D3.2 - Federated Data Assets Catalogue I

Title	D3.2 - Federated Data Assets Catalogue I
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	2024-02-29 [Month 14]
Delivered Date	2024-05-28
Internal Reviewers	LXS UPRC
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	2024-02-02	UNP	Table of Contents
0.2	2024-03-02	UNP, DAEM, IQB, MOH, NOVA, NOVO	Integrated version with contributions
0.7	2024-04-02	UNP, DAEM, IQB, MOH, NOVA, NOVO	Version for peer review
0.8	2024-05-21	UNP, DAEM, IQB, MOH, NOVA, NOVO	Update after peer review
0.9	2024-05-22	UNP, DAEM, IQB, MOH, NOVA, NOVO	Version for QA
1.0	2024-05-28	FTS EUBA JOT TRB	Version for submission

Definitions

Acronyms	Definition
AAI	authentication authorization infrastructure
AI	Artificial Intelligence
API	Application Programming Interface
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
FDAC	Federated Data Assets Catalogue
FIBO	Financial Industry Business Ontology
FIGI	Financial Instrument Global Identifier
GIS	Geographical Information System
ICT	Information Communication Technologies
ID	Identity
IEEE	Institute (of) Electrical (and) Electronic Engineers
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
OWL	Web Ontology Language (W3C)
RDF	Resource Description Framework
REST	Representational State Transfer
SA	Supervisory Authority
SOTA	State of The Art
TR	Technical Requirement
UI	User Interface
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Executive Summary

This deliverable is the first 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 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.

This deliverable also identifies next steps regarding the development of both components. Concerning the development of the FDAC, one of the major developments would include the integration of the User Authentication System and the Asset Policy Manager used by FAME, allowing a seamless experience for authenticated users during the process of asset addition, discovery, and edition. From the development of the Semantic Interoperability Middleware point of view, the major proposed actions are related with the identification of semantic ontologies relevant to the pilots and their domain, as well as the improvement of the approach for the generation of semantic matchings among semantic ontologies aiming to achieve an automatised process.

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 first 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 first prototype 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 2.1 - "Requirements, Specifications and Co-Creation" [2] where requirements from Pilot partners were collected and provided valuable information to guide the development of functionalities. Deliverable 2.2 - "Technical Specifications and Platform Architecture" [3] 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.

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.

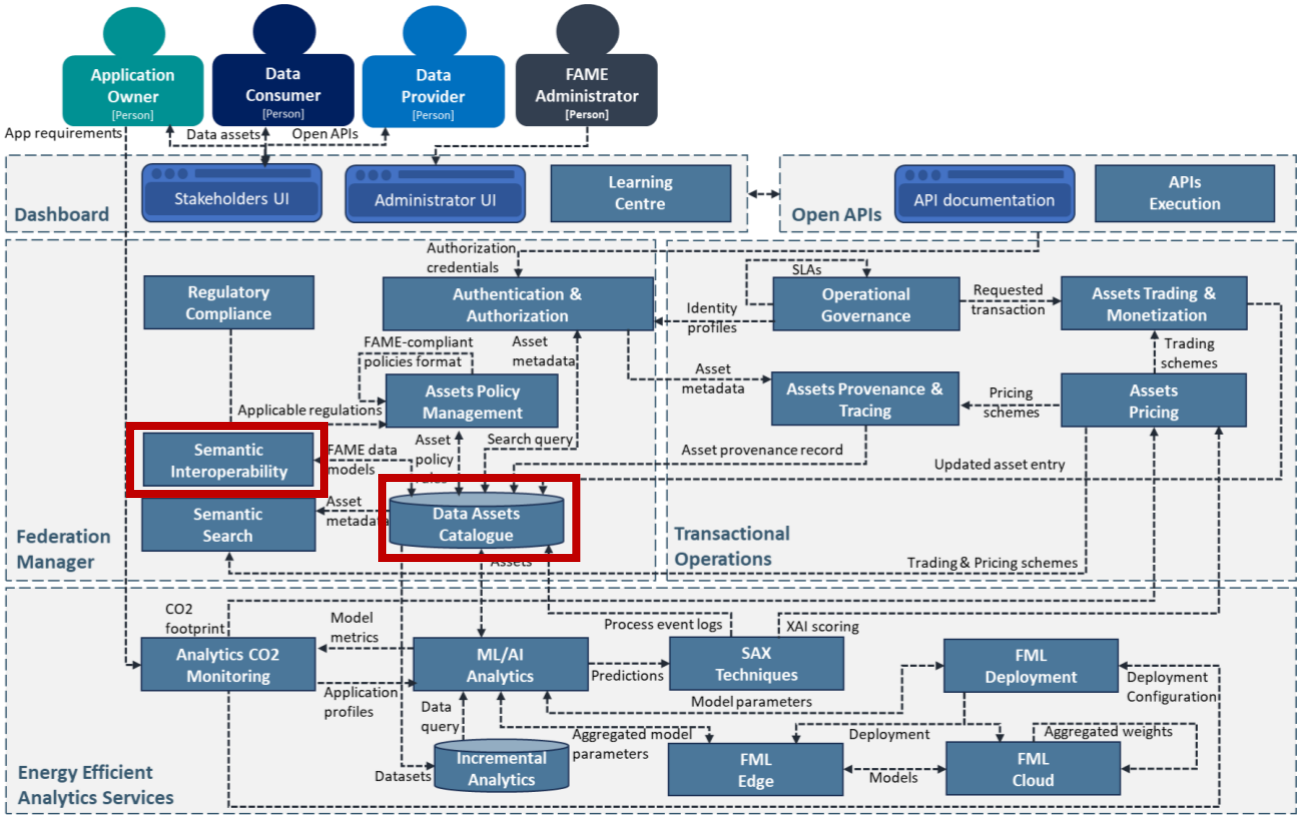

Figure 2-1 – FAME SA C4 container diagram

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

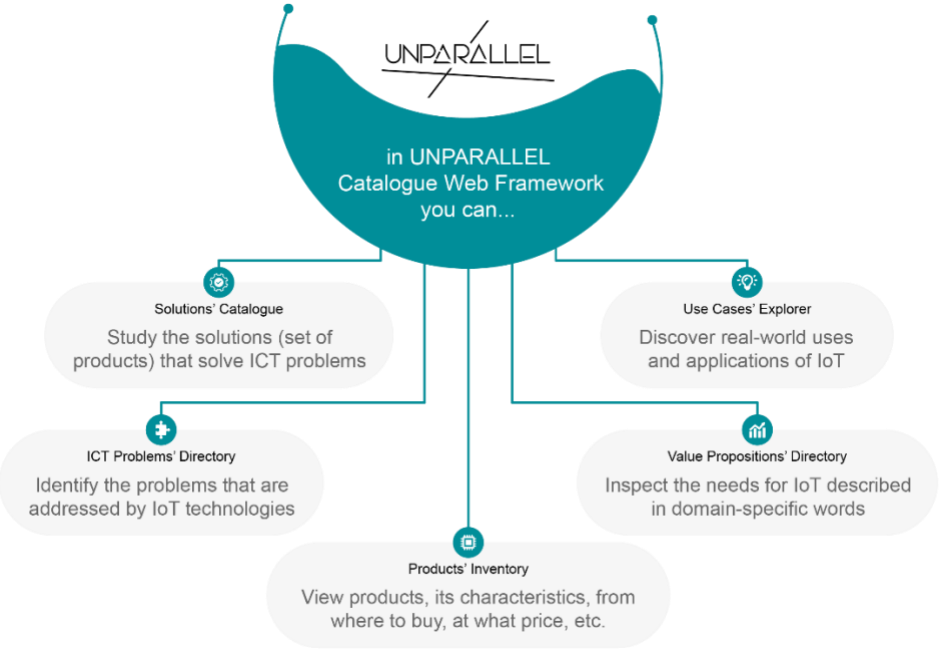

Figure 3-1 – UNPARALLEL Web Catalogue Framework

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 1 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 1- 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).

