format-specific information. For example, properties from the VoID vocabulary [VOID] can be used within DCAT to express various statistics about a dataset if that dataset is in RDF format.

In this sense, DCAT enables a publisher to describe datasets and data services in a catalogue using a standard model and vocabulary that facilitates the consumption and aggregation of metadata from multiple catalogues. This can increase the discoverability of datasets and data services. It also makes it possible to have a decentralized approach to publishing data catalogues and makes the federated search for datasets across catalogues in multiple sites possible using the same query mechanism and structure. Aggregated DCAT metadata can serve as a manifest file as part of the digital preservation process.

The original DCAT vocabulary was developed and hosted at the Digital Enterprise Research Institute (DERI), then refined by the eGov Interest Group, and finally standardized in 2014 [VOCAB-DCAT-20140116] by the Government Linked Data (GLD) Working Group. This revised version of DCAT was developed by the Dataset Exchange Working Group in response to a new set of Use Cases and Requirements [DCAT-UCR] gathered from peoples' experience with the DCAT vocabulary from the time of the original version, and new applications that were not considered in the first version.

3.2.2.1.2 Ontology Engineering tools

3.2.2.1.2.1 Methodology for Semantic Models, Ontologies Engineering and Prototyping

Ontologies played a fundamental role in INFINITECH while providing the necessary mechanisms for describing testbeds and pilot application domains while ensuring semantic interoperability between applications. In INFINITECH, a systematic engineering approach was needed to facilitate the design and development of high-quality and, above all, pilot-aligned ontologies to reference top-level ontologies for the domain.

As shown in Figure 3-20, the INFINITECH Methodology for Ontology Engineering shared terminology, definitions, activities and/or steps with the SAMOD methodology. It was an iterative process that aimed at building semantic models and ontologies by applying four steps. It was organized as a sequence of four sequential steps, namely:

1. **Collecting**. This step collected all the information about the application domain. It involved the following tasks and/or activities:
   a. Pilot Analysis: wrote down the motivating scenario, identified user expectations by writing down user stories and clarified everything by using a set of competency questions (User characterization); and
   b. Conceptualization: wrote down domain terminology, glossary of terms and taxonomies of concepts.
2. **Building**. This step built a new Interoperability test case (aka Modelet). The Modelet is a stand-alone model describing the application domain for the considered pilot and/or testbed. The step involved the following tasks and/or activities:
   a. Creation of a stand-alone model for the pilot or testbed describing the relevant aspects of the application domain;
   b. Connection with the top reference ontology(ies). This activity aimed at reusing as much as possible already defined concepts, relations and properties while pruning all the superfluous elements.
3. **Merging**. This step refined the generated modelet with concepts and relations extracted from reference ontologies for the domain to determine more generic domain ontologies. The step involved the following tasks and/or activities:
   a. Merge modelets in the same pilot/testbed;
   b. Merge modelets between different pilots/testbeds within the same application domain;
   c. Refinement of the current modelet;
   d. Merge modelets with reference ontologies;
   e. Merge modelets with pilot-specific dataset schema; and
   f. Implement generated modelets.
4. **Refactoring**. This step provided the final ontology and semantic model as a conceptual schema to be used within INFINITECH. This model delivered the complete description and characterization of the application domain aligned with reference ontologies while enabling any user of the INFINITECH application to seamlessly access diverse ontologies and thus concrete data.
5. **Linking**: This step linked the refactored models to real data while generating the so-called linked knowledge graph.

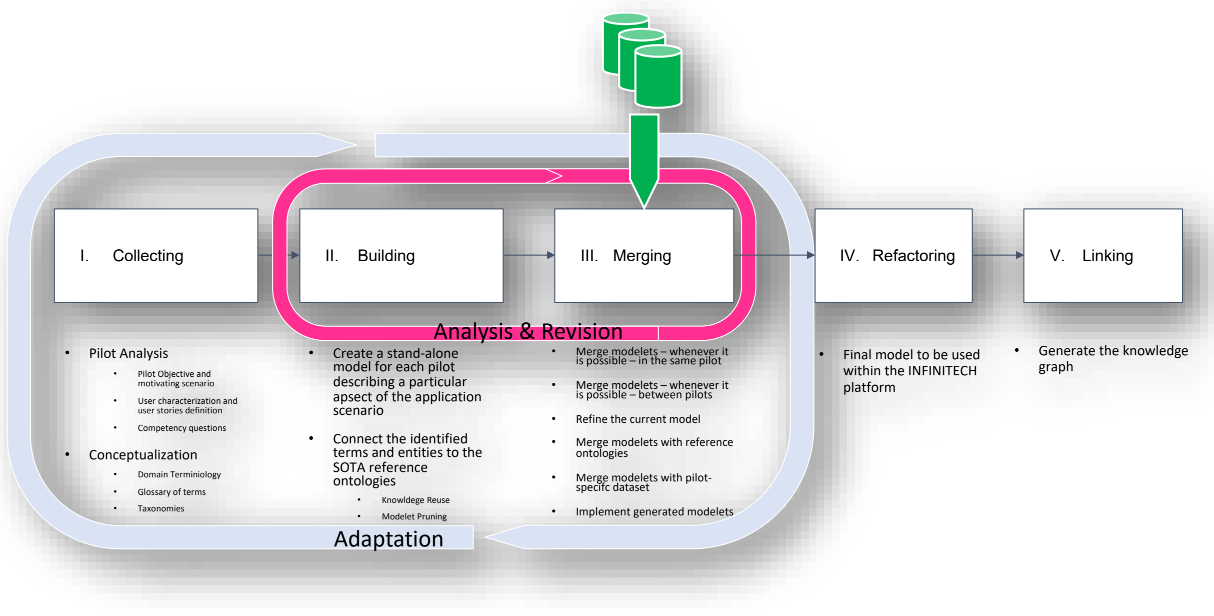
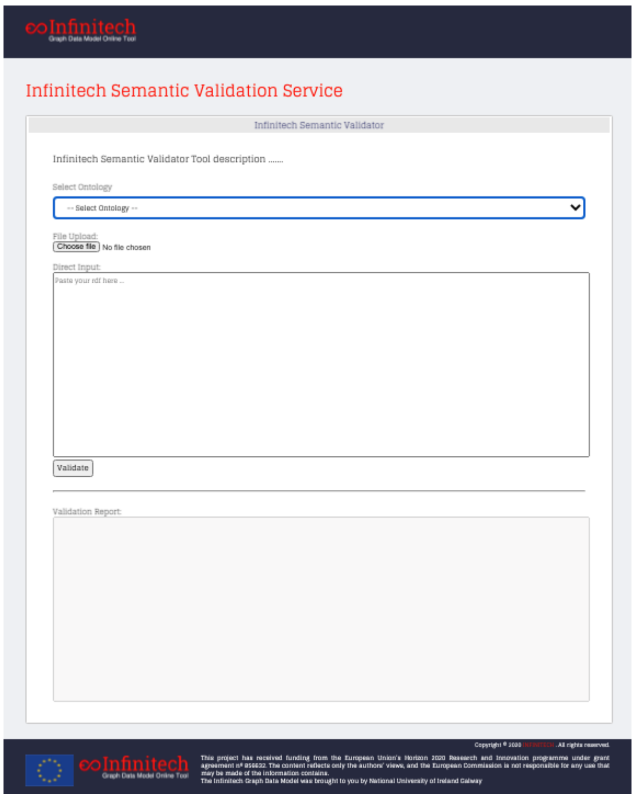

I. Collecting
II. Building
III. Merging
IV. Refactoring
V. Linking
Analysis & Revision
Adaptation
Pilot Analysis
Pilot Objective and motivating scenario
User characterization and user stories definition
Competency questions
Conceptualization
Domain Terminiology
Glossary of terms
Taxonomies
Create a stand-alone model for each pilot describing a particular apsect of the application scenario
Connect the identified terms and entities to the SOTA reference ontologies
Knowldege Reuse
Modelet Pruning
Merge modelets – whenever it is possible – in the same pilot
Merge modelets – whenever it is possible – between pilots
Refine the current model
Merge modelets with reference ontologies
Merge modelets with pilot-specifc dataset
Implement generated modelets
Final model to be used within the INFINITECH platform
Generate the knowledge graph

Figure 3-20 - INFINITECH Methodology for Ontology Engineering

The iteration cycles, Analysis & Revision and Adaptation, were part of the methodology. The Analysis & Revision iteration (executed essentially during the Building step) was aimed at analysing and reviewing the building process to guarantee alignment with the domain expert's expectations and requirements. The result of this step and related iterations was a preliminary model also called modelet. The Adaptation iteration included the steps of Collecting, Defining and Merging and was aimed to refine the generated modelets to cope with new knowledge and/or any change in user characterization, user needs, application domain or, more in general, any change that directly could have an impact on the way domain experts describe their own business and – thus – application domain.

3.2.2.1.2.2 INFINITECH Semantic Validator

The semantic validator (presented in Figure 3-21) allowed the INFINITECH developers to validate their data against the most used financial vocabularies and their related ontologies. Users could upload their data from a file or directly add the data in the provided textbox. The semantic validator service compares the data provided against the selected ontology and the result of the validation gives a validation report that identifies inconsistencies with the data.



Infinitech Semantic Validator Tool description ......

Copyright © 2020 INFINITECH . All rights reserved.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement nº 856632. The content reflects only the authors' views, and the European Commission is not responsible for any use that may be made of the information contains.
The Infinitech Graph Data Model was brought to you by National University of Ireland Galway

Figure 3-21 – INFINITECH Semantic Validator Online Tool

3.2.2.1.2.3 TAG-Tool

Another tool developed at Nova School of Science and Technology[4] is the Translators Automatic Generation Tool (TAG-Tool), which automatically generates translators to support the communication between heterogeneous systems/devices. As presented in Figure 3-22, it receives three files, as input, two XML Schemas (XSDs) of two systems annotated to a reference ontology and the ontology. The tool verifies if these two systems are semantically compatible, that is if it is possible to translate from one to the other. In case they are compatible, the tool returns the translator file in XSLT format.

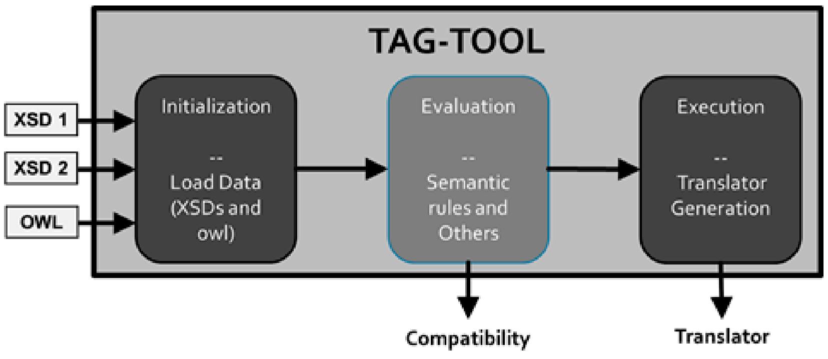


Figure 3-22 – TAG-Tool Interface and Phases

The tool uses both XSDs and the reference ontology to determine, among other things, the matches between the two systems, namely if the provider delivers all the consumer's needs and, if possible,

[4] https://www.fct.unl.pt

to generate the translator that will be used in the exchange of messages between the two systems during their communication. That is, the tool does not translate, it generates a translator that makes it available to be used during communication between the two systems/devices (Figure 3-23).

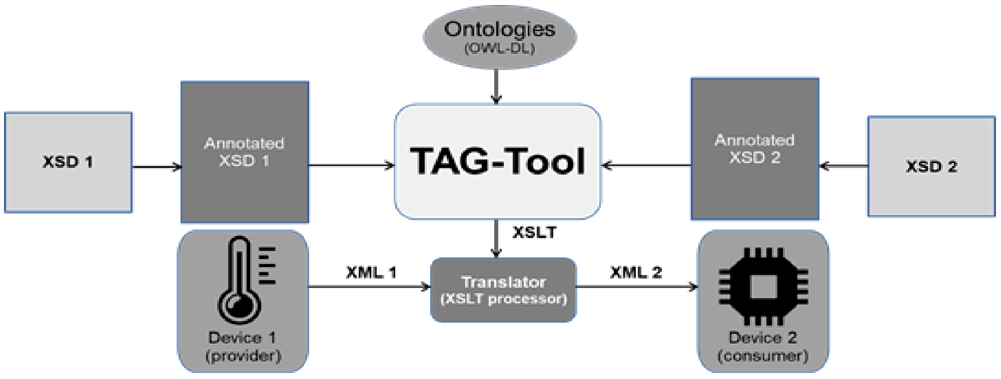


Figure 3-23 – TAG-Tool application scenario

3.2.2.1.2.4 Semantic Interoperability Toolkit for Data Marketplaces

The Semantic Interoperability Toolkit for Data Marketplaces approach was proposed by UNPARALLEL whose main target is to handle data from different sources from a semantic standpoint. This approach was inspired by the scenarios where several data sources with similar information are available on a Data Marketplace but described using different semantic concepts. The proposed approach would allow data consumers to discover relevant information across different data sources by translating the query concepts to the concepts supported by each data source and harmonizing all the results to the semantic concepts used by the data consumer.

The core of the proposed toolkit comprises two components – the Semantic Matchmaking Solution and the Semantic Engine – allowing users to identify similarities between concepts and manage queries against them respectively, regardless of their original representations.

