Federated decentralized trusted dAta Marketplace for Embedded finance

D2.4 - Integrated FAME Data Marketplace II

Title	D2.4 - Integrated FAME Data Marketplace II
Revision Number	2.0
Task reference	T2.3, T2.4
Lead Beneficiary	GFT
Responsible	Antonio Sottosanti
Partners	ATOS, ECO, ENG, IDSA, JOT, KM, LXS, MOH, NOVA, NOVO, UNP, UPRC
Deliverable Type	DEM
Dissemination Level	PU
Due Date	2025-06-30 [Month 30]
Delivered Date	2025-07-31
Internal Reviewers	NOVO INNOV
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	2025-05-13	GFT, UPRC	Initial Table of contents
0.2	2025-05-30	GFT, UPRC	Updated Chapter 2, Federation Process and User Journeys
0.3	2025-06-15	GFT , UPRC, ENG, FTS	Updated chapter 3, dashboard and federation Application (new) and Backend Integration
0.4	2025-06-20	GFT	Updated Chapter 2, Federation Process and User Journeys
0.5	2025-07-03	FTS	Added chapter 5, FAME Connector
0.6	2025-07-09	GFT , FTS, ETONEC	Added Chapter 8, FAME Scalability in the EU Strategy
0.7	2025-07-15	UNP	Updated Chapter 4, Data Asset Integration
0.8	2025-07-18	GFT	Updated Chapter 1 and 9. Introduction and Conclusion.
1.0	2025-07-20	GFT, INNOV, NOVO, UPRC	Version for Peer and Quality Assurance reviews
2.0	2025-07-31	GFT	Final Version for submission

Definitions

Acronym	Definition
AAI	authentication authorization infrastructure
AI	Artificial Intelligence
API	Application Programming Interface
APM	Assets Policy Management
AWS	Amazon Web Services
CBDC	Central Bank Digital Currency
CD	Continuous Development
CI	Continuous Integration
CPU	Central Processing Unit
CRUD	Create Retrieve Update Delete - Basic Operations in DBMS
CSS	Cascading Style Sheets
DCAT	Data Catalog Vocabulary
DLT	Distributed Ledger Technologies
DNS	Domain Name Server
ECB	European Central Bank
EIDAS	Electronic IDentification, Authentication and trust Services
ESG	Environmental, Social and Governance
EU	European Union
FAME	Federated decentralized trusted dAta Marketplace for Embedded finance
FAQ	Frequently Asked Questions
FDAC	Federated Data Assets Catalogue
FFGA	FAME Federation Governance Application
FGB	FAME Governance Board
FMAP	FAME Marketplace Platform
GA	Grant Agreement|General Assemply
GDPR	General Data Protection Regulation
GOV	Operational Governance
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
HW	HardWare
ID	Identity
IP	Internet Protocol
IT	Information Technology
JSON	JavaScript Object Notation

JWT	JSON Web Token
KYB	Know Your Business
KYC	Know Your Customer
LC	Learning Center
ML	Machine Learning
NFT	Non Fungible Token
OIDC	OpenID Connect
ORM	Object-Relational Mapping
OS	Operating System
OTP	One-Time Password
PAT	Pricing Advisory Tool
PDF	Portable Description Format
POC	Proof of Concept
POD	Pay On Delivery|Point Of Distribution
PSP	Payment Service Provider
RAM	Random Access Memory
RBAC	Role-Based Access Control
REST	Representational State Transfer
SA	Solution Architecture
SCSS	Source Code Control System
SEO	Search Engine Optimization
SLA	Service Level Agreement
SSH	Secure SHell
SSI	Server Side Includes
SSL	Secure Sockets Layer
SSO	Single Sign On
SUB	Subroutine
TM	Trade and Monetisation
UI	User Interface
URL	Uniform Resource Locator
VM	Virtual Machine
XAI	Explainable Artificial Intelligence

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

Executive Summary

The EU has made significant strides in harmonizing digital markets, exemplified by the European Strategy for Data [1]. Building on this, the Data Governance Act [2] and Data Act [3] aim to foster data sharing and trust within the EU.