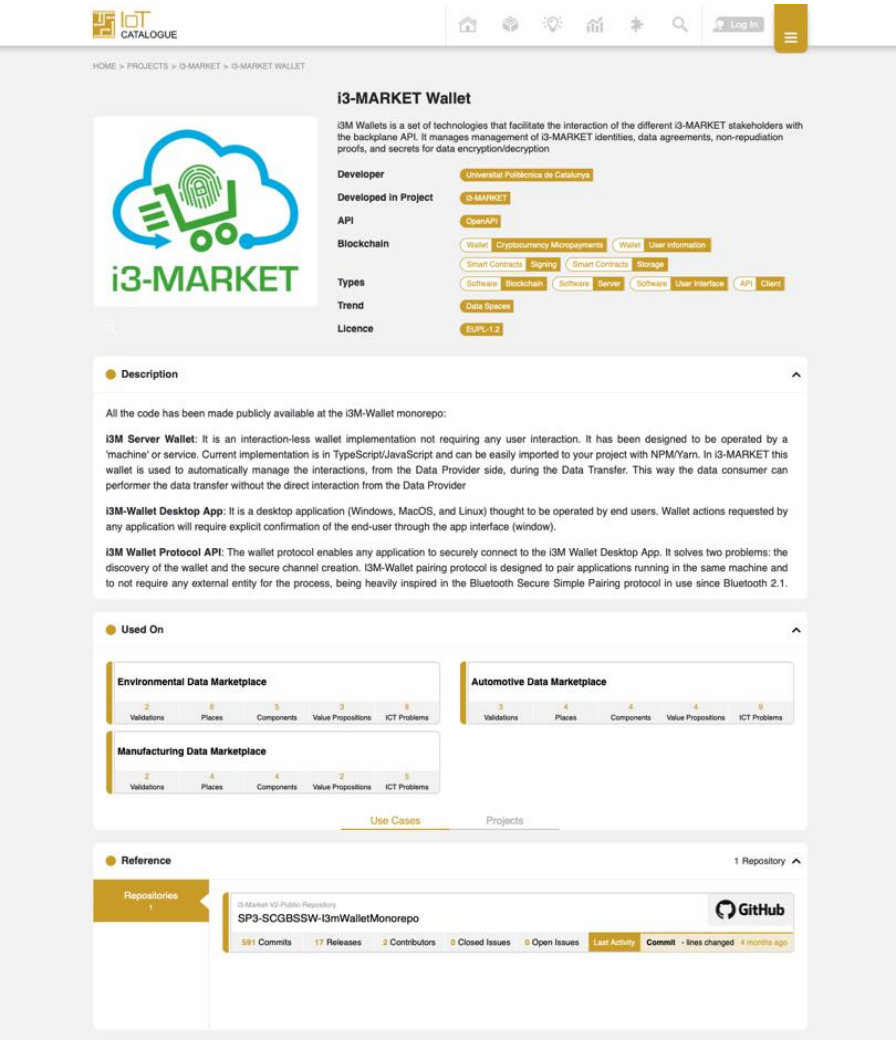

Figure 3-2 – Web interface for component visualization

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.

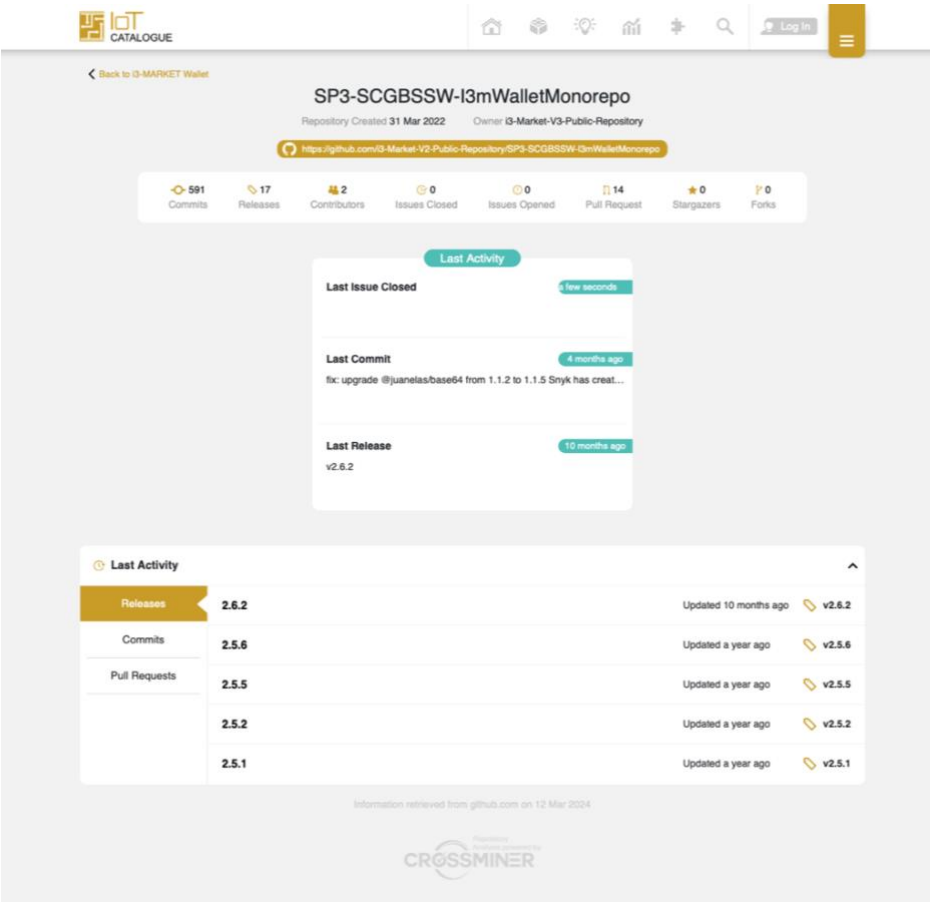

Figure 3-3 – Component repository details

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.

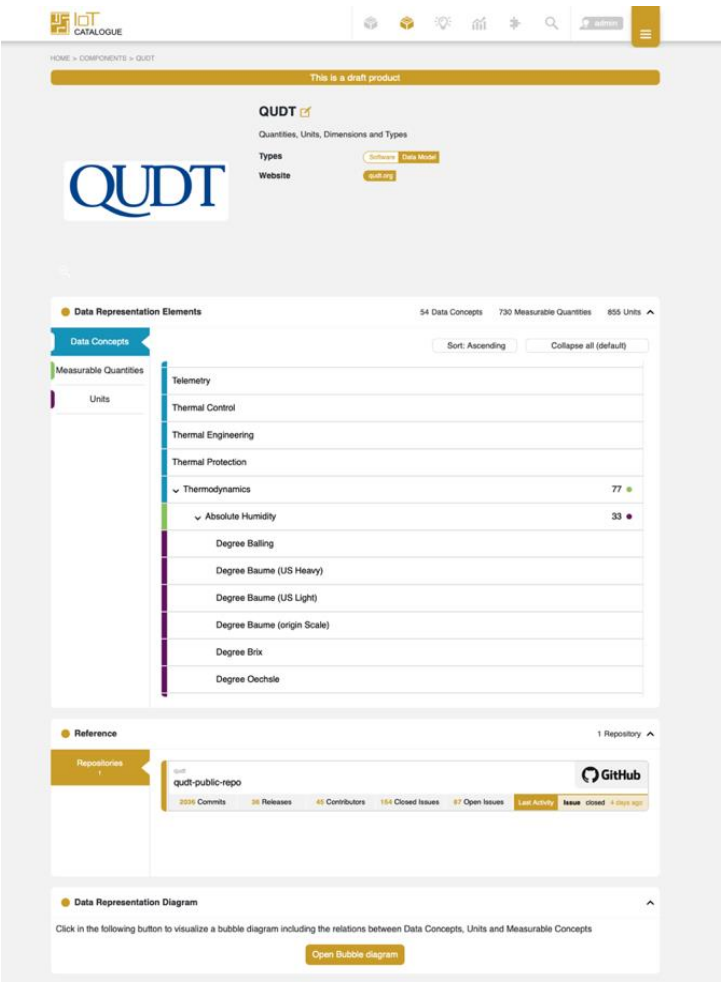

Figure 3-4 - Data Model page for QUDT (Quantities, Units, Dimensions and types)

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.