The Semantic Matchmaking Solution approach resorts to Semantic Search algorithms to aid in the process of finding semantic matchings between concepts on different ontologies. Semantic search relies on vector search technologies , which consists of encoding both the universe of results and the user input using the same Machine Learning model (usually a Deep Learning Neural Network) and then returning the results with the highest score (i.e., similarity) . For that purpose, the vectorial operations that are used create a compound formula that outputs the so-called score for each possible result. In the end, a k-Nearest Neighbours algorithm is applied to find the k results with the highest score. For that, the results are sorted by their score in descending order and the k-first ones are returned. This approach is expected to deliver accurate matches because, as it relies on vectorial representations of the data, the algorithm can find semantically equivalent sentences, i.e., similarities beyond the words themselves or the characters that form them.

On the other hand, the Semantic Engine approach implies reaching a solution that supports the execution of semantic data queries starting with the usage of GraphQL to express desirable queries. This language allows users to express queries in an easy-to-understand manner, by using a JSON-like language and providing a broad ecosystem of tools that support the handling and execution of such queries. Moreover, it provides a flexible way for users to define queries and the desirable structure of the query output.

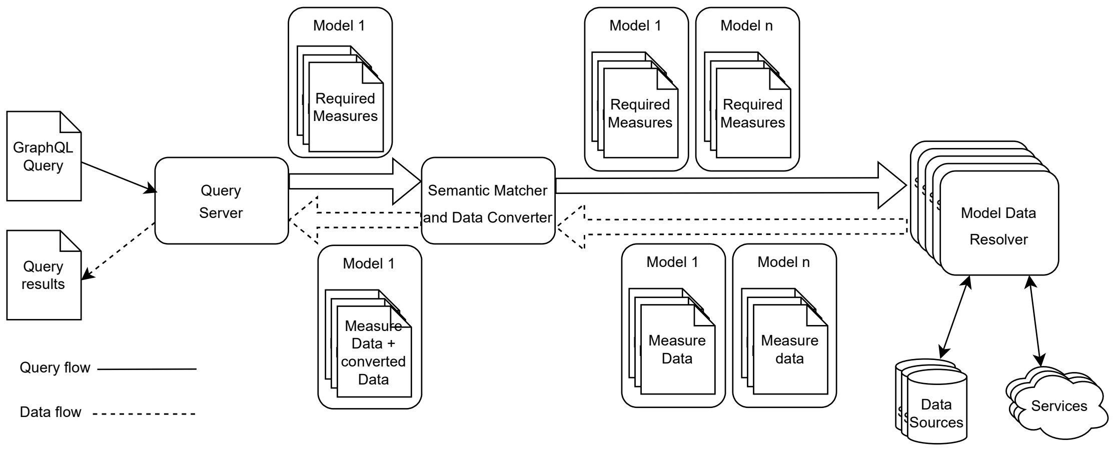


Figure 3-24 - Overview of the semantic query system

This system is composed by three main modules: Query Server, Semantic Matcher and Data Converter, and Model Data Resolver.

The Query Server is responsible for receiving the query request and performing its first processing. It starts by identifying the Data Model used by the user that serves as context for the data requested. Then, performs the usual behaviour of a GraphQL server and analyses the query, extracts the Measurable Quantities, and analyses any filtering or constraint information that applies to the collection of data from that measurable content and any operation that should be applied to the collected datasets. At the same time, this module is responsible for the data aggregation of the query response according to the structure defined in the GraphQL query. Before providing it to the query requester, any operation requested in the query is applied to the data.

The Semantic Matcher and Data Converter are responsible for performing the semantic matching between different data models and the execution of any data conversion required when matching data between different data models. When the Query Server analyses the GraphQL query, it extracts all the Measurable Quantities and any limitations that should be applied during the data discovery process. After that, the Semantic Matcher will receive the Measurable Quantities and discover their correspondence (if it exists) on other Data Models. This process is represented in the figure below.

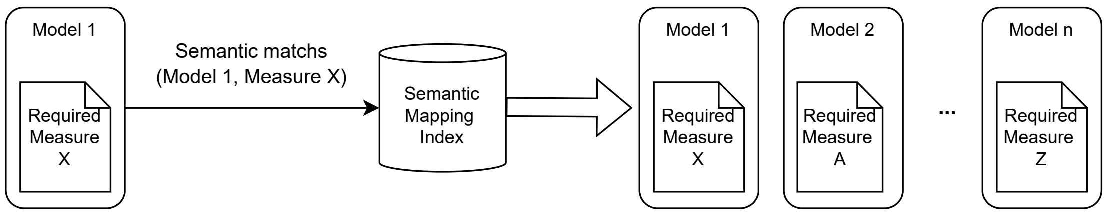


Figure 3-25 - Semantic matching discover process

After the generation of the different query equivalent representations on other data models, carried out by the Semantic Matcher and Data Converter, the original query and the equivalents must be executed, where the relevant information must be collected from data sources represented in each data model. The process of query execution is carried out by a component called Model Data Resolver. This flow is represented in Figure 3-26. This resolver implements the logic associated with the process of collecting data from a specific data source and presenting it according to a specific Data Model.

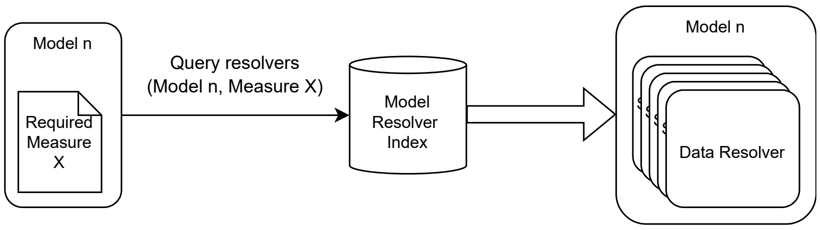


Figure 3-26 - Detail of data resolver discovery process

According to the paper Semantic Interoperability Toolkit for Data Marketplaces despite the potential of the proposed approach it has a major drawback as it requires the information to be enriched beforehand. This means that if the user searches for a term that the Semantic Matchmaking solution did not map with other semantic concepts, then the Semantic Engine will not be able to match the user input with the corresponding data contained in the database.

Advancements beyond the Related Work

In the context of FAME, capabilities of the related work technologies are expected to be extended to support the automatic generation of Model Mappings and the automatic execution of mappings.

As new semantic models are added to the system, support should be provided to enable users to insert and request information using their preferred semantic concepts. Semantic mappings should be automatically created to maximise the number of semantic options available for the user with reduced waiting time after the addition of a new Semantic Model to the system.

Whenever a user wants to access information using a specific semantic model, the information described on other semantic concepts needs to be translated in real-time to answer the query of the user as fast as possible to support. Such functionality requires automatic execution of the model mappings.

3.2.2.2 Technical Specification

3.2.2.2.1 Component-level C4 Architecture

The semantic middleware works along the FDAC and provides it with the capability to support the description of information using different Semantic Models. These functionalities may be used by users when they want to discover/add assets using a specific data model, different from the one used by FDAC, or to support communication with external systems using the semantic model supported by the external systems. The Architecture of the Semantic Middleware is presented in Figure 3-27.

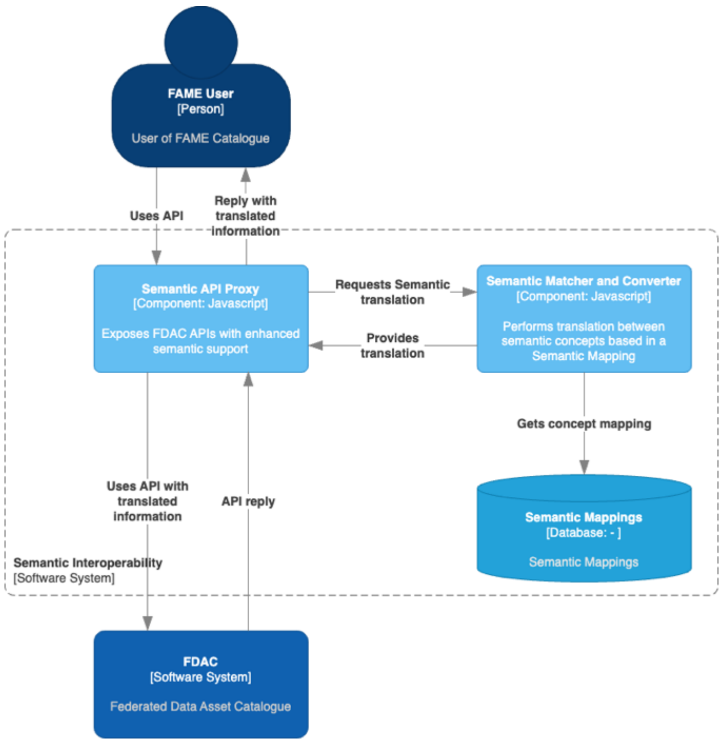
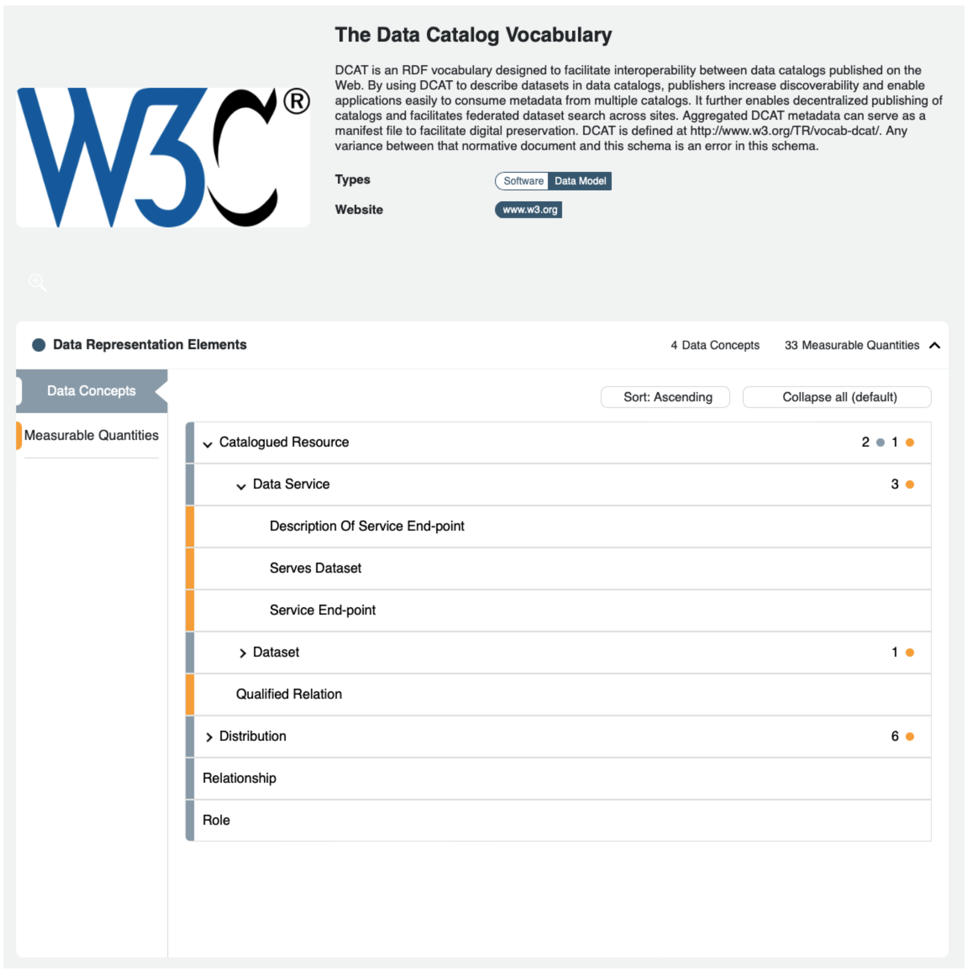

FAME User
[Person]
User of FAME Catalogue
Uses API
Reply with translated information
Semantic API Proxy
[Component: Javascript]
Exposes FDAC APIs with enhanced semantic support
Requests Semantic translation
Provides translation
Semantic Matcher and Converter
[Component: Javascript]
Performs translation between semantic concepts based in a Semantic Mapping
Gets concept mapping
Semantic Mappings
[Database: -]
Semantic Mappings
Uses API with translated information
API reply
Semantic Interoperability
[Software System]
FDAC
[Software System]
Federated Data Asset Catalogue

Figure 3-27 – Architecture of Semantic Middleware

This architecture is composed by three (3) components: Semantic Mappings repository, Semantic Matcher and Converter, and the Semantic API Proxy. These components are described in the following subsections.

3.2.2.2.2 Semantic Mappings repository

This component is responsible for indexing and storing representations of the Semantic models. This repository stores not only the metadata describing the Semantic model but also models the semantic concepts and corresponding relations. Since Semantic Models can be considered assets themselves, they can be indexed on the FDAC itself, acting FDAC as the Semantic Mappings repository. Semantic Models are then described using the FDAC data model, being represented by the name, description, logo, and website of a Data Model. Figure 3-28 represents an example of how a Data Model (DCAT) is represented and shown in the FDAC user interface. In this figure is possible to see an example of how the semantic concepts are modelled and represented in the web-interface. In a tree visualisation here, the semantic concepts are presented and shown the hierarchical relation between them.



The Data Catalog Vocabulary

DCAT is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web. By using DCAT to describe datasets in data catalogs, publishers increase discoverability and enable applications easily to consume metadata from multiple catalogs. It further enables decentralized publishing of catalogs and facilitates federated dataset search across sites. Aggregated DCAT metadata can serve as a manifest file to facilitate digital preservation. DCAT is defined at http://www.w3.org/TR/vocab-dcat/. Any variance between that normative document and this schema is an error in this schema.