Data Spaces are identified as strategic infrastructures to drive the European data economy while minimizing environmental impact. To address the challenges of trust, legal compliance, and technical interoperability, FAME emerges as a distinct Federated Data Space.

Tailored for the financial sector, FAME offers a unified platform for diverse data assets, including AI models, analytics, and executable services. Its federated governance model ensures secure, compliant data transactions. Leveraging advanced technologies like blockchain and AI, FAME supports dynamic identity and access management for sustainable self-sovereign identity adoption.

This document outlines the FAME-enabled process and user stories, providing a comprehensive overview of the integrated architecture underpinning the FAME Solution Architecture (SA).

This document introduces the FAME Integrated solution, outlining how independently developed modules have been combined into a cohesive system. By adopting a microservices approach, the architecture prioritizes flexibility and scalability.

The document describes a user-friendly dashboard connected to robust backend APIs. While providing a comprehensive overview of backend module integrations, detailed specifications are available in separate documents. A Continuous Integration/Continuous Delivery (CI/CD) infrastructure to support development and deployment is also detailed, ensuring efficient and reliable delivery through automated build, test, and deployment processes.

Leveraging DevOps principles and modern technologies like containers and microservices, the marketplace will be built using an agile, iterative approach. Integration of technical modules from other project phases and open-source data management tools, including data cleansing and synthetic data generation capabilities, is also planned and will be accomplished in the second part of the project.

This document presents the M30 results of Task 2.3 and Task 2.4. It outlines the developments in the CI/CD infrastructure, architectural integrations, and the enhancements of modules previously introduced in the earlier version of the same document.

In particular, it details the evolution of the user interfaces and the implementation for the Federation Application, the Dashboard, the Marketplace, and the Connector module. The FAME Connector is component of the FAME platform’s Federated Data Space architecture, enabling secure and transparent data exchange while allowing providers to retain control over their data on-premises.

The concluding section explores how FAME is poised to evolve and align itself with the regulatory and technological frameworks that are currently being finalised. This evolution is closely tied to the broader normative and technological advancements being driven by the European Union. In particular, the discussion highlights key initiatives such as eIDAS 2, the European Blockchain Services Infrastructure, and the development of the Digital Euro, illustrating how FAME can adapt to and support these emerging standards.

Table of Contents

1 Introduction 7
1.0 Objective of the Deliverable 8
1.1 Insights from other Tasks and Deliverables 8
1.2 Structure 9
1.3 Summary of Changes 10
2 Federation Process & User Journeys 12
2.0 Introduction 12
2.1 User Journeys 12
2.1.0 Onboarding of Federation Member (Federation Process) 12
2.1.1 Casual Navigation 15
2.1.2 Organization Federation 15
2.1.3 User Technical Onboarding 16
2.1.4 Asset Preparation by Producer Application 16
2.1.5 Asset Publishing 16
2.1.6 Asset Publishing from other Marketplace (Application to FAMP - M2M) 16
2.1.7 Define Offering 16
2.1.8 Asset Discovery for Trading 17
2.1.9 Asset Purchase 17
2.1.10 Asset Use by Consumer Application 17
2.1.11 User Offboarding 18
3 FAME Frontend & Backend Integration 19
3.0 Introduction 19
3.1 FAME Federation Application 22
3.1.0 FFGA design process 22
3.1.1 FFGA development process 23
3.2 FAME Dashboard 33
3.2.0 Dashboard design process 34
3.2.1 Dashboard development process 48
3.3 Integration of Frontend & Backend Services 73
3.3.0 Integration Methodology 73
3.3.1 Integration of FAME Federation Application & Backend Services 76
3.3.2 Integration of Dashboard & Backend Services 79
3.3.3 Internal Integration of Backend Services 94