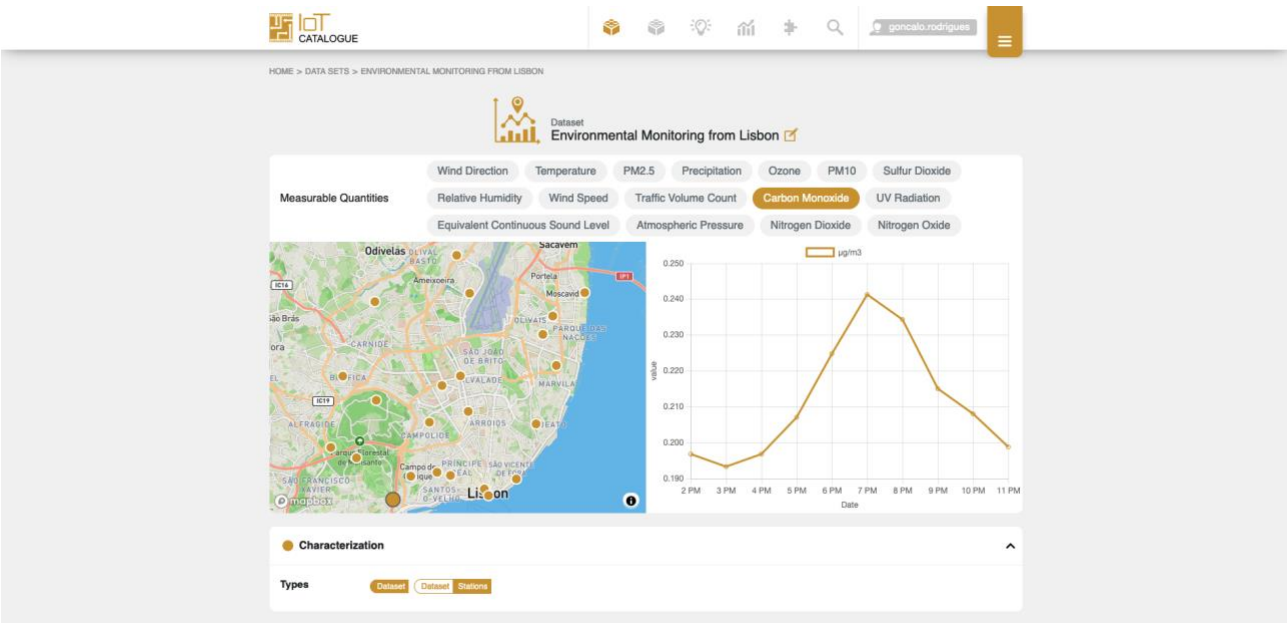

Figure 3-5 - Dataset page displaying information provided by Lisbon's environmental monitoring public API

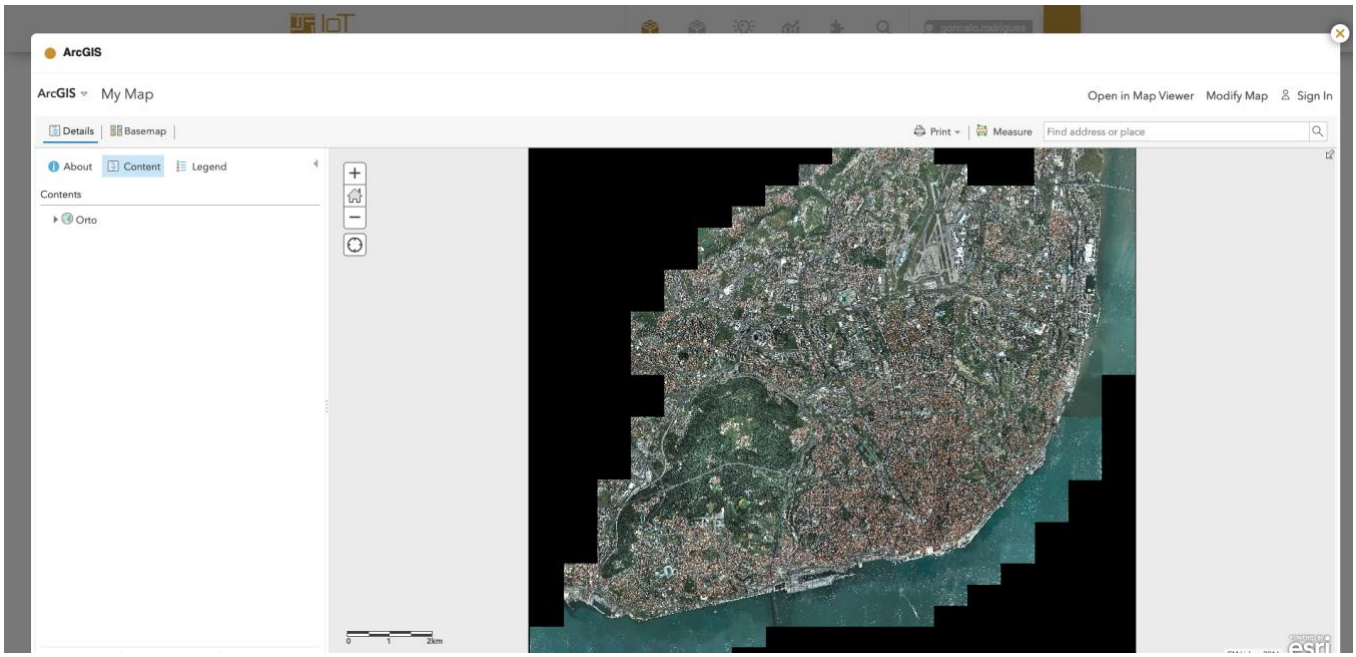

Figure 3-6 - GIS data visualization displaying an orthophoto map of Lisbon

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.