Types	Software Data Model
Website	www.w3.org

Data Representation Elements — 4 Data Concepts — 33 Measurable Quantities

Data Concepts | Measurable Quantities

Sort: Ascending | Collapse all (default)

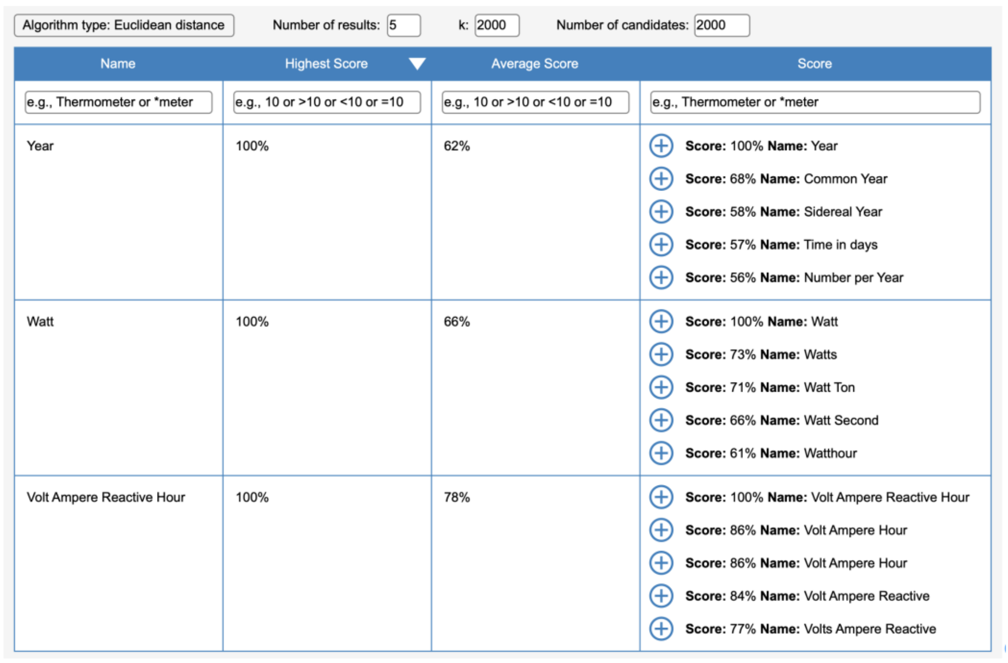
Catalogued Resource	2 ● 1 ●
Data Service	3 ●
Description Of Service End-point	
Serves Dataset	
Service End-point	
Dataset	1 ●
Qualified Relation	
Distribution	6 ●
Relationship	
Role	

Figure 3-28 – Example do DCAT model represented in FDAC!

3.2.2.2.3 Semantic Matcher and Converter

This component is responsible for the generation of the mappings between Semantic Models and the execution of semantic mappings on-demand. The approach for the generation of the mappings between the concepts of different semantic models resorts to finding the mapping between different concepts using techniques of semantic similarity. These techniques process the name and description of each semantic concept, using language processing models. These models, called Embedding Models, convert the information from the semantic concepts into vector embeddings. Similarity between concepts can be calculated based on the distance/proximity between the semantic vectors.

For each semantic model, each semantic concept was analysed and embedded in the model to create the corresponding vector. Vectors of one semantic model were compared with vectors of the different semantic models to identify similar concepts in different semantic models. Similarity functions provide a score to classify the similarity between vectors. A web form has been used to analyse and compare the results, as presented in Figure 3-29. In this form are shown the best semantic concepts of other models that have the highest score.



Algorithm type: Euclidean distance Number of results: 5 k: 2000 Number of candidates: 2000

Name	Highest Score	Average Score	Score
e.g., Thermometer or *meter	e.g., 10 or >10 or <10 or =10	e.g., 10 or >10 or <10 or =10	e.g., Thermometer or *meter
Year	100%	62%	**Score:** 100% **Name:** Year **Score:** 68% **Name:** Common Year **Score:** 58% **Name:** Sidereal Year **Score:** 57% **Name:** Time in days **Score:** 56% **Name:** Number per Year
Watt	100%	66%	**Score:** 100% **Name:** Watt **Score:** 73% **Name:** Watts **Score:** 71% **Name:** Watt Ton **Score:** 66% **Name:** Watt Second **Score:** 61% **Name:** Watthour
Volt Ampere Reactive Hour	100%	78%	**Score:** 100% **Name:** Volt Ampere Reactive Hour **Score:** 86% **Name:** Volt Ampere Hour **Score:** 86% **Name:** Volt Ampere Hour **Score:** 84% **Name:** Volt Ampere Reactive **Score:** 77% **Name:** Volts Ampere Reactive

Figure 3-29 – Web form used to analyse the results from Semantic Similarity

Several models were used to generate the vector embeddings and math operations to find calculate the semantic similarity between the vectors. For the embedding model, we used pre-trained Machine Learning Algorithms to process human language based on BERT. For the similarity functions there have been considered: i) the Euclidian distance between 2 vectors, ii) the Dot Product of two vectors and iii) the Cosine of the angle between two vectors.

While the generality of the approaches tested presented promising results that can support a human operator in the identification of the equivalent semantic concepts, the current results do not allow to automate the generation of mappings between Semantic Models.

The next steps will focus on exploring the usage of domain specific models, refined to better understand the meaning of each concept. Instead of using a Model pre-trained with the generic English language, use models pre-trained with technology-oriented resources that may produce more accurate assessments. Another approach that may be worthy to be tested is to include the semantic path of each concept in the computation of the similarity. These semantic paths represent a hierarchy between the semantic concepts that may be used to further define the semantic meaning.

3.2.2.2.4 Semantic API Proxy

The Semantic Proxy exposes to users and other systems a REST Interface capable of other semantics than the one used by FDAC. This proxy uses the mappings created in the *Semantic Matcher and Converter* component to expose new model options on-the-fly as new model mappings are added. When the endpoints of this Proxy are invoked, the *Semantic Matcher and Converter* component is also called to execute the mapping between the model selected by the user and the FDAC model, and vice-versa.

In the current version of this document, only the bidirectional mapping between the FDAC model and DCAT is presented. Others are expected to add new systems and pilots start using the FDAC interface and require the support of different data models.

3.2.3 New Approach: Usage of Large Language Models (LLMs)

3.2.3.1 Approach shift

The previous approach for the Semantic Interoperability Middleware between external asset's metadata and the FDAC asset functionalities relies on semantic similarity to expose interoperability FDAC API endpoints to support descriptions/information using data models others than the internal model used FDAC. Semantic similarity approach was a first step to seamlessly connect unknown external data models with FDAC by analysing concepts of external data models, their names and descriptions, to attempt to automatically map this extracted information with FDAC data model concepts. This map is then used to support new endpoints that "understand" the new data model, facilitating the addition of new components within the FDAC that are already described in that data model. In first approach towards the semantic support of external data models some limitations were discovered:

- Since Semantic similarity relies heavily on concepts' descriptions, the level of detail for each description (or lack of it) directly affects the results obtained;
- Inaccurate results derived from semantics differences of the language used to describe the data models; and
- Human error by misinterpretation when filling model parameters, by filling fields with incorrect information and giving an incorrect description for the model concepts.

Those limitations lead to dependency on available API frameworks to fetch data and adapt to different APIs manually, but also to inaccurate mappings based on human errors misinterpretation or the semantic in the descriptions provided.

With the strengths and opportunities that have been merging from LLM's, a different alternative for interoperability of tools can be adopted in the solution, shifting the approach to include the recent and popular LLM's capabilities. In addition, with other AI tools, LLMs can enhance the connection between external data and FDAC API functionalities. Instead of solely relying on semantic similarity, it can provide a prompt-based approach to interpret textual descriptions and generate output that aligns with the required FDAC parameters.

This new approach identifies new opportunities for implementation with characteristics such as:

- **Flexibility** by being capable to use context from a wider range of resources, instead of relying only on static data models.
- **Better contextual understanding** by using LLM functionality to interpret phrases semantic but also logic and ideas.
- **Automated context search** using real-time web data research, reducing the overall manual effort.

The new approach for the Semantic Interoperability, represented in Figure 3-30, starts with user input. The basis is of this approach is that after performing the onboarding process and getting the authentication credentials, a user can request FAME to add a new asset based on existing information. This information can be either in the format of a document or can be a link to a marketplace containing the assets to be federated into FAME. The semantic middleware will then use AI tools to generate queries related to the asset and search in the online website for information and characteristics of assets. Simultaneously, FDAC data model is used to understand which parameters inputs should be used to describe new assets according to the FDAC data model. Large Language Models (LLMs) are used to understand the semantics of the asset descriptions and generate values contextualized with official online information. The values generated, after being represented in FDAC Data Model, are inserted as an asset into FDAC using the available REST API by the means of a dedicated Model Context Protocol (MCP) server. The key feature of the new approach is adding an automation layer for the information gathering and decision-making, filtering relevant information and using LLM to generate values for parameters using that information. This process includes not only possible contextualization from the asset itself, but information can also be automatically enriched using resources available online, such as official documentation and characteristics described in trusted websites.

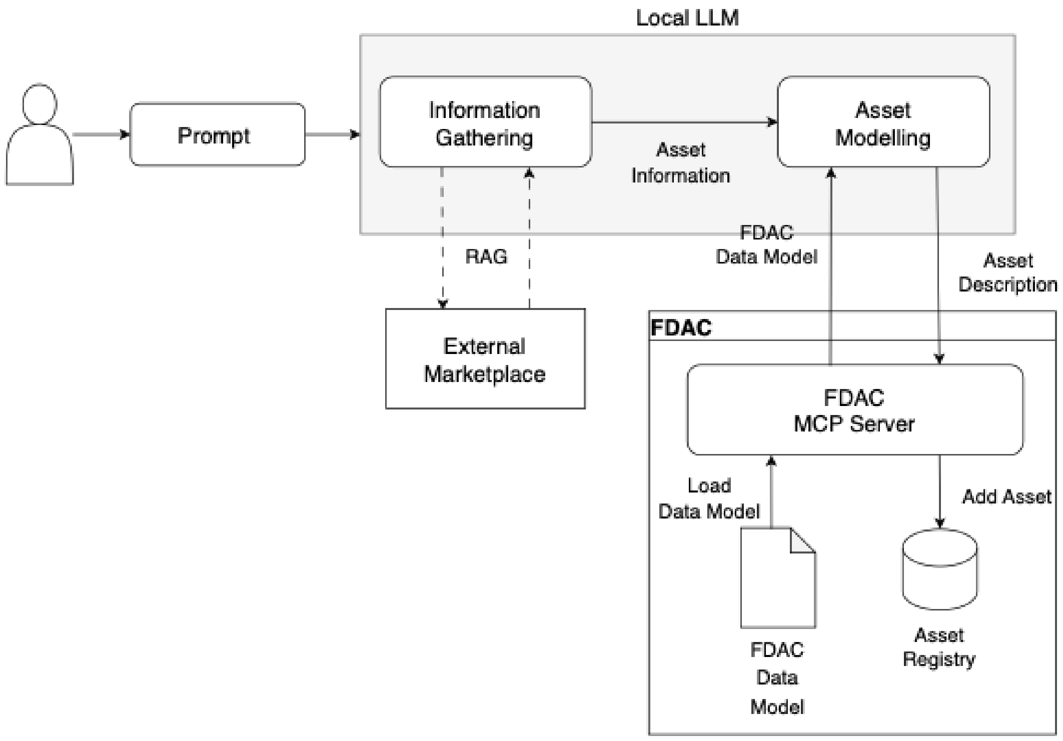


Figure 3-30 – Diagram representation of the new approach

3.2.3.2 Related Work

3.2.3.2.1 Large Language Models (LLMs)

Large Language Model (LLM) is a technology derived from artificial intelligence developments related to the natural language processing, which is the representation of words by their meaning and semantics. These are big models with very extensive amounts of text data encrypted that can generate human-like text based on the training examples used for that model. The amount of data used during training is what makes LLM to be very storage demanding but also understand the complex relations from human language with high accuracy. The most common models are already pretrained for the purpose to know how to answer questions and create a dialogue between a user simulating a conversation between two humans where the LLM has the knowledge to answer to a wide range of topics, which further could be improved or specialized with additional training to the model. This additional training creates sub-models that can be smaller and specialized to a specific desired task. From the article, this pre-training improves performance on a wide variety of tasks, such as BERT with capabilities to be trained for sentence classification, question answering or entity recognition. The most popular approach that spiked the research for applications of LLMs was the appearance of ChatGPT and the promising accuracy and quality of question answering. The key distinction between LLMs and earlier generative AI systems lies in their linguistic capabilities, where LLMs are designed as general-purpose tools with a broader world knowledge, including diverse domains such as biology, electronics, finance and even political issues.

This AI technology showed interest in different research areas such as:

- Medicine and Healthcare: In the domain of medicine, tests have been made for patient care, research and for medical education. LLMs shows good results in simplifying complex medical language to patients, assisting in communication between doctor and patient. In the research area, LLMS demonstrated to aid navigation through the scientific literature and summarizing and highlighting key findings. However, LLMs are not updated in real-time and can be missing the most recent and up to date information from new papers and articles. The biggest concern demonstrated in this article is the weak reproducibility that these models have, where the same prompt can generate two different patient case evaluations that highlights both correct relevant procedures but different approaches, which complicates to ensure consistency and transparency in the medical procedures.
- Education: The article highlights LLMs to support not only students but also teachers, by demonstrating strong performance in tasks like essay scoring, feedback generation, question creation, and personalized learning adapted to the learning rate that the student wants. However, the lack of transparency from LLMs to demonstrate the reasoning and where is the source of knowledge still concerns on the reliability for students to interpret LLM responses as factual true knowledge or biased/deviated judgements, or even incorrect responses (hallucinations).
- Engineering: In the engineering research area, LLM showed to be a useful tool not only for software engineers but for basic arithmetic and scientific reasoning based on scientific rules and principles. LLM has great results at tasks like code generation, debugging and documentation of coding, demonstrating a robust knowledge about coding logic and different types of programming language. Additionally, most LLMs nowadays support frameworks like tool calling that enables interaction of LLMs directly with external APIs, enhancing LLMs autonomy capabilities and utility in engineering contexts. The main concerns from LLMs in engineering are similar to other areas, where the reliability in the responses can't be strong since these models are sensitive to the prompt given as input that generates different possible responses but also some responses can also be hallucinations and need additional user interaction to align the flow of reasoning from the model.