4 FAME Data Assets Integration 99
4.0 Introduction 99
4.1 Federated Data Asset Catalogue (FDAC) 99
4.1.0 REST API 100
4.1.1 Iframe 101
4.2 Integration 102
4.2.0 Dashboard Integration 102
4.2.1 Asset Policy Manager Integration 103
5 FAME Connector 104
5.0 Introduction 104
5.1 Overview of Data Consumption Components 104
5.1.0 Component Descriptions 105
5.2 Justification of Decentralized Data Storage 105
5.3 Current State and Implementation Approach 105
5.3.0 Data Gate Overview 106
5.3.1 Proof-of-Concept Data Access Portal 107
5.4 Provider-Side Implementation Requirements 111
5.4.0 Prerequisites 111
5.4.1 Implementation Steps 111
5.5 Consumer-Side Data Access Workflow 111
5.5.0 Prerequisites 111
5.5.1 Access Steps 111
5.5.2 Detailed Sequence Diagrams 112
6 FAME CI/CD Integration Infrastructure 115
6.0 Introduction 115
6.1 Microservices Approach 115
6.1.0 Microservices with Containers 116
6.1.1 Kubernetes Containers Orchestration 117
6.2 Development View 119
6.3 Deployment View 121
6.3.0 Exposing the Application 122
6.3.1 Storage & Volumes Management 123
7 Blueprint Guidelines for FAME Deployment 124
7.0 Guidelines Overview 124
7.0.0 Development Cycle 125
7.1 Building a FAME Component 126

7.2 Building an Image with Pre-compiled Libraries/Binaries 126
8 FAME Scalability in the EU Strategy 127
8.0 Introduction 127
8.1 FAME & EIDAS Compliance 127
8.1.0 Background 127
8.1.1 Potential Impacts on the FAME Project 129
8.2 European Blockchain 130
8.2.0 The European Blockchain Services Infrastructure (EBSI) 130
8.2.1 Similarities Between FAME and EBSI 130
8.2.2 Potential Options to Use EBSI with FAME for Scalability 131
8.3 FAME & Digital Euro 133
8.3.0 Introduction 133
8.3.1 Digital Euro integration in FAME 133
8.3.2 Design implementation hypothesis 135
8.3.3 Federation PSP Mockup User interface 138
8.3.4 Advantages and Disadvantages 141
9 Conclusions 142
References 143

List of Figures

Figure 1 : Conceptual view of FAME 19
Figure 2 : FAME C4 architecture 20
Figure 3 : Home page 25
Figure 4: Trusted Community page 27
Figure 5: Federation page 28
Figure 6: Login page 30
Figure 7: Control panel page 31
Figure 8: Request details 32
Figure 9 : Positioning of FAME Dashboard within FAME SA 33
Figure 10 : FAME Dashboard design process 34
Figure 11 : FAME Dashboard components' requirements collection 36
Figure 12 : FAME Dashboard components' needed UIs feedback 36
Figure 13 : FAME Dashboard components' UIs interactions feedback 36
Figure 14 : FAME Dashboard end-users' Visual Sitemap 39
Figure 15 : FAME Dashboard administrators' Visual Sitemap 40
Figure 16 : FAME Dashboard administrators' home page Wireframe 41
Figure 17 : FAME Dashboard templatized visual elements 41
Figure 18 : FAME Dashboard end-users' skeleton page 43
Figure 19 : End-users' home page Mockup 44
Figure 20 : FAME Dashboard administrators skeleton page 45
Figure 21 : Administrators' home page Mockup 46
Figure 22 : Home page 55
Figure 23 : About Us page 56
Figure 24 :Detailed pages 57
Figure 25 : Helpdesk page 58
Figure 26 : Login page 59
Figure 27 : Profile page 60
Figure 28 : Trading Account Enrollment page 60
Figure 29 : Asset Publishment page (1) 61
Figure 30 : Asset Publishment page (2) 62
Figure 31 :Asset Publishment page (3) 63
Figure 32 : Data Assets Catalogue page 64
Figure 33 : Asset Details page 65
Figure 34 : Offering Details page (1) 66
Figure 35 : Offering Details page (2) 67
Figure 36 : Offering Definition page (1) 68
Figure 37 : Offering Definition page (2) 69
Figure 38 : Asset Policy Manager page 70
Figure 39 : Learning Centre page 70
Figure 40 : Course Details page 71
Figure 41 : Terms and Conditions page 72
Figure 42: Asset Publishment 96
Figure 43: Offering Publishment 96
Figure 44: P&T service endpoints for internal integration 97
Figure 45: GOV service endpoints for internal integration 98
Figure 46 : FDAC in C4 Architecture 99