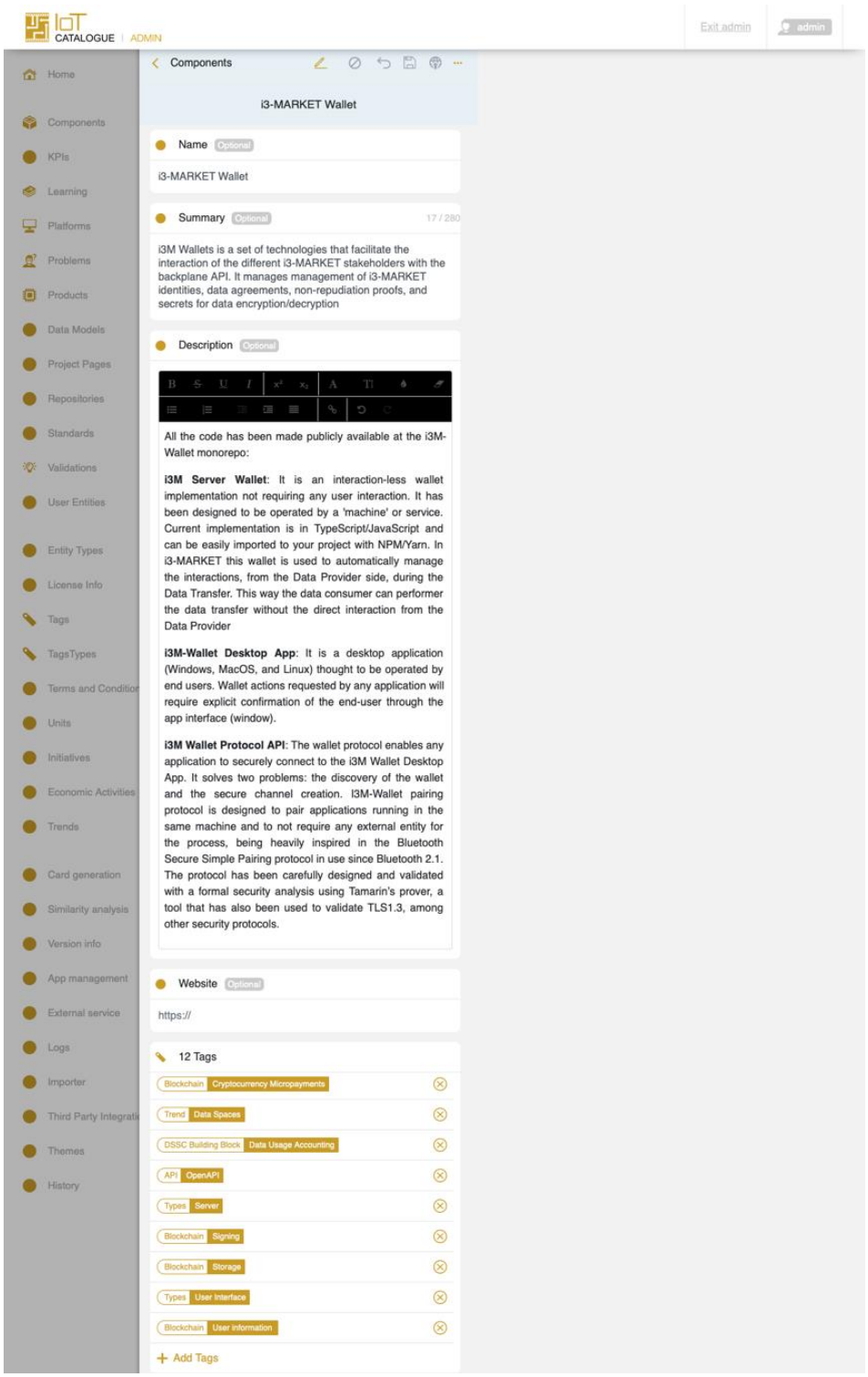

Figure 3-7 - Admin dashboard for asset creation

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. These 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. The remaining components of FDAC (External Source Manager, External Source Resolver, and Policy Extractor) will be described in the next version of this document.

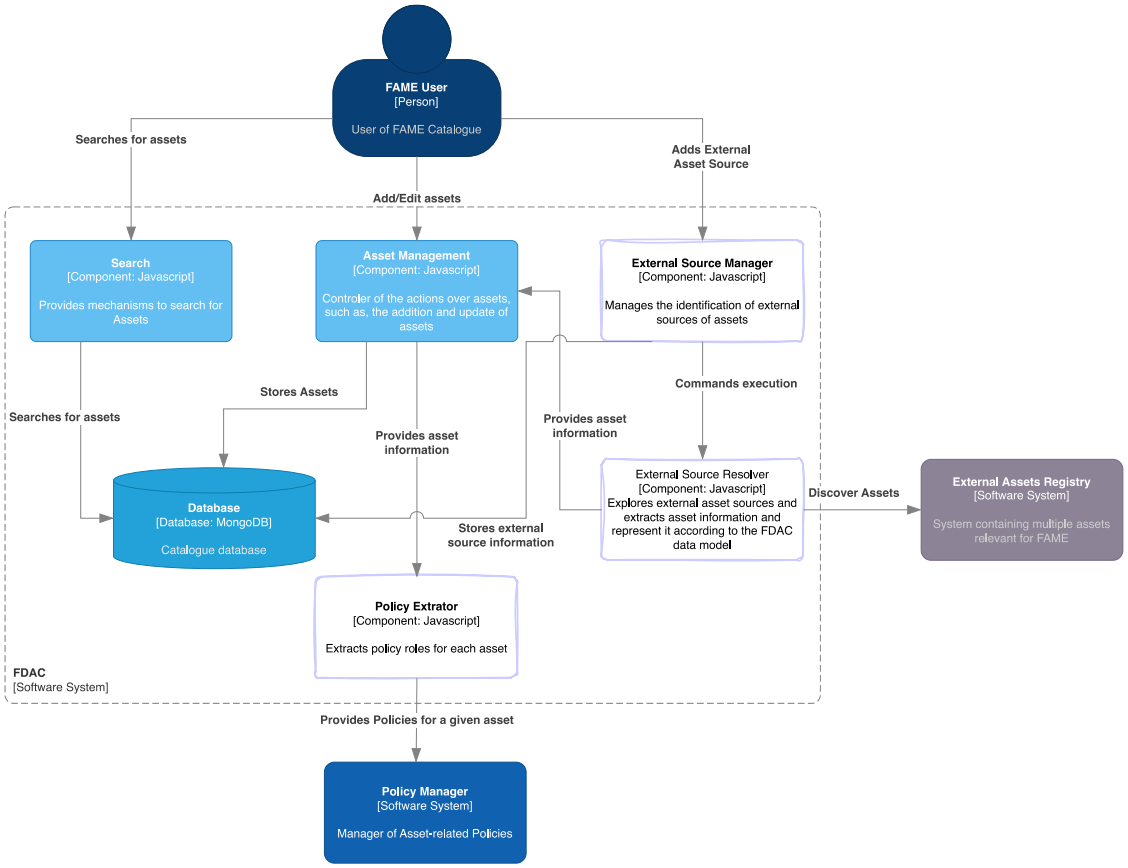

Figure 3-8 – FDAC C4 architecture

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 [4]. This initial implementation used DDP (Datagram Delivery Protocol) [5] while the one used by FDAC is REST (Representational State Transfer) [6] based. DDP is better suited for controlled environments, such as for internal development, while the REST standard with an OpenAPI [7] specification, significantly facilitates its integration with heterogeneous components using different technologies, hence the change.

The FDAC REST API [8], shown in Figure 3-12, 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 3-13) 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.

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.

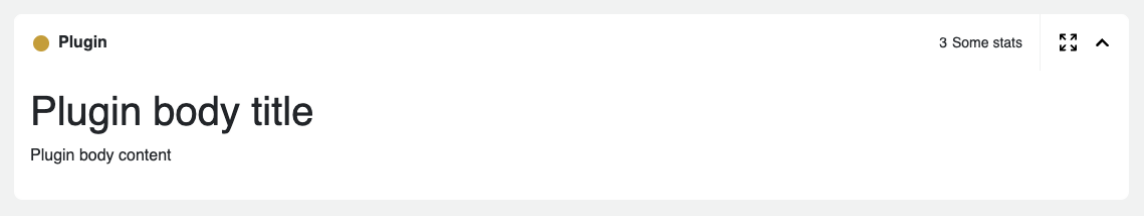

Figure 3-9: Plugin Bar available inside the asset page

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.

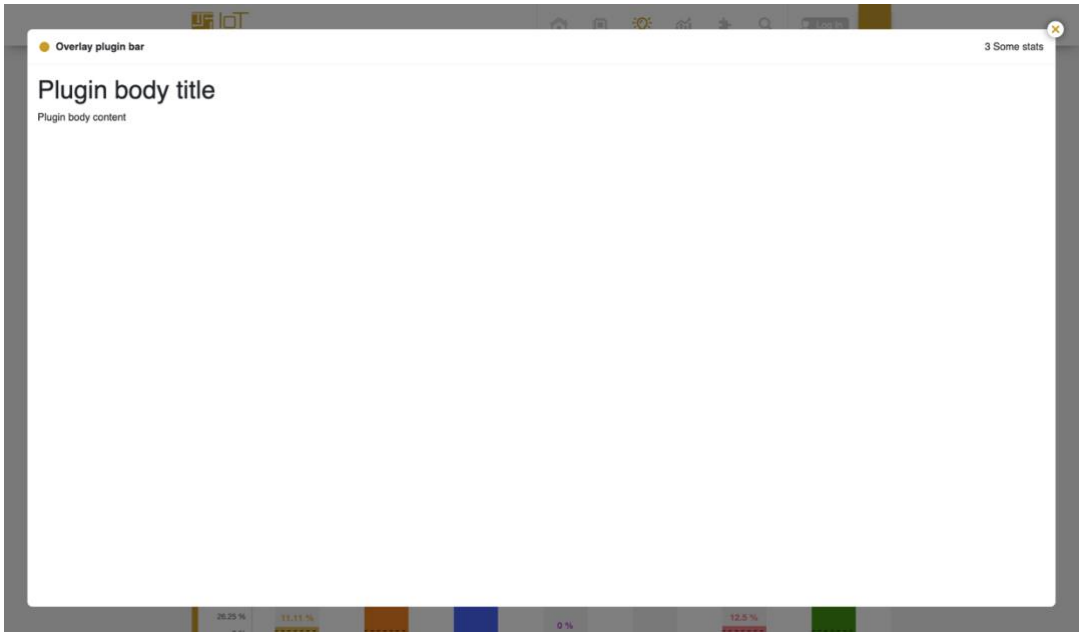

Figure 3-10: Plugin Overlay bar view

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.