In order to use LLM tools and functionalities, there are different options to consider according to several factors of the project, where the alternatives range from cloud services or local environment deployment of LLMs. Since LLM demand huge amounts of computer resources and energy, cloud service alternatives can be a solution to avoid expenses on setting up an environment ready to run LLMs and build a specialized infrastructure for AI tools, where big companies such as OpenAI or Microsoft can handle the infrastructure and its expenses while exposing the AI functionalities as services to be used by customers. The other alternative is running locally and manage a custom infrastructure that can handle custom and open-source tools in development, which usually is the approach using in research environments consider the full control and flexibility to adapt not only the hardware, but the software associated to those tools. Another crucial reason to run LLMs locally is to avoid using external services with sensitive data, with the guarantee that the information cannot be exposed outside of the project environment. For this case, it is important to run the models locally and guarantee full control and flexibility on the software architecture and not restrain the project to a specific environment, but also there will be sensitive data handled during the workflow that should not be available to members outside of the project. Therefore, it is important to analyse what frameworks are available to deploy a local custom environment to run LLMs and other AI functionalities.

3.2.3.2.1.1 HuggingFace

HuggingFace is a well-known leading platform for running open-source LLMs that offers tools and services for both development and deployment. As the core, it is composed of Transformers library that supports thousands of pre-trained models with inputs and outputs that can be text, image, audio or metadata. HuggingFace also provides tools to fine-tune the available models with custom datasets and publish those custom models into the HuggingFace models repository, supporting a community driven database of models. For the deployment, HuggingFace provides integration both for LLMs that are not open-source and for custom deployment in local machines. It is also possible to create multimodal agents to enable LLMs to interact with tools like image generation and web search. Most recent updates allow to implement complex any-to-any multimodal models that can accept any type of input and output any type of data, ranging from text to audio and images. It also simplifies training with tools like AutoTrain, PEFT for parameter efficient fine-tuning, and Accelerate for distributed training, making it a state-of-the-art ecosystem for LLM experimentation and production.

3.2.3.2.1.2 Ollama

Ollama is a popular framework for running LLMs locally with remarkable simplicity and efficiency. Designed for deployment, Ollama enables seamless deployment of open-source models on different operating systems or containerized environments using simple command-line interface. Ollama also support model quantization to reduce memory usage and GPU acceleration automatically without sacrificing performance, which is crucial for environments with restricted computer resources and to abstract deployment from the complexity of managing these hardware resources.

The Ollama is a robust solution deployment of LLMs in any software environment thanks to the simplicity, lightweight framework and automatic handling of resources. This shifts the focus of deployment to the selection and experimentation of different models and customization of LLMs using prompt engineering. The downside from the alternative HuggingFace is the lack of support to an abstract model, since supported models must be included in the Ollama library in order to be deployed. Ollama also does not support fine-tuning of LLMs, restraining the customization of models by prompt engineering and other simple alternatives.

3.2.3.2.1.3 LangChain

LangChain is a framework designed to simplify applications powered by LLMs that require reasoning, memory, and tool calls. It provides a modular architecture that allows developers to build LLM workflows and enable LLMs' functionalities for programming environments, such as memory storage, output formatting and external tools calling like APIs. One of the main features of LangChain is to support retrieval-augmented generation (RAG), enabling LLMs to access and reson over external knowledge sources in real time. It also integrates seamlessly with vector databases, agents, and cloud services, making it ideal for building chatbots. LangChain excels as an additional tool to complement LLM usage, helping to integrate with external resources and other AI applications in Python. However, the tool can be too complex for new developers and not the best solution to simpler solutions where it is only needed to use the text generation functionality without any chat memory or usage of tool calls.

3.2.3.2.2 Model Context Protocol (MCP)

The Model Context Protocol (MCP) is a standardized communication framework designed to bridge LLM's with external tools, APIs, and contextual data sources. As described in MCP documentation, MCP can be similarly interpreted as the USB-C port for AI applications with external services. The core key components from MCP architecture includes:

- **MCP Host:** The MCP host involves the application that provides the environment to run MCP clients and the AI logic behind the main objective of integrating MCP standard in the project. As for example, Claude Desktop includes an AI agent architecture to implement a chatbot with a MCP that can search through exposed MCP servers running in the network and decide which tools from the MCP server are relevant to answer user's prompts.
- **MCP Client:** The MCP client is the interface layer between the AI application that uses LLM and the MCP servers available. MCP clients enables the functionality to request searching for available MCP server and description of their tools for a seamless interaction of LLM text generation capabilities with external contextualization.
- **MCP Server:** The MCP server is the component that exposes the external services as server's tools to be accessible by MCP clients. These components need to have complete and robust description of the different tools in order for LLMs and AI agent orchestrator to understand and decide on what resources can be extracted from the tools exposed.

The MCP standard can use the most common and popular communication mechanisms and transport layer, such as JSON-RPC, STDIO or HTTP to communicate between MCP clients and servers. Also to help in orchestration and defining consistent and high-usage MCP servers, a concept called Prompts creates reusable prompt templates and workflows for processes that are complex and extensive but are also systematically used by MCP clients, avoiding repetitive manual definition of those cases. Orchestration can also be enhanced by the concept Sampling that allows MCP servers to request for completion of information or metadata before executing its task, allowing a sub-orchestration from the server side to ask for additional input and enabling more sophisticated agentic behaviours.

The MCP standard capacities addresses a key limitation of LLMs of their dependency to maintain persistent context by interacting seamlessly with external systems, each one with a different integration layer API to expose its services. MCP enables LLMs to operate more like intelligent agents by providing a structured way to manage session data, integrate external information and dynamically call external services. MCP enhances LLM tool calling capabilities by maintaining a continuity with external systems and therefore introducing persistent context layer to allow LLMs to remember user preferences and adapt the tool usage according to the interaction between the user and the LLM.

3.2.3.2.3 AI orchestration platforms

AI orchestration coordinates and manages AI models and services in a system, including deployment, implementation or the integration from a software workflow with AI tools. The orchestration can involve the integration of AI in automated processes with usage of large quantities of data transfer during different steps of the pipeline. Those tasks involve AI automation to reduce human intervention on non-predefined tasks. Specifically for this case, the most relevant implementation of AI orchestration is through AI agents that can autonomously perform tasks based on a pre-designed workflow with conditional paths that are decided by the AI agent itself without requiring a human decision to analyse the best workflow path for each scenario.

Agentic architecture orchestration has several important roles that must be addressed by one or several agents: Planning, Services Execution, Validation and Self-Replanning. These roles ensure the system can set plans according to goals, execute the steps to achieve those goals and self-evaluate the results obtained before giving the final output. There are several architecture workflows possible according to the task that the system is solving, ranging from how controlled and autonomous the system is desired to make decisions by its own and one example is illustrated in the Figure 3-31.

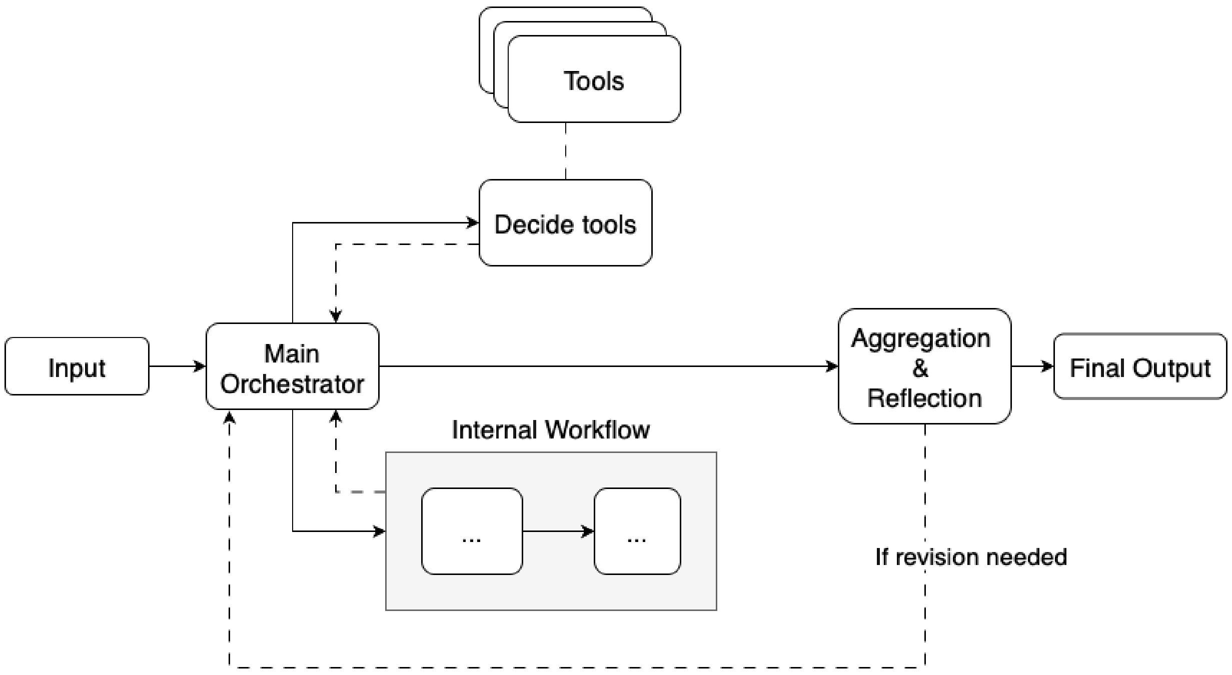


Figure 3-31 – Example of AI agent architecture with main orchestrator component

The LLMs take a crucial role on AI agentic based architecture implementation, since these are the models used for the decision-making. The evolution of LLMs and agentic architecture was mainly possible thanks to both technologies new developments aiding to propose new solutions, such as LLMs now having the possibility to have extra features like memory and API calls usage while agent architectures can have dynamic reasoning, enabling to adapt plans in the middle of the workflow in real-time by evaluating possible results obtained. With that in mind, agent platforms have different integrations for LLMs, and their specific functionalities should be taken into consideration when selecting the platform to start building an orchestration system based on AI agents.

3.2.3.2.3.1 CrewAI

CrewAI is an open-source framework designed to facilitate the orchestration of multiple AI agents working collaboratively towards a main goal. Its main characteristic is its approach of a role-based architecture where each agent is assigned a specific function, giving a high-level vision for the management and delegation of tasks for each agent. Additionally, agents can be integrated with external tools by using pre-defined functionalities to easily adopt this functionality into the system.

This tool strengths focus on the intuitive design compared to other alternatives that enables to easily implement simple AI agent orchestration based on delegating what tasks are needed and what type of agents are needed and how they should communicate between them and design the main workflow. However, by using pre-defined agents, it loses the full control of each agent execution flow, and some logic can't be easily modified to tweak minor changes in the logic of the agent. This problem implies possible additional logic implementation for more specific case scenarios or to create a big and complex architecture.

3.2.3.2.3.2 OpenAI Agents SDK

The OpenAI Agents SDK is a developer-friendly framework designed to simplify the creation and deployment of AI agents using the OpenAI environment tools available. The platform provides a good control of the architecture while abstracting the implementation and other tool integration concerns, amplifying the focus of the creation to the architecture structure and the goal-driven agents. Additional key concepts such as Handoffs mechanism and Guardrails contribute to create complex flows that need to transfer decisions control between agents and optional validation layers to ensure robust decision making during the workflow.

The strengths of this relies on the simple and fast prototyping and deployment of the whole architecture, thanks to the platform being inside OpenAI ecosystem that already has another powerful AI tools to monitor the system and to integrate other AI functionalities from OpenAI. The encapsulation inside of the OpenAI ecosystem can be also a problem for most projects, where it locks the AI functionalities to OpenAI tools only, with harder integration with open-source tools and unknown APIs. Also, the abstraction approach from the tool shows good control from the architecture behaviour but not full control for agent processes and logic, being a tool less suited for dynamic workflows that need dynamic branching logic.

3.2.3.2.3.3 LangGraph

LangGraph is an advanced orchestration framework designed to model complex, stateful, and multi-agent workflows using a graph-based architecture. It enables developers to define AI applications as graphs, where graph theory can be applied for this tool. Each node represents a step in the process that can be an agent, a tool, or a function, and each edge represents the workflow and branching logic behind the node connections in the architecture. LangGraph is inserted in an environment with tools that can integrate open-source LLM alternatives and a framework to analyze and monitor graphs' behaviour and workflow.

LangGraph is the tool that demonstrates the best control over the logic and behaviour of the entire graph, since nodes and edges logic are designed during the architecture implementation, giving control over conditional edges logic and the data workflow. Some other advantages include the ease to integrate open-source or custom tools into the architecture and the possibility for visual debugging to monitor complex workflows. The drawbacks from this approach are the complexity involving to setup and plan simple architectures, leading to an extensive time-consuming process to setup relevant tools and adapt to the steep learning curve.

3.2.3.2.3.4 SmolAgents

SmolAgents is a lightweight, open-source AI agent framework alternative developed by Hugging Face to build minimal tool-using agent's architectures. This tool emphasizes on simplicity, efficiency and straightforward code execution to enable fast prototype and deployment, including functionalities for easy integration of any type of LLMs and in any type of operating system.