Figure 47 : FDAC REST API Swagger 100
Figure 48 : FDAC REST API Search response example 101
Figure 49 : Highlight of FDAC's iframe to show list of assets 102
Figure 50 : Highlight of FDAC's iframe to show asset details 103
Figure 51 : Information flow of the integration between Dashboard, APM and FDAC Iframe 103
Figure 52 : high-level architecture of the FAME Connector 104
Figure 53 Data Gate API 106
Figure 54 : API swagger documentation 107
Figure 55 : Demo Portal. List of Available Data Assets 108
Figure 56 : Demo Portal, Validating the Data Customer Signature 109
Figure 57 : Demo Portal, Access Restriction Message 109
Figure 58 : Demo Portal, Data Access Page 110
Figure 59 : Data Access via Data Gate for subscription-based token 112
Figure 60 : Data Access via Gate for Pay-As-You-Go 113
Figure 61 :Data Access via Gate for Pay-As-You-Use 114
Figure 62 : Monolithic vs Microservice 116
Figure 63 : VM vs Container 116
Figure 64: Container kernel properties 117
Figure 65 : Kubernetes 119
Figure 66: Graphical view of a POD in Kubernetes 119
Figure 67: FAME CI/CD Infrastructure - Logical View 120
Figure 68: Kubernetes Cluster 121
Figure 69: Kubernetes Persistent Volumes 123
Figure 70: FAME CI/CD Workflow 124
Figure 71 : Logical Flow 135
Figure 72: Acquire sequence diagram 136
Figure 73: Redeem sequence diagram 137
Figure 74: Login page 138
Figure 75: Wallet Page 139

List of Tables

Table 1 :D2.4 Updates from D2.3 10
Table 2 : Status of backend services UIs 37
Table 3 : API endpoints of Federation Application and Dashboard integration. 81
Table 4 : API endpoints of Regulation Tool and Dashboard integration 81
Table 5 : API endpoints of FDAC and Dashboard integration. 82
Table 6 : API endpoints of Learning Centre and Dashboard integration. 83
Table 7 : API endpoints of TM and Dashboard integration. 83
Table 8 : API endpoints of PT (Internal) and Dashboard integration. 84
Table 9 : API endpoints of PT (Public) and Dashboard integration. 85
Table 10 : API endpoints of GOV (Internal) and Dashboard integration. 86
Table 11 : API endpoints of GOV (Public) and Dashboard integration. 88
Table 12 : API endpoints AAI and Dashboard integration. 90
Table 13 : API endpoints PAT and Dashboard integration. 91
Table 14 : API endpoints Advanced Search and Dashboard integration. 93
Table 15 : API endpoints APM and Dashboard integration. 93

1 Introduction

For Europe to operate cooperatively and in accordance with European principles, such as self-determination, privacy, openness, security, and fair competition, the digital market is essential.

A thriving digital market is crucial for Europe to operate cohesively in line with principles of self-determination, privacy, openness, security, and fair competition. The expanding data economy needs a legal framework to define data protection, fundamental rights, safety, and cybersecurity. The harmonization of digital markets is one of the fundamental policy achievements of the European Union (EU), with the European Strategy for Data [1] serving as its primary tangible product.

The Data Governance Act [2] comes next, with the goal of promoting usable data by boosting EU-wide data sharing procedures and enhancing trust in data intermediaries.

The Data Act [3], a legislative proposal that intends to establish a framework that will promote business-to-government data sharing, is also another vision of the EU data economy.