Figure 3-11: Plugin Page

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[3]. This allows users to easily test out the API and see the available functionalities. The endpoints to edit the assets are listed in Figure 3-12.

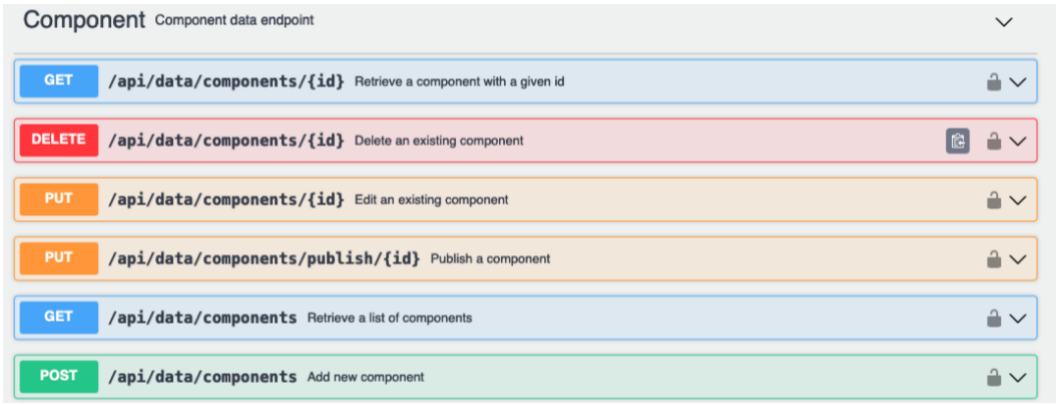

Figure 3-12 – FDAC REST API Swagger

The search endpoint, which allows users to submit queries to the FDAC for components that match specific criteria, is shown in Figure 3-13. 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.

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

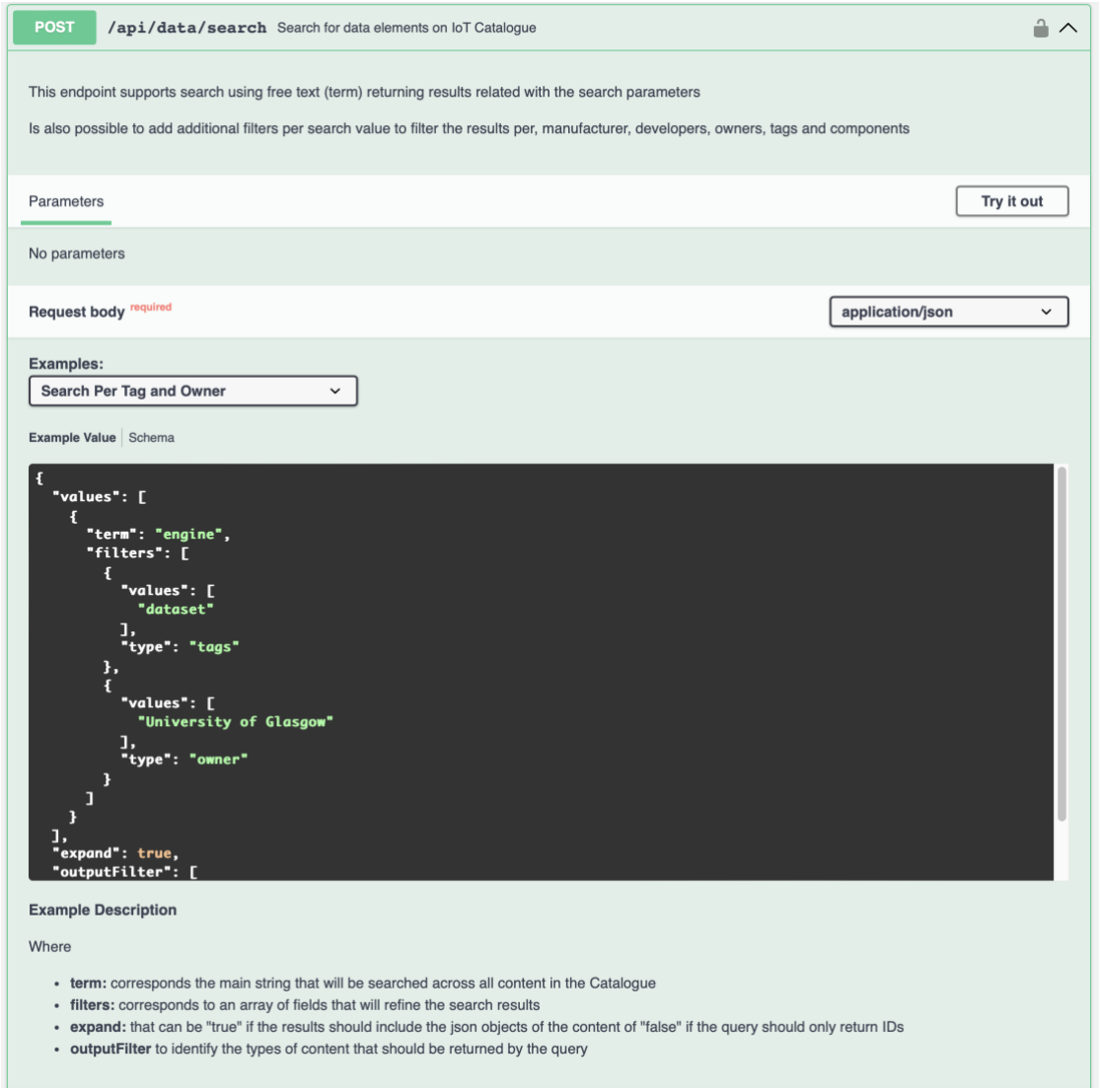

Figure 3-13 - FDAC REST API Search response example

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. 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 exposing of API methods that support the description of the information in other formats than the ones used internally in *FDAC*.

3.2.2 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 Relevant Ontologies

3.2.2.1.1 INFINITECH Datapack

The INFINITECH project[4] 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 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.

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

3.2.2.1.2 Data Catalogue Vocabulary (DCAT)

Beyond these Ontologies, there is another one that is important to refer to, the **Data Catalogue Vocabulary (DCAT)** [9]. 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.

3.2.2.2 Ontology Engineering tools

3.2.2.2.1 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 3-14, 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.