This tool simplicity is suited for simples open-source projects with flexibility for various possible environments alternatives while maintaining high efficiency on performance and time efficiency. However, it faces more challenging problems when used to implement complex workflow, since it does not define the architecture as an abstraction model to ease the management, like graph-based or role-based frameworks.

3.2.3.2.4 Search and Retrieval of Information

In today's AI-driven landscape, the ability to access and process real-time information from the web is becoming increasingly vital. Large Language Models (LLMs), while powerful, are inherently limited by the static nature of their training data. To overcome this, developers are turning to web search tools that can dynamically retrieve, filter, and structure information from the internet. These tools are especially important in Retrieval-Augmented Generation (RAG) systems, where external knowledge is used to ground model outputs, reduce hallucinations, and improve factual accuracy.

Modern web search APIs are no longer just wrappers around search engines. They are intelligent systems, offering semantic search, summarization, content extraction, and integration with agent frameworks. This subsection explores several of these tools, analysing their core functionalities and highlighting real-world use cases that demonstrate their value in practical applications.

3.2.3.2.4.1 Exa

Exa is a web search API designed specifically for integration with large language models (LLMs) and AI agents. Unlike traditional search engines, Exa is optimized for real-time information retrieval, semantic filtering, and multi-hop reasoning, making it particularly well-suited for use in Retrieval-Augmented Generation (RAG) pipelines and intelligent agent frameworks.

One of Exa's key strengths lies in its ability to understand and process natural language queries, returning results that are not only relevant but also structured in a way that LLMs can easily consume. It supports both neural and keyword-based ranking, allowing for flexible retrieval strategies depending on the task. Additionally, Exa offers a suite of APIs that enable developers to tailor the search experience to specific use cases.

From a technical and operational standpoint, Exa is built for scale and privacy. It supports real-time indexing and content scraping, offers custom filters by domain, date, and category, and complies with SOC2 standards. It also enforces zero data retention, making it a privacy-conscious choice for enterprise applications.

In the 2024 study *"Enhancing Patient Medication Safety at Home: A Patient-Facing Technology Architecture Integrating REDCap, Visualization Dashboards, and an AI-Driven Chatbot"*[29], Exa was integrated into a biomedical research assistant powered by an LLM. The agent was designed to answer complex medical queries by retrieving up-to-date information from the web. Exa played a central role in this process by sourcing content such as PubMed abstracts, clinical trial summaries, and other domain-specific documents. The integration followed a tool-augmented agent loop. The LLM would determine when a web search was necessary, issue a query via Exa, and then use the retrieved content to generate a grounded, contextually accurate response. This setup significantly improved the factual reliability of the agent's answers, especially in a field where timeliness and precision are critical.

In *"Coding Agents with Multimodal Browsing are Generalist Problem Solvers"*[30], Exa was integrated into the OpenHands-Versa agent, a general-purpose LLM framework capable of solving diverse tasks such as coding, web navigation, and information retrieval. Exa served as one of the agent's core tools for real-time web search, enabling it to retrieve relevant content during task execution. Its semantic filtering and structured output allowed the agent to ground its reasoning in up-to-date information, contributing to its ability to outperform more specialized systems. This same study also evaluated Brave and Tavily as alternative search tools within the agent's toolset, highlighting their complementary strengths in retrieval quality and integration flexibility.

In *"Evaluation Report on MCP Servers"*[31], Exa was benchmarked as one of several web search tools integrated into the Model Context Protocol (MCP) framework. Within this setup, Exa was exposed as a callable endpoint that LLMs could use to retrieve external knowledge when their internal context was insufficient. The study assessed tools based on accuracy, latency, and token efficiency, and Exa demonstrated strong performance in real-time retrieval and semantic relevance. The same benchmark also included Brave and Tavily, highlighting their respective strengths in privacy-focused search and long-form content extraction. Together, these tools represent a new generation of AI-native search infrastructure tailored for tool-using LLMs.

3.2.3.2.4.2 LangSearch

LangSearch is a lightweight semantic web search API designed to integrate seamlessly into AI applications, particularly those using Retrieval-Augmented Generation (RAG). It emphasizes contextual relevance and simplicity using embedding-based semantic search to retrieve content that aligns closely with the intent of natural language queries. While its public documentation is limited, LangSearch is built for easy integration into LLM pipelines and may support filtering by domain or content type, making it a practical choice for developers seeking a minimal yet effective search layer.

In *"Pangu DeepDiver: Adaptive Search Intensity Scaling via Open-Web Reinforcement Learning*" [32], LangSearch was used as the core search backend for the DeepDiver agent. This agent dynamically adjusted how often and how deeply it searched the web based on the complexity of the user's question. LangSearch enabled the retrieval of semantically relevant passages that were then used for reasoning and answer generation. The study introduced the WebPuzzle benchmark and showed that DeepDiver, powered by LangSearch, outperformed larger models by being more strategic and efficient in its use of web search.

3.2.3.2.4.3 Zenserp

Zenserp is a web search API that provides structured access to Google Search results, emulating the behavior of a traditional search engine while returning data in a machine-friendly JSON format. Although it doesn't offer native AI functionalities like semantic search or summarization, its strength lies in its ability to deliver ranked URLs, metadata, and SERP snapshots that can be easily parsed and integrated into downstream pipelines. It supports location- and language-specific queries, making it suitable for multilingual and geographically targeted applications. Zenserp is often used in scenarios where structured access to search engine data is more valuable than raw HTML scraping.

In the paper *"Knowledge Enhanced Reflection Generation for Counseling Dialogues"* [33], Zenserp was used to power the web mining component of a system designed to generate empathetic and informative responses in therapeutic conversations. The system constructed search queries using predefined templates and sent them to Google via the Zenserp API. The top 100 matching websites were retrieved and parsed into sentences using SpaCy. These sentences were then filtered for medical and psychological concepts and used as external knowledge to support the LLM's response generation. The study found that integrating this retrieved knowledge, alongside commonsense reasoning from COMET, significantly improved both automatic evaluation metrics and human judgments of generated dialogue quality.

Zenserp was also featured in a study titled *"A Machine Learning Python-Based Search Engine Optimization Audit Software"* [34], where it served as the data collection layer for an automated

SEO auditing tool. The system used Zenserp to fetch SERPs for specific keywords, collecting data on rankings, metadata, and page structure. This information was then processed using Python-based machine learning models to evaluate the SEO performance of websites. The integration of Zenserp allowed the tool to operate in real time, enabling dynamic audits that could adapt to changes in search engine behavior and keyword trends.

In *"Do You See What I See? Images of the COVID-19 Pandemic Through the Lens of Google"* [35], Zenserp was employed to programmatically retrieve image search results for COVID-19-related queries across 78 countries and 10 languages. The researchers automated Google Image searches using Zenserp, collecting the top 100 image URLs for each query. These images were then analysed using computer vision tools to identify visual themes such as the presence of people, masks, or medical imagery. The study aimed to understand how different cultures visually represented the pandemic and how search engine results might shape public perception. Zenserp's consistent and reproducible access to image search results made this large-scale, multilingual study feasible.

3.2.3.2.4.4 Brave Search API

Brave Search API is a privacy-focused, independent web search solution built on Brave's proprietary index of over 30 billion web pages. Unlike APIs that rely on third-party engines, Brave offers a fully autonomous search infrastructure, making it a compelling choice for developers seeking transparent, unbiased, and privacy-respecting data sources. It supports semantic search, summarization, and structured result formatting, and is designed for seamless integration with LLMs and AI agents.

Brave is particularly well-suited for RAG pipelines, where grounding and hallucination reduction are critical. Its consistent data structure, high scalability (50+ queries/sec), and access to diverse content types (news, images, videos, and general web results) make it a versatile tool for real-time knowledge retrieval. Importantly, Brave does not track users or personalize results, ensuring reproducibility and neutrality in search outcomes.

In *"Coding Agents with Multimodal Browsing are Generalist Problem Solvers"* [30], Brave Search API was integrated into the OpenHands-Versa framework, an LLM-based agent capable of solving a wide range of tasks, from software engineering to general knowledge queries. Brave served as one of the agent's core tools for real-time web access, alongside code execution and file manipulation. The agent used Brave to retrieve up-to-date information during task execution, enabling it to reason over current documentation, news, or technical content. The study demonstrated that this multimodal, tool-augmented approach allowed the agent to outperform more specialized systems by leveraging flexible, real-time search capabilities. Brave's structured output and privacy guarantees made it a reliable and scalable component of the agent's toolset.

In *"NileChat: Towards Linguistically Diverse and Culturally Aware LLMs"* [36], Brave was used to collect culturally relevant web content in underrepresented languages such as Arabic and Swahili. The goal was to build a pretraining corpus that better reflected the linguistic and cultural contexts of non-Western communities. Brave's ability to retrieve documents in multiple languages, without personalization or regional bias, was crucial for ensuring the inclusivity and representativeness of the dataset. The project highlighted Brave's role in supporting linguistic diversity and cultural grounding in LLM development, helping to close the gap in AI accessibility for global populations.

In the previously described *"Evaluation Report on MCP Servers"* [31], Brave was evaluated alongside Exa and Tavily as part of a comprehensive benchmark of web search tools integrated into the Model Context Protocol (MCP) framework. Brave stood out for its privacy-first architecture and scalability, offering consistent, structured results without personalization or tracking. Its integration into the MCPBench framework demonstrated that Brave could serve as a reliable, production-grade search backend for tool-using LLMs, particularly in scenarios where neutrality and reproducibility are essential.

3.2.3.2.4.5 Tavily

Tavily is an AI-powered web search API built specifically for integration with LLMs and Retrieval-Augmented Generation (RAG) systems. It offers a rich set of features tailored to the needs of intelligent agents and research workflows, including semantic search, summarization, and long-form content extraction (supporting documents over 3,000 characters). Tavily can scan and extract information from the entire web pages, enabling deep contextual understanding and precise information retrieval. It is designed to deliver accurate, real-time, and context-aware information, making it ideal for tasks that require deep understanding and factual grounding.

Beyond its core search capabilities, Tavily includes a research assistant mode, where users can define objectives and receive structured results via email. It also supports high-throughput querying (up to 1,000 calls per minute), making it scalable for enterprise-level applications. While not open source, it offers a generous free tier and is widely used in production-grade AI systems.

Once again, in *"Evaluation Report on MCP Servers"* [31], Tavily was one of the evaluated tools as part of the MCPBench framework. Tavily was recognized for its production-readiness and declarative interface, which simplified integration and improved retrieval precision. Its ability to deliver structured, relevant content made it a strong candidate for tool-using LLMs operating in dynamic environments.

A 2025 study named *"AI Property Valuation Tool for the UAE Real Estate Market"* [37] introduced a hybrid property valuation system that combined historical data with real-time web search. Tavily was used to retrieve live market data, which was fed into a valuation model powered by LLMs and vector databases. The system achieved an 80% reduction in valuation time and a 21% improvement in accuracy, demonstrating how Tavily's integration enabled faster and more reliable decision-making in real estate analytics.

In *"ASTRAL: Automated Safety Testing of Large Language Models"* [38], Tavily was used to generate dynamic, real-world prompts for testing LLM safety. By retrieving up-to-date web content, Tavily helped the system create edge-case scenarios that better reflected current risks. This approach nearly doubled the detection rate of unsafe behaviours compared to static datasets, highlighting Tavily's value in robust AI evaluation pipelines.

In another 2025 study called *"An AI Agent-Based System for Retrieving Compound Information in Traditional Chinese Medicine"* [39], Tavily was part of a multi-source retrieval system designed to gather compound information in Traditional Chinese Medicine. The system combined structured databases, domain-specific APIs, and open web search via Tavily. The retrieved content was processed by LLMs to enhance coverage and accuracy. The hybrid system achieved 96.67% accuracy, outperforming standard RAG baselines by over 25%, showcasing Tavily's effectiveness in specialized biomedical applications.

3.2.3.2.4.6 SerpAPI

SerpAPI is a widely used web search API that provides structured access to search engine results, including Google, Bing, Yahoo, and others. It's designed to return real-time, JSON-formatted SERP data, making it ideal for applications that require accurate and up-to-date search results. While not inherently AI-powered, SerpAPI is frequently used in conjunction with LLMs and machine learning pipelines for data extraction, analysis, and automation.

In "Cross Domain Answering FAQ Chatbot" [40] a chatbot capable of answering questions across multiple domains is proposed, to overcome the limitations of traditional FAQ systems that are restricted to predefined knowledge bases. The chatbot integrates SerpAPI to perform real-time web searches when it encounters queries outside its trained domain. The API retrieves relevant snippets from the internet, which the chatbot then uses to generate informed responses. This dynamic querying mechanism allows the system to extend its knowledge coverage and provide more accurate, up-to-date answers, significantly enhancing user experience.

"Agriconnect: A Data-Driven Platform for Optimizing Crop Production with Fine-Tuned Large Language Models" [41] presents a platform that combines fine-tuned LLMs with real-time data sources to support decision-making in agriculture. It aims to optimize crop production by integrating environmental, market, and agronomic data. In this work, SerpAPI is used to retrieve real-time agricultural information—such as weather updates, pest alerts, and commodity prices—from the web. This data is then processed by the LLM to generate context-aware recommendations for farmers. The integration of SerpAPI ensures that the system remains responsive to changing conditions, enabling more precise and timely agricultural decisions.

Advancements beyond the Related Work

Advancements beyond the Related Work
The approach to be followed will consist in the multiple usage of the identified tools for the implementation of an information flow that allows the analysis of information and its automatic representation in the data model used by FDAC. While, from a technological standpoint, no innovation will be present since the implementation resorts to an orchestration of existing tools, LLMs are used to extract information from existing assets by categorising and adding them to a catalogue. This allows to assess the accuracy/performance of the State-of-the-Art technologies on the execution of such tasks.