The European Commission has identified Data Spaces as strategic infrastructures to promote the European data economy's growth while minimizing human and environmental carbon footprints.

These infrastructures ensure reliable data sharing and exchange based on agreed principles. This entails numerous challenges:

- **Business/Organizational Challenges**: Companies must build trust, adhere to EU rules, and keep pace with market changes.
- **Legal Compliance Challenges**: GDPR and other legal rules protect privacy and determine data ownership, aligning with EU-wide policies.
- **Technical Challenges**: Interoperability, provenance, quality assurance, and scalability issues in data sharing solutions require attention.

FAME is a Federated Data Space and distinguishes itself from other data marketplaces in several key aspects:

- **Market Focus**: FAME, tailored for the financial sector, provides a unified platform for customized Data Asset utilization, fostering innovation and efficient service delivery.
- **Products Diversity**: FAME offers diverse federated data assets, including classical datasets, value-added assets like AI/XAI models, analytics, algorithms, executable services, and educational content.
- **Legal/ Governance Model**: FAME implements a federated governance model to ensure trustworthy, sovereign, private, and secure data transactions in line with EU regulations, fostering transparency and consistency.
- **Cutting-edge Technologies**: FAME employs advanced technologies like semantic interoperability standards, AI, machine learning for enhanced analytics, and blockchain for secure, transparent transactions, and asset tokenization, adaptive authentication and authorization infrastructure (AAI) to deal with dynamic target platforms, enabling adaptability for dynamic identity and access management, dynamic policy management process enables policy adaptability for dynamic identity and access management (IAM) for sustainable adoption of SSI (self-sovereign identity).

Aligned with the presented vision, this document details the process and user stories supported by FAME and provides a comprehensive overview of the integrated architecture underpinning the FAME SA solution.

This document presents the Month 30 (M30) results of Task 2.3 and Task 2.4 within the scope of the FAME project. It provides a comprehensive overview of the progress made in the development and refinement of the CI/CD infrastructure, the integration of architectural components, and the enhancement of modules previously introduced in earlier iterations of this report.

A particular focus is placed on the evolution of the user interfaces across several key components: the Federation Application, the Dashboard, the Marketplace, and the FAME Connector module. The FAME Connector serves as a fundamental element within the Federated Data Space architecture of the FAME platform. It facilitates secure, scalable, and transparent data exchange between providers and consumers, while ensuring that data remains under the control of the original providers, hosted on their own premises.

The final section of the document explores the potential for FAME to evolve in alignment with emerging regulatory frameworks and technological standards currently being finalised by the European Union. This includes considerations related to eIDAS 2, the European Blockchain Services Infrastructure (EBSI), and the Digital Euro initiative. These developments are expected to shape the future landscape of digital identity, decentralised data exchange, and digital finance, and FAME is positioned to adapt accordingly.

1.0 Objective of the Deliverable

This document introduces the FAME Integration Architecture. It outlines the processes and user stories derived from the requirements analysis and details how independently developed modules have been integrated. The architecture leverages a modern microservices approach, enhancing flexibility and scalability.

The document describes a user-friendly dashboard and its connection to backend APIs. While a comprehensive overview of backend module integrations is provided, detailed specifications can be found in dedicated deliverables.

Additionally, the document presents the CI/CD pipeline, which automates build, test, and deployment processes, ensuring efficient and reliable delivery of FAME technologies and pilots.

1.1 Insights from other Tasks and Deliverables

The FAME Integration Architecture represents a critical milestone in the project. Building upon the requirements outlined in the FAME Solution Architecture (D2.6 - Technical Specifications and Platform Architecture II [5]) and the detailed requirements analysis (D2.5 - Requirements Analysis, Specifications and Co-Creation II [6]), this architecture defines the principles and methods for integrating the various components of the FAME platform.

It integrates findings from previous work packages, including the development of core functionalities (D3.5 - Federated Data Assets Catalogue II [7], D4.4 - Blockchain-based Data Provenance Infrastructure II [8], D4.5 - Pricing, Trading and Monetization Techniques II [9]), the design of the user interface (D3.6 - Mechanisms and Tools for Regulatory Compliance II [10]), and the initial development of analytical assets (D5.1 - Trusted and Explainable AI Techniques I [11], D5.2 - Energy Efficient Analytics Toolbox . I [12]).