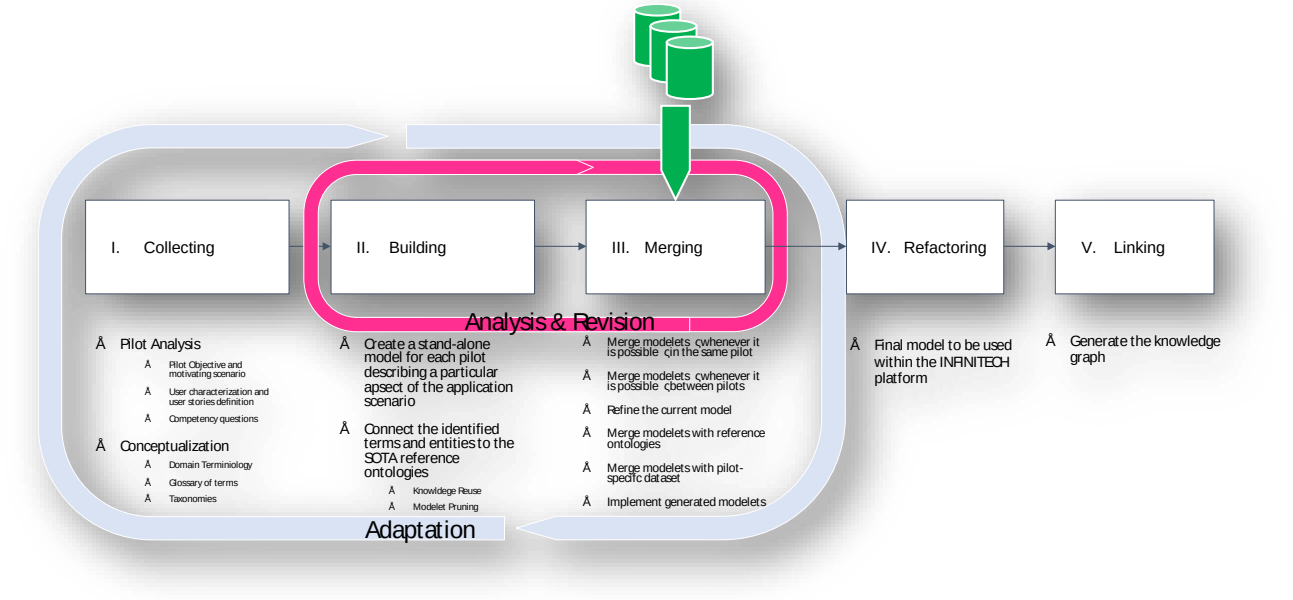

Figure 3-14 - INFINITECH Methodology for Ontology Engineering [1]

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.

3.2.2.2.2 INFINITECH Semantic Validator

The semantic validator (presented in Figure 3-15) 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.

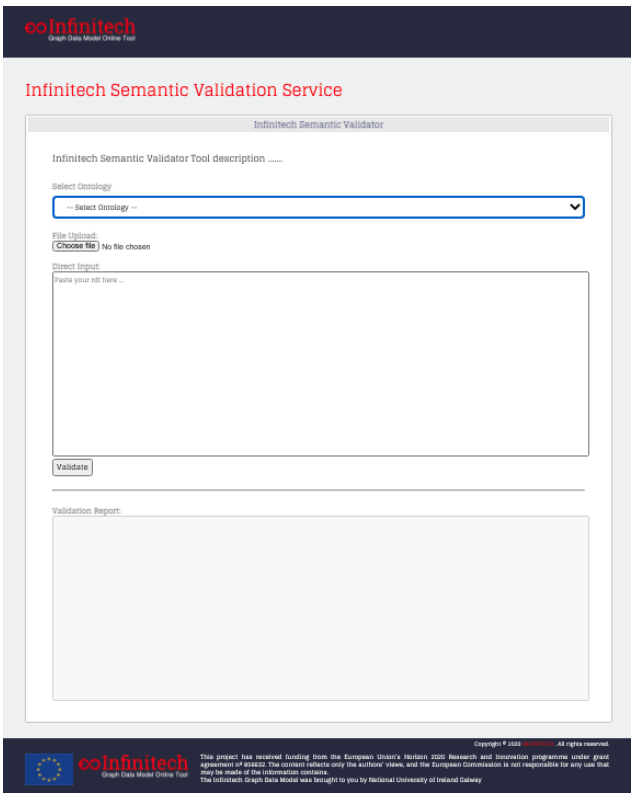

Figure 3-15 – INFINITECH Semantic Validator Online Tool

3.2.2.2.3 TAG-Tool

Another tool developed at Nova School of Science and Technology[5] is the Translators Automatic Generation Tool (TAG-Tool), which automatically generates translators to support the communication between heterogeneous systems/devices. As presented in Figure 3-16, 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.

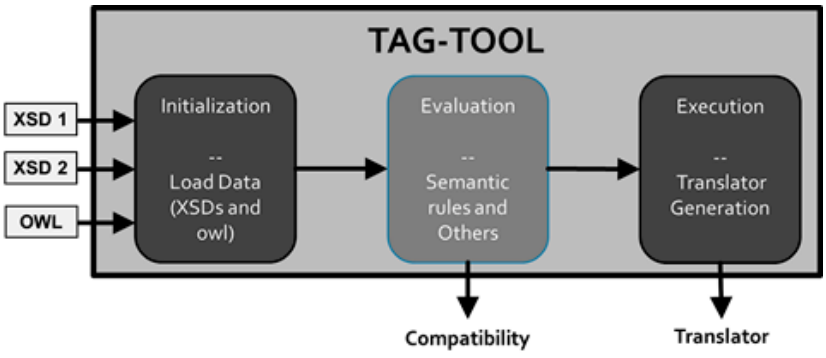

Figure 3-16 – TAG-Tool Interface and Phases

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,

[5] 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 3-17).

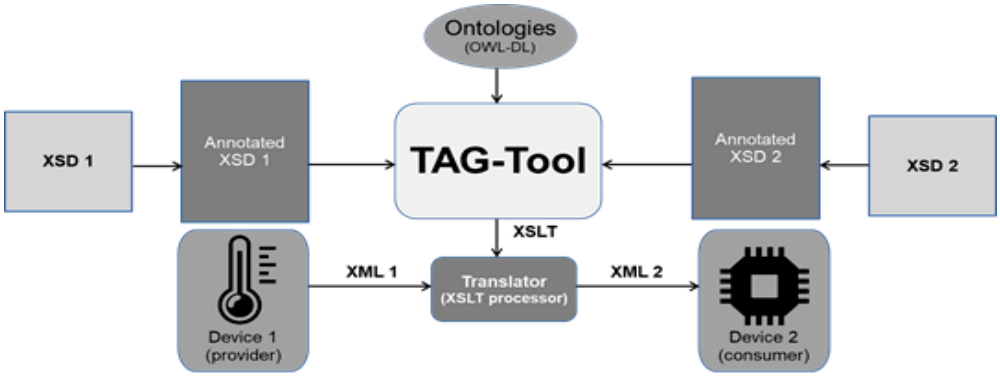

Figure 3-17 – TAG-Tool application scenario

3.2.2.2.4 Semantic Interoperability Toolkit for Data Marketplaces

The Semantic Interoperability Toolkit for Data Marketplaces [10] 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 [11] [12] [13] [14], 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) [15]. 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.

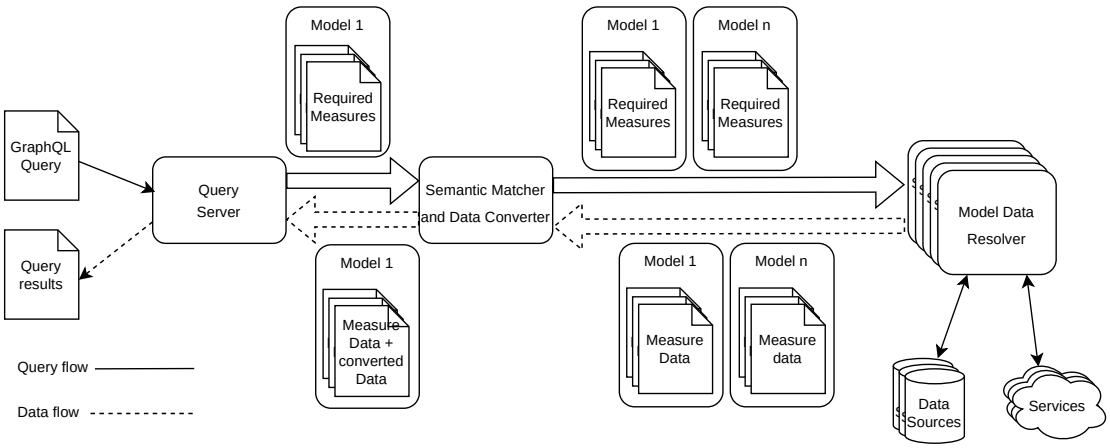

Figure 3-18 - Overview of the semantic query system

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 [10] 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.

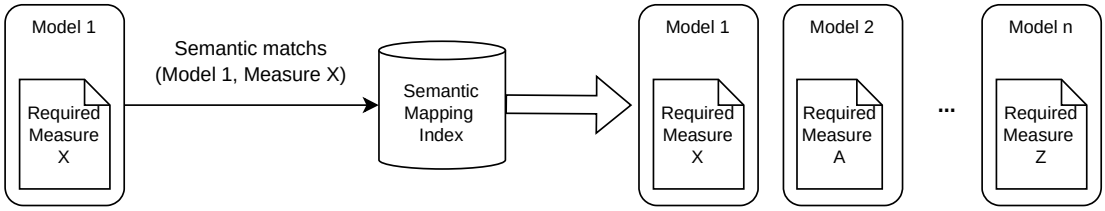

Figure 3-19 - Semantic matching discover process

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 3-20. 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.