3.2.3.3 Implementation

In the implementation, it is important to model system architecture workflow and choose relevant tools to execute the required tasks by acknowledging the main objective of integrating external models from a marketplace into FDAC models. The important tasks include formatting the FDAC model template to understand what parameters are required to build a new model, decide tools that can extract information from external models with formats based on the marketplace, integrate the information extracted with LLM generation parameter's values, and the creation of the new FDAC model using FDAC API. To better understand these processes, the implementation of the workflow can be divided in different steps:

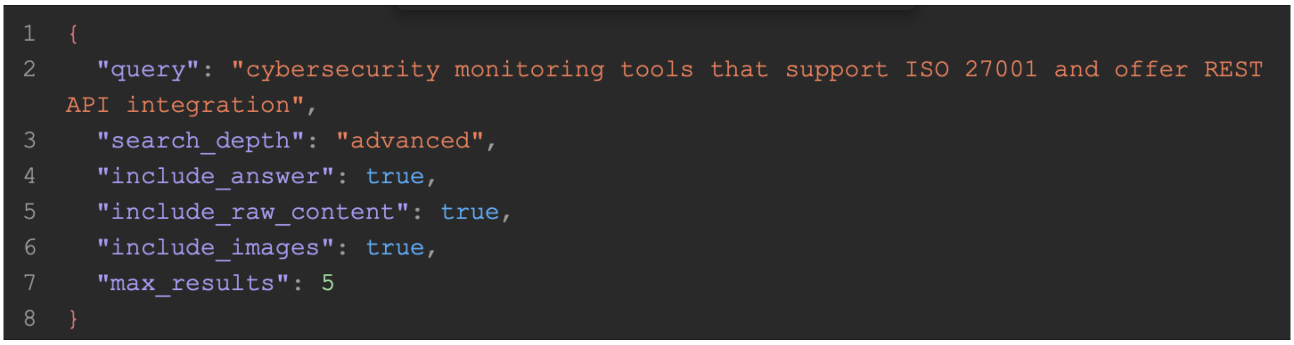
- Marketplace information extraction
- Generating values for the required FDAC's model parameters
- Integrating FDAC functionalities with LLM tool calling functionalities

3.2.3.3.1 Search

Web information extraction typically implies filtering what websites and content that are available online are relevant for our purpose, finalising with selection of relevant information for the goal proposed. In this scenario, the tool Tavily can filter through the marketplace website based resources and extract content according to a prompt given as feedback, using data extraction and data validation to evaluate and decide until it confidently understands that it has enough information in order to fill a required goal, which in this case is having enough information to generate values for each parameter of the FDAC model. To achieve this, Tavily operates as a prompt-driven web search and extraction engine, capable of scanning domain-specific marketplaces that host technical descriptions, service metadata, and tool specifications.

Tavily begins by interpreting the prompt as a high-level objective and performs semantic search across the available resources, identifying pages or entries that are contextually aligned with the task. This allows it to locate relevant content even when terminology varies across tools or vendors. Once relevant sources are identified, Tavily performs long-form content extraction, parsing web pages including structured sections such as tables, bullet points, and metadata blocks, to isolate key attributes. These may include tool names, supported standards or protocols, deployment models, integration capabilities, use cases or application domains and contact or vendor metadata. Tavily is designed to handle both structured and semi-structured content, enabling it to extract meaningful data even from heterogeneous formats. Tavily also applies contextual filtering based on the original prompt. For example, if the goal is to extract only tools that support a specific compliance network, Tavily will discard irrelevant entries and focus on those that match the criteria semantically.

To illustrate how Tavily can be used in this context, the following figure shows an example of a real API request that could be issued to retrieve relevant information from a domain-specific marketplace:



```
{
  "query": "cybersecurity monitoring tools that support ISO 27001 and offer REST API integration",
  "search_depth": "advanced",
  "include_answer": true,
  "include_raw_content": true,
  "include_images": true,
  "max_results": 5
}
```

Figure 3-32 – Tavily search API request used to extract validated content for FDAC model generation

Each field in this request plays a specific role in guiding Tavily's behaviour:

- **query**: A natural language description of the information required. In this case, it targets tools relevant to FDAC parameters such as compliance standards and integration capabilities.
- **search_depth**: Set to "advanced" to enable deeper crawling and more comprehensive extraction from each result, which is useful when scanning technical marketplaces.
- **include_answer**: Set to true to request a summarized answer based on the extracted content, helping the LLM quickly understand the context.
- **include_raw_content**: Set to true to retrieve the full raw content of the matched pages, essential for structured extraction of specific fields.

- **include_images**: Set to true to return relevant images (e.g., diagrams, screenshots, logos) associated with the content. This can be useful for visual validation or enriching the LLM's context.
- **max_results**: Limits the number of search results returned to 5, ensuring focused, high-quality results.

This type of request allows Tavily to return both a high-level summary and the raw content needed to extract structured data, which can then be passed to the LLM for FDAC model parameter generation. After extraction, Tavily applies a validation layer that includes redundancy filtering, consistency checks across multiple entries, and schema alignment. This ensures that the extracted data is accurate, non-duplicated, and compatible with the FDAC model template. Tavily continues this process iteratively, in case the initial extraction is incomplete or ambiguous, re-querying or refining its search and extraction until it reaches a confidence threshold that the gathered information is sufficient to support the generation of parameter values. The final output is structured in a format suitable for LLM ingestion - JSON format - allowing the LLM to reason over the data and generate the required FDAC model parameters. This integration between Tavily and the LLM ensures that the model creation process is grounded and validated, real-time information extracted directly from the marketplace.

3.2.3.3.2 Local LLM deployment

For the local deployment of LLM models, the Ollama framework is used to manage and configure models in a local environment. It allows users to select which models to run and create Modelfiles to change their characteristics, balancing performance and computational efficiency. Ollama has the possibility to run in the local machine, or the approach chosen for this deployment, which is using an official Ollama container that runs in a Kubernetes cluster. This container, called pod in Kubernetes terminologies, already has built-in the capability to pull models and run them inside the pod. However, to boost significantly the performance of models is important to connect this pod with GPU resources. This requires configuring the Linux-based pod to connect to system's graphic cards using official GPU drivers. After the setup of Ollama and the use of GPU performance enabled, this pod needs to expose Ollama functionalities to provide those functionalities to be used in external devices, including the management of LLMs but also configuration and usage. At the end, the Ollama exposes the models it is running locally and the tools to use and receive their answers as an API based in HTTP to be used in real-time.

For the generation of values to fill FDAC's model parameters, an open-source LLM, such as Llama3 model, is used and run locally. By using a template prompt where the contextualization is inserted along with the task to fill each parameters' values, we can query this prompt to the model, and it will generate the desired values based on the contextualization. This process involves software to run the LLM locally and tools that can handle with LLM outputs and format it to be used in variables in run-time. To enhance the robustness of the output from the LLM, it is used the framework LangGraph to run an AI agent orchestration to assure the quality of the LLM output for each parameter and improve the results output.

3.2.3.3.3 AI-Agent Framework

The LangGraph models the flow of processes as a graph described in **Error! Reference source not found.**, where each node represents different tasks from the workflow and each edge represents connections between tasks with a description of the data that is transferred from node to node. The flow starts with the search on the marketplace according to the initial user prompt and get the information that Tavily finds relevant. A decision node is placed after to evaluate the quality and completeness of the information extracted, accepting the information as enough to fill the desired parameters from the FDAC model or if extra information from marketplace should be searched and what prompt should be used for that. When enough information is gathered, a mapping of FDAC models' parameters are used to generate values using the context obtained previously using the LLM. By using LangGraph environment, a node is created that can connect with the FDAC MCP server and attribute the values generated to a new model and finally integrating the new asset as a new model in FDAC with success.

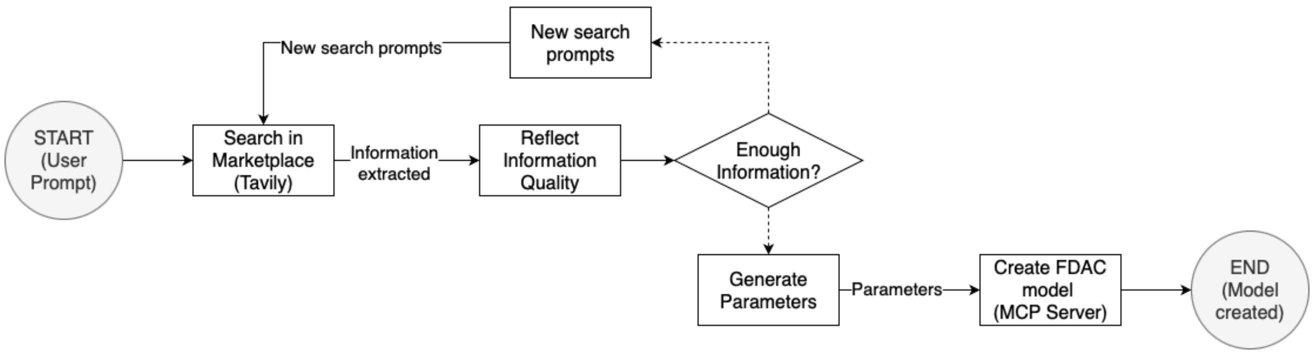
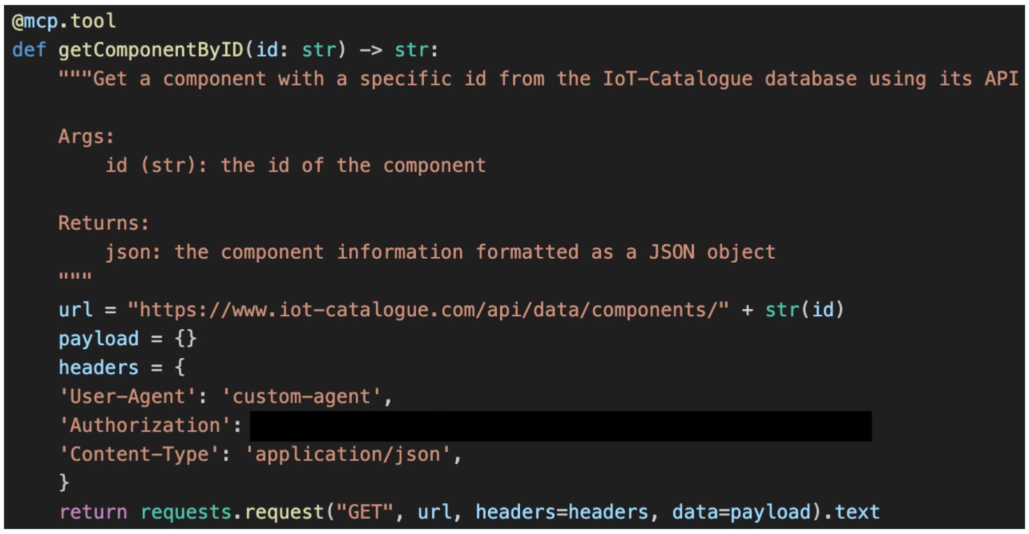


Figure 3-33 – Graph representation of the architecture flow

3.2.3.3.4 FDAC API MCP server

For the step that integrates LLM tool calling with the FDAC API and the parameters generated to create the model, it is crucial to adapt the FDAC functionalities with the standard layer MCP that relates FDAC functionalities with the tool's layout, enabling LLM to interpret and understand their purpose. By creating a MCP server for FDAC functionalities, LLM can decide which tools can be used and automatically select correct parameters and directly execute calls for those functionalities.



```
@mcp.tool
def getComponentByID(id: str) -> str:
    """Get a component with a specific id from the IoT-Catalogue database using its API

    Args:
        id (str): the id of the component

    Returns:
        json: the component information formatted as a JSON object
    """
    url = "https://www.iot-catalogue.com/api/data/components/" + str(id)
    payload = {}
    headers = {
    'User-Agent': 'custom-agent',
    'Authorization': [redacted]
    'Content-Type': 'application/json',
    }
    return requests.request("GET", url, headers=headers, data=payload).text
```

Figure 3-34 – Example tool from IoT-Catalogue MCP Server

The implementation of FDAC MCP server is similar to any other MCP server, where different functionalities from the API should be adapted into a MCP tool, such as the example in Figure 3-34. As the example of IoT-Catalogue, tools are implemented as a normal Python function that interacts with IoT-Catalogue. The main characteristic of MCP tools is the description given, since this should have a complete and extensive explanation of the tool and the purpose of each parameter in order for the LLM interpret and improve its decision-making. To implement a MCP Server, several features can be implemented:

- **Tools:** Functions that will be used by the LLM. For this case, these functions represent different functionalities from the FDAC API. It is important to give a description for each respective tool but also other descriptions for each parameter of the function in order for the LLM choose not only the correct tool but to know how the parameters should be filled.
- **Resources:** Input and output data format of each tool, to document and represent how the data should be structured to work properly with the tool. It is not developed for this case, but can be a possible future improvement.
- **Prompts:** Default pre-defined prompts that can be reutilized for cases where a prompt template is used several times, grouping that template with the respective MCP server that is related to. In this case, MCP prompts will be used in future implementation to aid LLM decide the use of tools for specific scenarios, as the example of always requesting for an information in the FDAC API that is always needed like the FDAC asset's template.

After calling the respective API, the result text can be directly sent back from the MCP server or additional steps can be made to handle the data and properly format the results. The implementation of an MCP server implies the creation of MCP client that requests and listens information from the server. The MCP client searches for the server and its available tools in the network and passes to the LLM that will make the decision of how to utilize the correct tools. The same principle of handle data transfer from MCP server to client can be done in the client side to filter the relevant information prior to input into the LLM.