This integration architecture provides a roadmap for future work, such as the development of the FAME Marketplace (WP2, WP3, WP4), the integration of advanced analytical capabilities (WP5), pilot implementations in real-world scenarios (WP6), and exploitation/dissemination of project results (WP7).

1.2 Structure

The deliverable is structured as follows:

- *Section 1* (current Section) includes an introduction FAME project's vision including the objectives of the current deliverable, as well as the relation of this deliverable with the existing ones and the project's technical WPs
- *Section 2* details the user journeys and flows resulting from the requirements analysis. It outlines how users and external systems interact with the system and describes the principles and process of joining the federated network.
- *Section 3* presents the dashboard through which users interact with the system. It details how the dashboard integrates with backend systems and how the various backend modules interact with each other.
- *Section 4* describes how managed assets are integrated into the FAME Federated Catalogue and how they are managed.
- Section 5 describes the FAME Connector, the component of the FAME platform's Federated Data Space architecture, enabling secure and transparent data exchange while allowing providers to retain control over their data on-premises.
- *Section 6* documents the DevOps practices for the FAME project. It illustrates the application of the FAME CI/CD architecture and its role in the development and integration of the FAME marketplace components.
- *Section 7* documents the guidelines and specific CI/CD processes defined for building the FAME platform, both for platform-specific components and for the development of pilot components that utilize the project infrastructure.
- *Section 8* describe FAME's potential to evolve in line with key EU initiatives—eIDAS 2, EBSI, and the Digital Euro—which are set to shape the future of digital identity, decentralised data exchange, and digital finance, with FAME well-positioned to adapt accordingly.
- *Section 9* contains the conclusions drawn from the FAME integration architecture.

1.3 Summary of Changes

Compared with the previous version of this deliverable (D2.3 - Integrated FAME Data Marketplace I), the Sections/sub-Sections that have been updated /added within D2.4 are depicted in Table 1.

Table 1: D2.4 Updates from D2.3

Section	Status	Description of Update/Addition
		Section 1
1 - Introduction	Updated	Alignment regarding the technical activities and progress of the project
1.1 - Insights from other Tasks and Deliverables	Updated	Alignment of D2.6 input regarding the technical activities and progress of the project.
1.2 - Structure	Updated	Alignment of the document structure with the new sections from the previous version D2.3
1.3 - Summary of Changes	New	Description of updates between D2.4 and D2.3.
		Section 2
2.1 - User Journeys	Updated	Alignment to the progress of the project
		Section 3
3.1 - FAME Federation Application	New	New section describing the FAME Federation Application and Dashboard, design and development process, including a comprehensive user manual detailing UI
3.2 - FAME Dashboard	Updated/ New	This section has been revised to reflect enhancements in the Dashboard's design and development process, including a comprehensive user manual detailing its UI
3.3 - Integration of Dashboard & Backend Services	Update/New	Alignment to the progress of the project. Added detail the services that are interacting with the Federation Application and the Dashboard and the integration of backend services
		Section 4
4 - FAME Data Assets Integration	Updated	Alignment to the progress of the project of the integration mechanism provided by Federated Data Asset Catalogue (FDAC) to support the integration with the other FAME components
		Section 5
5 - FAME Connector	New	The Section describe the component of the FAME platform's Federated Data Space architecture, enabling secure and transparent data exchange while allowing providers to retain control over their data on-premises
		Section 6
6 - FAME CI/CD Integration Infrastructure	Updated	This section illustrates the application of the FAME CI/CD architecture and has been updated with the introduction of the Argo CD continuous deployment tool.
		Section 7
7 - Blueprint Guidelines for FAME Deployment	Updated	This section outlines the guidelines and specific CI/CD processes defined for building the FAME platform and has been updated in accordance with the infrastructure updates (addition of Argo CD).