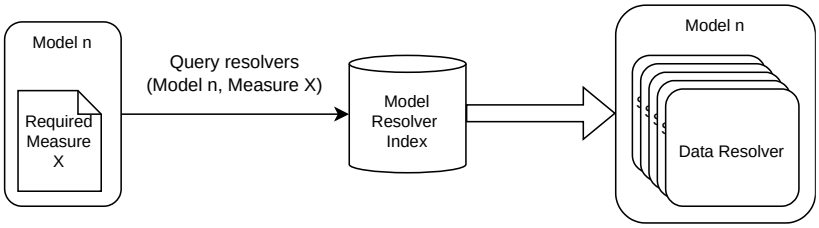

Figure 3-20 - Detail of data resolver discovery process

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.

3.2.3 Technical Specification

3.2.3.1 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 3-21.

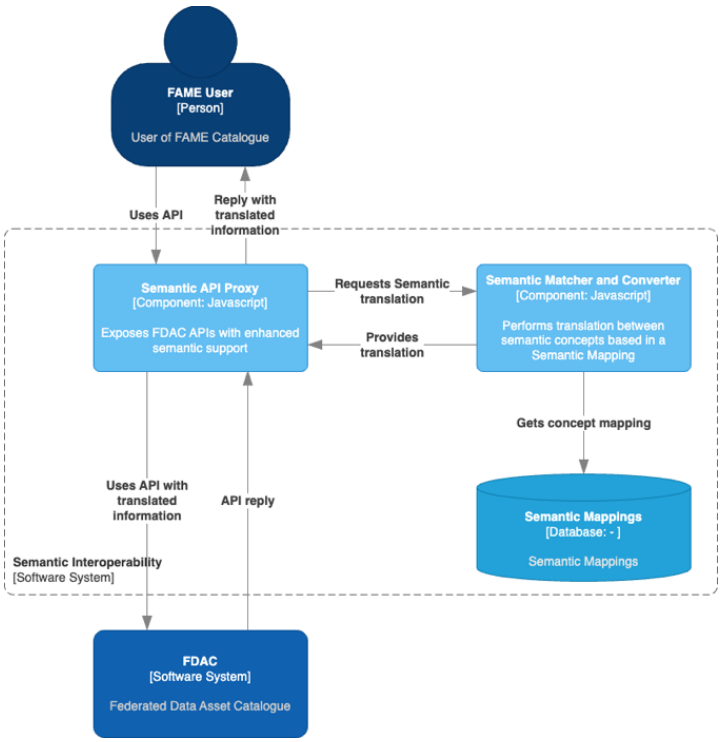

Figure 3-21 – Architecture of Semantic Middleware

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.

3.2.3.2 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 3-22 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.

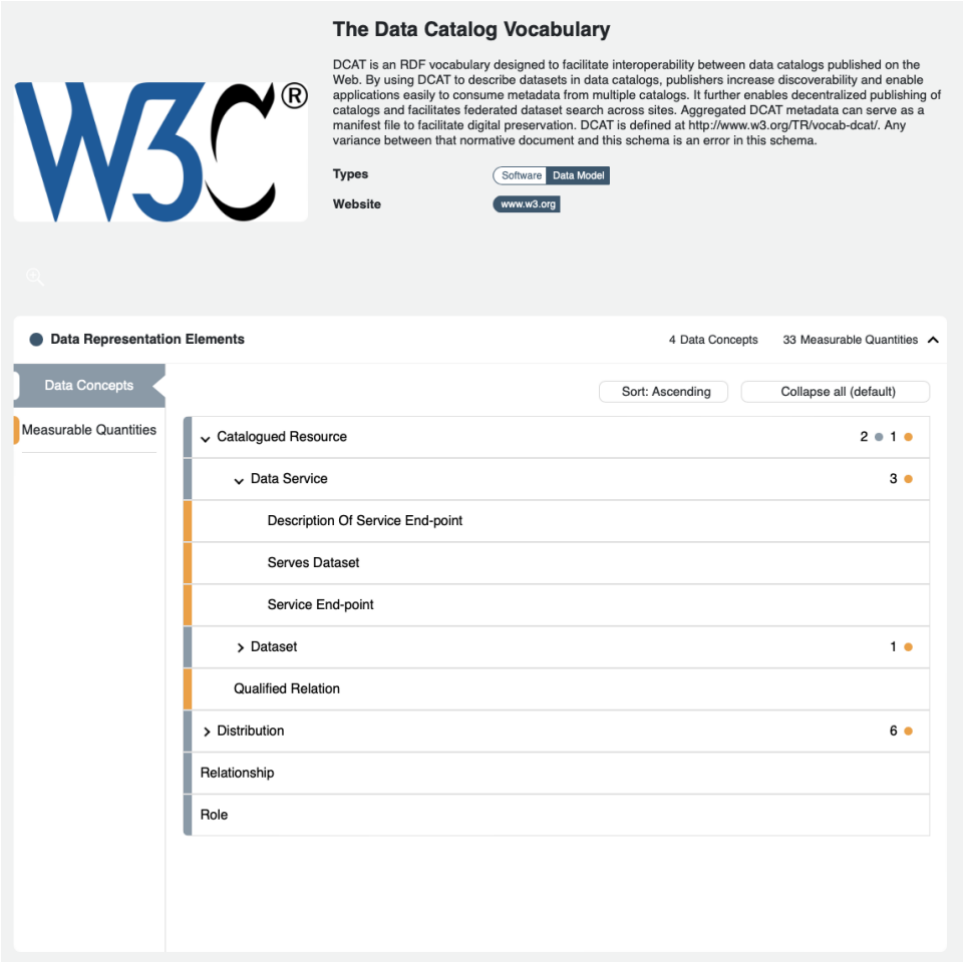

Figure 3-22 – Example do DCAT model represented in FDAC!

3.2.3.3 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 3-23. In this form are shown the best semantic concepts of other models that have the highest score.

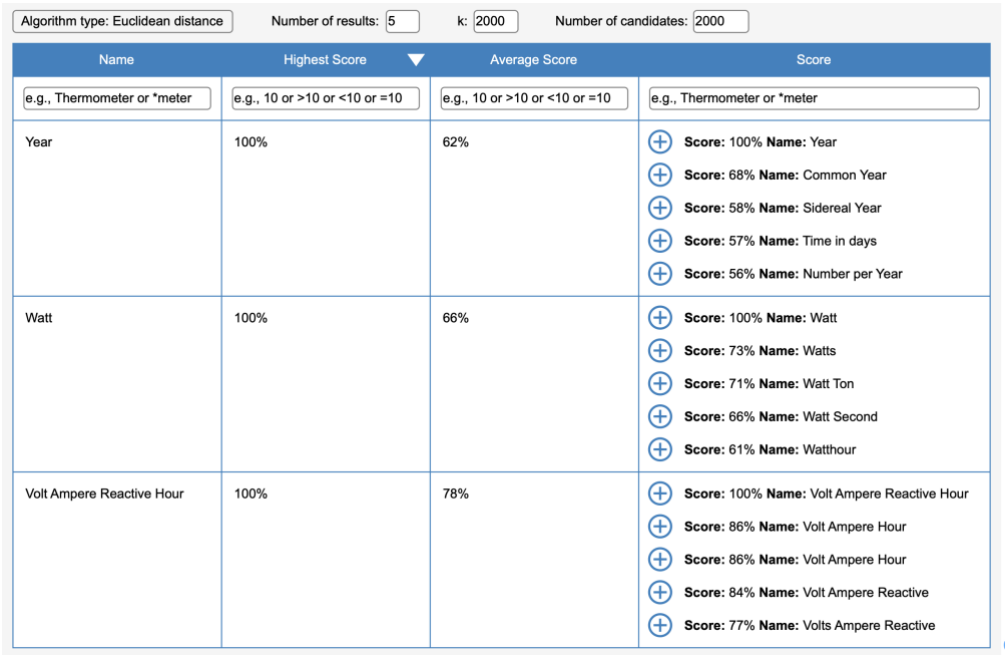

Figure 3-23 – Web form used to analyse the results from Semantic Similarity

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 [16]. 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 [10]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.