3.2.3.4 Beyond FAME

The approach of using LLMs to interpret Assets descriptions and represent the information modelled in data model used by FDAC is limited by the capabilities of the LLM models used. These limitations primarily affect the accuracy and completeness of information interpretation and its subsequent mapping to the data model. This includes potential misalignment in field-level representation, incorrect attribution of data to schema elements, and the omission of relevant information during the transformation process.

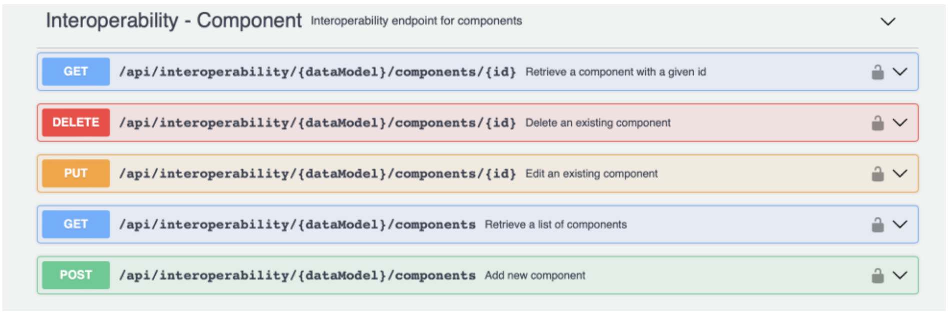
Different LLM models may present different results regarding how information is interpreted and mapped to the data model used by FDAC. Therefore, in theory, an approach that supports the execution of multiple LLMs in parallel, executing the same task, and capable of evaluation and combining the results of the multiple executions, would provide better results. Is in this task that another Horizon Europe project – HE MOSAICO (grant agreement ID: 101189664) – can be of help.

The MOSAICO project [42] is dedicated to transforming the landscape of software development by establishing a collaborative ecosystem where artificial intelligence agents and human developers work in unison. The initiative focuses on the development of a sophisticated AI platform that facilitates seamless interaction and coordination between these agents and human contributors. By designing intelligent AI agents tailored to specific stages of the software development lifecycle—such as requirements analysis, code generation, and testing—the project aims to significantly enhance productivity and software quality. Furthermore, MOSAICO places a strong emphasis on ethical and responsible AI practices, ensuring that all developments are guided by principles of transparency, fairness, and human oversight.

MOSAICO could be used automatically to replicate the information interpretation tasks, asking AI Agents connected to different LLMs to execute it, and applying a consensus algorithm to all the collected responses, to decide the best solution. UNP, as part Use Case partner in MOSAICO will test such approach.

3.2.4 Interfaces

The Semantic Proxy exposes its functionality as a REST API interface to provide access to the component's endpoints using data models other than the FDAC internal data model. The OpenAPI description is shown Figure 3-35. This description is accessible here[5].



Interoperability - Component Interoperability endpoint for components

GET `/api/interoperability/{dataModel}/components/{id}` Retrieve a component with a given id

DELETE `/api/interoperability/{dataModel}/components/{id}` Delete an existing component

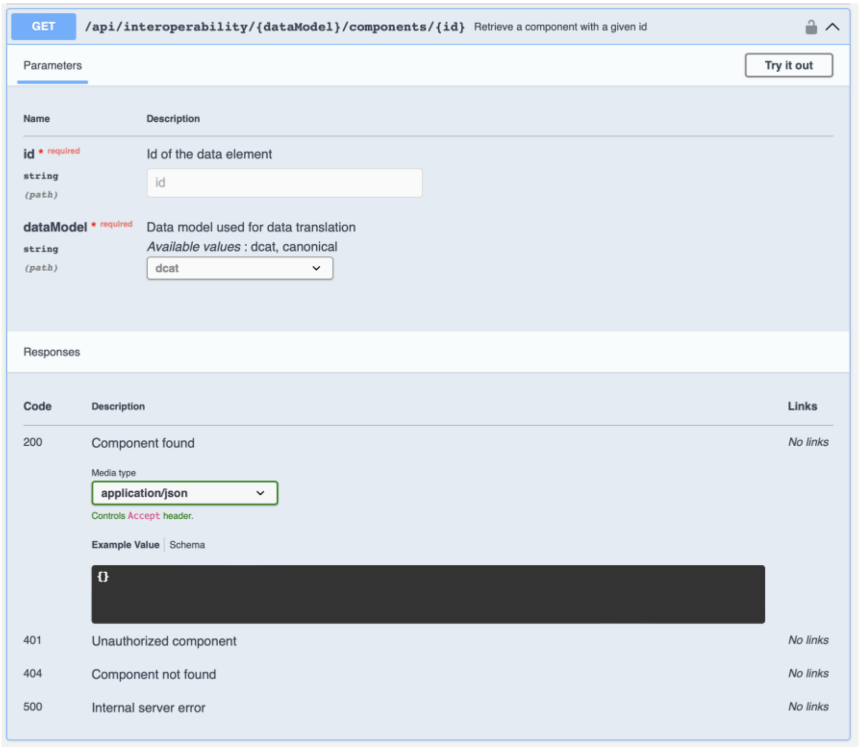
PUT `/api/interoperability/{dataModel}/components/{id}` Edit an existing component

GET `/api/interoperability/{dataModel}/components` Retrieve a list of components

POST `/api/interoperability/{dataModel}/components` Add new component

Figure 3-35 – Endpoints exposed by the Semantic Proxy

Each endpoint requires an extra parameter “dataModel” defined in the Path of the endpoint that corresponds to the data model that the user of the API wants to use. The usage of this parameter can be seen in Figure 3-36. Currently, only DCAT is supported.



GET `/api/interoperability/{dataModel}/components/{id}` Retrieve a component with a given id

Parameters

Try it out

Name	Description
id * required string *(path)*	Id of the data element id
dataModel * required string *(path)*	Data model used for data translation *Available values* : dcat, canonical dcat

Responses

Code	Description	Links
200	Component found Media type application/json Controls Accept header. Example Value | Schema `{}`	*No links*
401	Unauthorized component	*No links*
404	Component not found	*No links*
500	Internal server error	*No links*

Figure 3-36 – Details about the invocation of a endpoint provided by the Semantic Proxy

[5] https://fame-fdac.iot-catalogue.com/swagger/653a8be4b93bd08e33dd5e8e

4 Components Demonstration

4.1 Deployment Available

A FDAC deployment is available at https://fame-fdac.iot-catalogue.com which provides a web interface allowing to visualise public or private (using a login) information. As the Semantic Interoperability Middleware is integrated in FDAC, there is no dedicated demonstration of this component, being it involved when invoked any interoperability-related endpoint.

It is possible to access several data elements including assets, while clicking on an asset will display more detailed information.

The FDAC instance also provides a REST API allowing retrieval or adding new information to FDAC. The access must be authenticated using an access token that must be provided by and administrator of FDAC. The API is described using an OpenAPI 3.0 specification through the following swagger: https://fame-fdac.iot-catalogue.com/swagger/653a8be4b93bd08e33dd5e8e

4.2 Create Asset

Two POST REST endpoints are provided when adding a new asset. The selection of the endpoint depends on the data model that the user wants to use to describe the asset: FDAC data model or DCAT.

4.2.1 Component Endpoint

This endpoint is used when adding a new component using the internal FDAC data model. The following example illustrates how to add a new asset with a sample name and description.

If the asset is added successfully, then the ID of the asset will be returned, otherwise an error code with a message will be returned.

```
curl -X 'POST' \
  'https://fame-fdac.iot-catalogue.com/api/data/components' \
  -H 'accept: */*' \
-H 'Authorization: Bearer <Access Token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Sample asset using component api",
  "description": "Sample description"
}'
```

4.2.2 Interoperability Component Endpoint

This endpoint allows to add a new asset while defining a data model, allowing to use a JSON of an asset described with a specific standard, the following example shows how to add a new asset described with the DCAT model.

```
curl -X 'POST' \
  'https://fame-fdac.iot-catalogue.com/api/interoperability/dcat/components' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <Access Token>' \
  -H 'Content-Type: application/json' \
  -d '{
"dcterms_title":"Sample asset using interoperability API with DCAT data model",
"dcterms_description_short":"Sample description"
}'
```

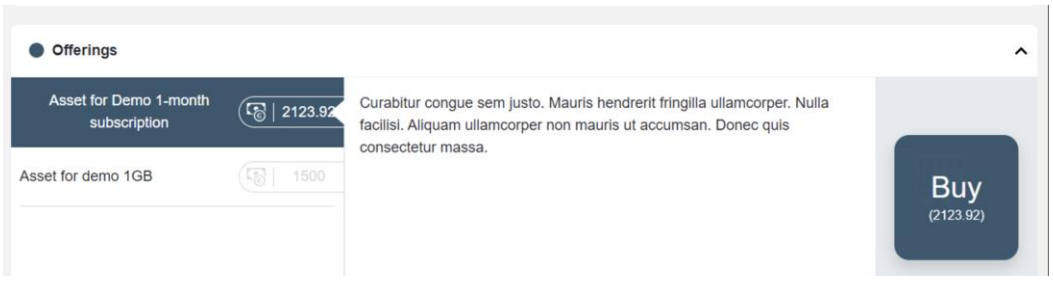
In this example "dcat" is being defined as a data model meaning that FDAC instance will be responsible to convert the asset from “dcat” structure to the internal FDAC data model by resorting to the Semantic Interoperability Middleware.

4.3 Plugin Integration

Information from external systems can be associated to an asset through the plugin API provided to authenticated users, but first the plugin must be registered including all the visual elements needed to present the information provided. The following steps provides a brief example on how to register and use a plugin:

- Register the plugin
    - Include React.JS visual elements including, bar (can include overlay), and plugin page that can be composed by other bars
    - Implement the method “getData” used to obtain the plugin data specific for the asset
- Add new information using the plugin API by defining the following info
    - **Access token:** Token required for authentication and the plugin identification
    - **Content:** Define the content that will be used to display information to the user
    - **Element id:** The asset that is going to display this content through the plugin
- Visualise the information by open the asset through the FDAC instance

The following bar exemplifies a Bar provided by a plugin of the Trading a Monetization Component that is responsible by showing a list of offerings available for a given asset, allowing to start the purchase by using “Buy” button.



Curabitur congue sem justo. Mauris hendrerit fringilla ullamcorper. Nulla facilisi. Aliquam ullamcorper non mauris ut accumsan. Donec quis consectetur massa.

Figure 4-1 – Offerings plugin used on the Trading and Monetization component

4.4 Search (via REST API)

This endpoint allows to search for assets available on the FDAC instance. The following example shows a search being performed to find the assets added by the previous examples by using the search term “sample”.

```
curl -X 'POST' \
  'https://fame-fdac.iot-catalogue.com/api/data/search' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer <Access Token>' \
  -H 'Content-Type: application/json' \
  -d '{
  "values": [
    {
      "term": "Sample"
    }
  ],
  "expand": true,
  "outputFilter": [
    "components"
  ]
}'
```

In the response, an object will be returned with the number of results and an array with the results, where each result includes a “weight” property with a score indicating the relevance of the search term, where the higher the “weight”, the more relevant the result is.

```
{
  "numberOfResults": 2,
  "results": [
    {
      "id": "TuvYZug3vQgBvLRnH",
      "type": "components",
      "weight": 11.055243,
      "dataElement": {
        "name": "Sample asset using component api",
        "longDescription": "Sample description",
        "tags": [],
        "developers": [],
        "manufacturers": [],
        "linkedComponents": [],
        "standards": []
      }
    },
    {
      "id": "4aMZdcE8ELRneNpmw",
      "type": "components",
      "weight": 8.145176,
      "dataElement": {
        "name": "Sample asset using interoperability API with DCAT data model",
        "tags": [],
        "developers": [],
        "manufacturers": [],
        "linkedComponents": [],
        "standards": []
      }
    }
  ]
}
```

4.5 Asset Certification Endpoints

The following curl request shows an example on how to certificate a component, get the certifications of a component and delete a certification, for these requests the token of the certificate authority must be used.

This request shows how to certificate an asset using the authentication token of the certificate authority, a response will be shown with the certification id.

```
curl -X 'POST' \
  'https://fame-fdac.iot-catalogue.com/api/data/components/<ASSET_ID>/certificate' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <CERTIFICATE_AUTHORITY_TOKEN>'\
  -d ''

Response: "Rp5tn8ykpdRiHspW9"
```

Is possible to obtain all the existing certifications associated with an asset, this would be a necessary step before deleting an asset certification.

```
getCertification
curl -X 'GET' \
  'https://fame-fdac.iot-catalogue.com/api/data/components/<ASSET_ID>/certificate' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <CERTIFICATE_AUTHORITY_TOKEN>'

Response: [
		{
			"userName": "FAME Workspace",
			"date": "2025-07-23T14:44:19.674Z",
			"relationId": "Rp5tn8ykpdRiHspW9"
		}
```

The relationId field of the response above contains is certification id value that must be used when deleting an asset certification.

```
curl -X 'DELETE' \
  'https://fame-fdac.iot-catalogue.com/api/data/components/<ASSET_ID>/certificate/Rp5tn8ykpdRiHspW9' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <CERTIFICATE_AUTHORITY_TOKEN>'

"Deleted certification with certification id: 'Rp5tn8ykpdRiHspW9'"
```

4.6 Requesting Asset Iframes

This example shows how the dashboard uses FDAC Public API to generate the Iframe URL containing the asset list, a header with the JWT token must be included if the user is authenticated in the FAME Marketplace.

```
curl -X 'GET' \
  'https://fdac-open-api.fame-horizon.eu/generateAssetListIframeURL?origin=https%3A%2F%2Fmarketplace.fame-horizon.eu' \
  -H 'accept: */*' \
  -H 'Authorization: <JWT Token>'

Response :https://www.iot-
catalogue.com/embedded/componentListNoBar?4cdec1=%7B%22token%22%3A%220rW7Ndp3ouHROi0GoetxkNDz8B26ysuQGpPt
Sb_pUod%22%2C%22useResizeSensor%22%3Atrue%2C%22origin%22%3A%22https%3A%2F%2Fmarketplace.fame-
horizon.eu%22%7D
```

When the user selects an asset from the list an iframe is generated containing only authorisation to visualise the asset page requested, the ensures that the user of the dashboard is blocked from accessing unauthorized asset details pages.

```
curl -X 'GET' \
  'https://fdac-open-api.fame-horizon.eu/generateAssetDetailsIframeURL/<ASSET_ID> \
  -H 'accept: */*' \
  -H 'Authorization: <JWT_TOKEN>

Response :https://www.iot-
catalogue.com/embedded/components/9xYwmMjy9sujrsMT9?4cdec1=%7B%22token%22%3A%22jsLur3dykRMPMZwUAj_RkipMV_
sPyXHFSZJT7C0FY9N%22%2C%22useResizeSensor%22%3Atrue%7D
```

4.7 Iframe Post message API example

These examples show the mechanism to exchange messages between the dashboard and the Iframe containing the asset page of FDAC, the communication is done by using the "window.postMessage" method on the Iframe element instance. This enables the dashboard to dynamically interact with a loaded Iframe, applying filter and sorting functions over content scoped in the iframe.

```
// It sorts the assets by its name descending

iframe.contentWindow.postMessage({action:”setSort”,value:{name:-1}},”*”)

// Search the assets using the term “asset”

iframe.contentWindow.postMessage({action:"search",value:"asset"},"*")

// Filter the elements to only show the Models on the asset list

iframe.contentWindow.postMessage({action:"tagFilter",value:[{name:"Model",type:"componentType"}]},"*")
```

5 Conclusions

This deliverable reported the development of the Federated Data Assets Catalogue and the Semantic Interoperability Middleware, identifying the background technologies used in the development of those components, the functionalities provided by the background technologies and the functionalities added in the context of FAME.

The UNPARALLEL Web Catalogue Framework serves as the backbone of the *Federated Data Assets Catalogue*. This framework provides FDAC with capabilities to visualize asset information, edit assets, manage asset permissions, and provide a data model to describe the assets. Those functionalities have been extended by exposing asset edition capabilities on a REST API, enhanced with a search endpoint supporting the execution of search queries, and the implementation of a plugin system to extend assets' information and visualization information with resources provided by external systems.

The Semantic Interoperability Middleware is based on tools and ontologies from INFINITECH project, that combined with approaches and technologies previous developed by partners of Task T3.4, support the design and implementation of this middleware. Using FDAC as a repository of semantic ontologies, the middleware then uses those semantic ontologies to create mappings concepts between ontologies. These mappings can be automatically executed, supporting the provisioning of a dynamic API that supports the execution of actions over assets using different data formats than those used by FDAC.

In the second version of this document component descriptions and diagrams were updated to reflect the changes on the final version of the components. In the context of FDAC some changes were made to the flow of getting a list of assets and details of a specific asset, integrating the APM on each flow to enforce related policies before providing any information to users. During this integration a new sub-component – FDAC Public API - was also developed expose endpoints to FAME users and implement specific integration flows. This component also allows to separate FDAC internal authentication mechanism from FAME user authentication system. FDAC also implemented functionalities to allow to represent that asset descriptions are certified by FAME, assuring that asset registration and descriptions are controlled by FAME Provenance system.

Regarding to the Semantic Middleware, a new approach for supporting interoperability with assets descripted on data models different from the one used by FDAC was presented. This approach relies on modern AI and LLM technologies and frameworks, delegation the task of extracting information described on data models or human readable format to LLM models. This allows users to provide descriptions on documents or webpages without having to represent the information it in specific data format/ontology before registering it. Moreover, with the search capabilities of AI tools, entire websites can be automatically scanned to collect information, eliminating the need of adapting to marketplace specific-APIs to retrieve the asset information.

6 References

[1] "FAME-D2.1 - “Requirements, Specifications and Co-Creation”," 2023.

[2] "FAME-D2.2 - “Technical Specifications and Platform Architecture”," 2024.

[3] "IoT-Catalogue-Data-API," Unparallel Innovation, 25 09 2023. [Online]. Available: https://github.com/unparallel-innovation/IoT-Catalogue-Data-API. [Accessed 2024 05 3].

[4] "DDP Specification," Meteor, 13 05 2021. [Online]. Available: https://github.com/meteor/meteor/blob/devel/packages/ddp/DDP.md. [Accessed 3 05 2024].

[5] "What is RESTful API," AWS, [Online]. Available: https://aws.amazon.com/what-is/restful-api/. [Accessed 3 5 2024].

[6] "OpenAPI," OpenAPI Initiative, [Online]. Available: https://www.openapis.org/. [Accessed 3 05 2024].

[7] Unparallel Innovation, "FDAC REST API," Unparallel Innovation, [Online]. Available: https://fame-fdac.iot-catalogue.com/swagger/653a8be4b93bd08e33dd5e8e. [Accessed 3 05 2024].

[8] W3C, "Data Catalog Vocabulary (DCAT) - Version 3," [Online]. Available: https://www.w3.org/TR/vocab-dcat-3/. [Accessed 26 January 2024].

[9] "INFINITECH-D4.2 – Semantic Models and Ontologies III.pdf," 2022.

[10] M. M. a. B. A. a. G. R. a. G. R. a. A. G. a. T. T. a. P. Malo, "Semantic Interoperability Toolkit for Data Marketplaces," *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT),* pp. 541-547, 2023.

[11] Z. X. a. X. G. R. Zhang, "ELECTRE II method based on the cosine similarity to evaluate the performance of financial logistics enterprises under double hierarchy hesitant fuzzy linguistic environment," *Fuzzy Optimization and Decision Making, vol. 22, no. 1,* pp. 23-49, March 2023.

[12] X. C. a. D. P. D. Liu, "Some cosine similarity measures and distance measures between q -rung orthopair fuzzy sets," *International Journal of Intelligent Systems, vol. 34, no. 7,* pp. 1572-1587, July 2019.

[13] J. L. a. Z. Z. S. Wu, "New distance measures of hesitant fuzzy linguistic term sets," *Phys Scr, vol. 96, no. 1,* p. 015002, January 2021.

[14] D. W. a. J. M. Mendel, "Similarity Measures for Closed General Type-2 Fuzzy Sets: Overview, Comparisons, and a Geometric Approach," *IEEE Transactions on Fuzzy Systems, vol. 27, no. 3,* pp. 515-526, March 2019.

[15] J. A. a. A. N. Al-kenani, "Vector Similarity Measures of Dual Hesitant Fuzzy Linguistic Term Sets and Their Applications," *Symmetry (Basel), vol. 15, no. 2,* February 2023.

[16] "BERT," [Online]. Available: https://huggingface.co/docs/transformers/en/model_doc/bert. [Accessed 2024].

[17] E. Kasneci, K. Sessler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel and J. Pfeffer, "ChatGPT for good? On opportunities and challenges of large language models for education," *Learning and*

Individual Differences, vol. 103, p. 102274, 2023.

[18] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan and D. S. W. Ting, "Large language models in medicine," *Nature Medicine,* vol. 29, no. 8, pp. 1930--1940, 2023.

[19] L. Yan, L. Sha, L. Zhao, Y. Li, R. Martinez-Maldonado, G. Chen, X. Li, Y. Jin and D. Gašević, "Practical and ethical challenges of large language models in education: A systematic scoping review," *British Journal of Educational Technology,* vol. 55, no. 1, pp. 90-112, 2024.

[20] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang and X. Xie, "A Survey on Evaluation of Large Language Models," *ACM Trans. Intell. Syst. Technol.,* vol. 15, no. 3, 2024.

[21] Hugging Face, Inc., "Hugging Face – The AI community building the future.," [Online]. Available: https://huggingface.co. [Accessed 24 June 2025].

[22] Ollama, "Ollama," [Online]. Available: https://ollama.com. [Accessed 30 June 2025].

[23] LangChain, Inc., "LangChain," [Online]. Available: https://python.langchain.com/docs/introduction/. [Accessed 7 July 2025].

[24] Anthropic PBC, "Introduction - Model Context Protocol," [Online]. Available: https://modelcontextprotocol.io/introduction. [Accessed 11 July 2025].

[25] CrewAI™, Inc., "CrewAI," [Online]. Available: https://www.crewai.com. [Accessed 10 7 2025].

[26] OpenAI, "OpenAI Agents SDK," [Online]. Available: https://openai.github.io/openai-agents-python/. [Accessed 10 7 2025].

[27] LangChain, Inc., "LangGraph," [Online]. Available: https://www.langchain.com/langgraph. [Accessed 19 6 2025].

[28] A. V. d. M. T. W. L. v. W. E. K. Aymeric Roucher, "HuggingFace smolagents," [Online]. Available: https://github.com/huggingface/smolagents. [Accessed 10 7 2025].

[29] Y. Yu, Y. Shi, E. Yang, K. Gahn, H. Mason, Y. Jiang, and Y. Gong, "Enhancing Patient Medication Safety at Home: A Patient-Facing Technology Architecture Integrating REDCap, Visualization Dashboards, And an AI-Driven Chatbot," *AMIA Annu. Symp. Proc.*, vol. 2024, pp. 1284–1293, May 2025. PMID: 40417505; PMCID: PMC1209933

[30] A. B. Soni, B. Li, X. Wang, V. Chen, and G. Neubig, "Coding Agents With Multimodal Browsing Are Generalist Problem Solvers", arXiv preprint arXiv:2506.03011, Jun. 2025. [Online]. Available: https://arxiv.org/abs/2506.03011

[31] Z. Luo, X. Shi, X. Lin, and J. Gao, "Evaluation Report on MCP Servers", arXiv:2504.11094, Apr. 2025. [Online]. Available: https://arxiv.org/abs/2504.11094

[32] W. Shi, H. Tan, C. Kuang, X. Li, X. Ren, C. Zhang, H. Chen, Y. Wang, L. Shang, F. Y and Y. Wang, "Pangu DeepDiver: Adaptive Search Intensity Scaling via Open-Web Reinforcement Learning", arXiv preprint arXiv:2505.24332, May 2025. [Online]. Available: https://arxiv.org/abs/2505.24332

[33] S. Shen, V. Pérez-Rosas, C. Welch, S. Poria, and R. Mihalcea, "Knowledge Enhanced Reflection Generation for Conseuling Dialogues", in Proc. 60th Annu. Meeting Assoc. Comput. Linguistics (ACL). Dublin, Ireland, May 2022, pp. 3096-3107. [Online]. Available: https://aclanthology.org/2022.acl-

long.221

[34] M. A. Albahli and M. A. Alsaeedi, ”A Machine Learning Python-Based Search Engine Optimization Audit Software”, Informatics, vol. 10, no. 3, p. 68, Aug. 2023. [Online]. Available: https://www.mdpi.com/2227-9709/10/3/68

[35] A. B. Makhortykh, S. Urman, and M. Ulloa, ”Do You See What I See? Images of the COVID-19 Pandemic Through the Lens of Google”, Inf. Process. Manag., vol. 58, no. 6, p. 102542, Nov. 2021. [Online]. Available: https://doi.org/10.1016/j.ipm.2021.102542

[36] A. El Mekki, H. Atou, O. Nacar, S. Shehata, and M. Abdul-Mageed, ”NileChat: Towards Linguistically Diverse and Culturally Aware LLMs for Local Communities”, arXiv preprint arXiv:2505.18383, May 2025. [Online]. Available: https://arxiv.org/abs/2505.18383

[37] L. A. A. Sultan and M. Kaddura, ”AI Property Valuation Tool for the UAE Real Estate Market”, Middle East Research Journal of Enginering and Technology, vol. 5, no. 2, pp. 33-39, 2025. [Online]. Available: https://kspublisher.com/media/articles/MERJET_52_33-39.pdf

[38] S. Zheng, A. Gupta, J. Li, R.Shokri, and M. Vechev, ”ASTRAL: Automated Safety Testing of Large Language Models”, arXiv preprint arXiv:2501.17132, Jan. 2025. [Online]. Available: https://arxiv.org/abs/2501.17132

[39] Y. Zhang, Y. Liu and Y. Wang, ”An AI Agent-Based System for Retrieving Compound Information in Traditional Chinese Medicine“, Information, vol. 16, no. 7, p. 543. [Online]. Available: https://www.mdpi.com/2078-2489/16/7/543

[40] G. K. Reddy, A. Pal, S. Krishna, J. Rishi and K. Saritha, ”Cross Domain Answering FAQ Chatbot”, 2022 International Conference on Innovations in Information Technology (IIT), Abu Dhabi, United Arab Emirates, 2022, pp. 218-223. [Online]. Available: https://ieeeplore.ieee.org/document/9752986

[41] R. Jeya, D. Singh, A. Jha and S. Mallick, ”Agriconnect: A Data-Driven Platform for Optimizing Crop Production with Fine-Tuned Large Language Models”, 2024 5th International Conference on Signal Processing and Machine Learning (SIGML), Tokyo, Japan, 2024, pp. 1-6. [Online]. Available: https://ieeexplore.ieee.org/document/10776860

[42] HE MOSAICO, "MOSAICO – Management, Orchestration and Supervision of AI-agent Communities," [Online]. Available: https://mosaico-project.eu. [Accessed 17 July 2025].