3.2.3.4 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.

3.2.4 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 3-24. This description is accessible here[6].

Figure 3-24 – 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 3-25. Currently, only DCAT is supported.

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

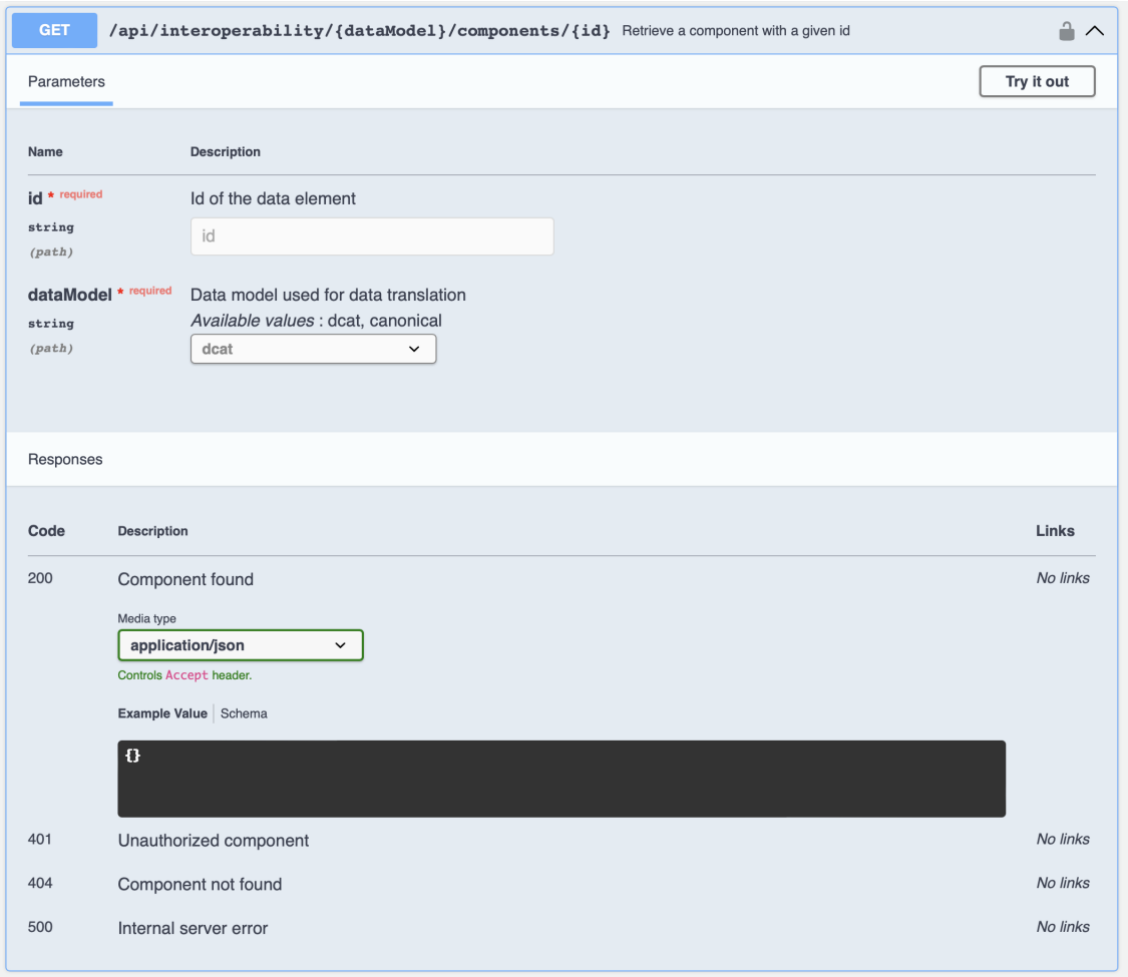

Figure 3-25 – Details about the invocation of a endpoint provided by the Semantic Proxy

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",
```

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.

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

```
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"
```

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.

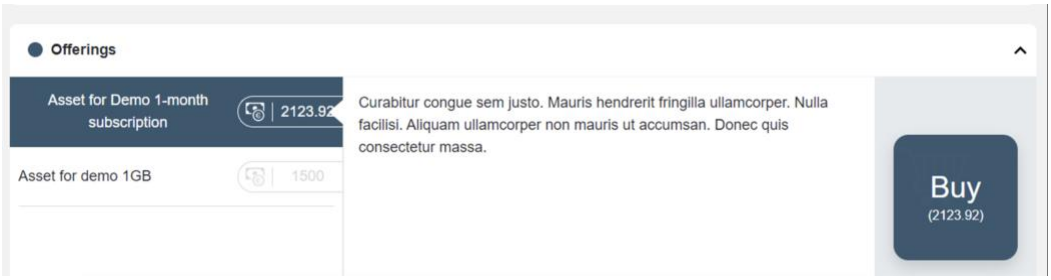

Figure 4-1 – Offerings plugin used on the Trading and Monetization component

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"
    }
  ],
```

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": {
```

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.

The next steps in the development of the FDAC include the integration of the User Authentication System used by FAME, allowing a seamless experience for authenticated users during the process of asset addition, discovery, and edition. This integration is complemented with the integration of the Policy Manager system which allows assessing if a specific user has the needed permissions to visualize and edit assets. Further development also needs to be employed on the plugin system to refine the tools available for 3rd-parties to develop their plugins and to allow information added by the plugins to be discoverable by the search mechanism and corresponding API.

Regarding the next steps in the development of the Semantic Interoperability Middleware, the major actions will be the identification of semantic ontologies relevant to the pilots and their domain and the improvement of the approach for the generation of semantic matchings between semantic ontologies aiming to achieve an automatized process.

Table 2 presents the current state of the KPIs related with this deliverable.

Table 2 – D3.2 Related KPIs

Obj. ID	KPI ID	Measured KPI	Expected Value [*M9*]	Achieved Value [*M9*]	Expected Value [*M14*]	Achieved Value [*M14*]
O3	**3.1**	Ontologies to be supported	0	0	2	2
O3	**3.2**	Reduction of time to discover cross-sector datasets	0	0	0	0
O3	**3.3**	Data Assets in FDAC	0	0	5	64
O3	**3.4**	Repurposing instances of datasets in multiple applications	0	0	0	0
O3	**3.5**	Number of downloads of data assets of the FAME Catalogue	0	0	0	0

6 References

[1] INFINITECH, D4.2 – Semantic Models and Ontologies III, (2022).

[2] FAME, D2.1 – Requirements, Specifications and Co-Creation, (2023).

[3] FAME, D2.2 – Technical Specifications and Platform Architecture, (2024).

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

[5] DDP Specification, Meteor, 13 05 2021, https://github.com/meteor/meteor/blob/devel/packages/ddp/DDP.md (retrieved 2024-05-03).

[6] What is RESTful API, AWS, https://aws.amazon.com/what-is/restful-api/ (retrieved 2024-05-03).

[7] OpenAPI, OpenAPI Initiative, https://www.openapis.org/ (retrieved 2024-05-03).

[8] FDAC REST API Unparallel Innovation, https://fame-fdac.iot-catalogue.com/swagger/653a8be4b93bd08e33dd5e8e (retrieved 2024-05-03).

[9] W3C, Data Catalog Vocabulary (DCAT) – Version 3, https://www.w3.org/TR/vocab-dcat-3/. (retrieved 2024-01-26).

[10] M. M. a. B. A. a. G. R. a. G. R. a. A. G. a. T. T. a. P. Malo (2023), 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.

[11] Z. X. a. X. G. R. Zhang (2023), 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.

[12] X. C. a. D. P. D. Liu (2019), 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.

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

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

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

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