		Section 8
8 - FAME Scalability in the EU Strategy	New	This section outlines how FAME is set to align with upcoming EU regulations and standards, including eIDAS 2, EBSI, and the Digital Euro, positioning it to adapt to the evolving digital identity and finance landscape.
		Section 9
9 - Conclusions	Updated	The conclusion section has been revised and expanded to incorporate the updates introduced in this version of the document (D2.4), highlighting the key differences and enhancements made in comparison to the previous version (D2.3).

2 Federation Process and User Journeys

2.0 Introduction

To illustrate the services offered by the FAME marketplace and platform, this section details the steps users take to complete core actions such as federating, searching for assets, proposing assets for sale, purchasing assets and more. These processes are outlined in terms of **user journeys** and **user flows**.

User journeys and user flows are essential tools in understanding and improving user experiences:

- **Focus on the user:** Both methods prioritize the user's perspective, aiming to identify their needs, pain points, and goals.
- **Design-centric:** They are integral to the design process, aiding in both the initial conceptualization and refinement of products or services.
- **Goal-oriented:** User journeys and flows are structured around specific user objectives, providing a clear path to follow.

By combining these elements, FAME was designed, implemented and integrated in a more intuitive way, with the goal to provide experiences for the users.

2.1 User Journeys

2.1.0 Onboarding of Federation Member (Federation Process)

The Federation process describes the steps that members must follow to be included into the federation:

1. FAME is a regulated Marketplace for regulated access of assets (data assets and technologies)
2. Every producer / consumer must guarantee:
    a. Compliance with EU regulations (e.g GDPR, Data Act [2], Data Governance Act [1], AI Act [13], …)
    b. Ethical behaviour:
        i. Conduct Code
        ii. Adhere to SDG (Sustainable Development Goals)
    c. Legal Obligations (IP protection, Law and regulation compliance…)
3. FAME enables search, publication, of assets for EmFi applications

The FAME marketplace will distinguish different types of users with different roles as illustrated in D2.5 - Requirements Analysis, Specifications and Co-Creation II [6]. For the sake of the following user stories, the following users will be considered.

- Private User (non-affiliated users, individual, citizens, …)
- Entity (Legal entities, Organizations, Companies, Universities, profit & no-profit)
- Marketplace (for M2M interaction)
- EU projects

FAME has the objective to create a community of trusted users, that brings value to the marketplace and has a federation mechanism to manage users in particular business organizations.

The FAME governance onboarding structured approach ensures a systematic process for user and organization federation, covering the entire lifecycle from onboarding to active participation and continuous improvement.

The FAME user federation mechanism is a process with onboarding and consensus criteria that involves several detailed steps. Below is the FAME structured process that encompasses user and organization onboarding, with consensus criteria, and maintaining the federation.

This process happens mostly offline in an application that will be developed once the business and legal framework of FAME will be defined.

By following the following steps, FAME can create an inclusive, efficient, and transparent user federation with well-defined onboarding and consensus criteria.

There are two applications involved:

- FAME FEDERATION GOVERNANCE APPLICATION (**FFGA**)
- FAME MARKETPLACE PLATFORM (**FMAP**)

The FAME Governance Board (**FGB**) is the actual ensemble of all federated identity present at a given time into the FFGA. They are entitled to vote for application and memberships to the FAME marketplace. Once the

2.1.0.0 Step 1: Define Objectives and Scope

Objectives:

- To create a value organization/user federation that share FAME common values
- To ensure smooth onboarding of new users
- To establish clear consensus criteria for decision-making

Scope:

- Applicable to users and organization willing to join the FAME federation.
- Covers the entire lifecycle from onboarding to active participation.

2.1.0.1 Step 2: User Onboarding Process

Pre-Onboarding Preparation

- Documentation: FAME provides onboarding guides, terms of service, privacy policies, and user agreements.
- Tools: FAME provides necessary tools and platforms for user interaction (e.g., forums, communication channels) in FFGA.

User Application Process

- Application Form: FFGA provides an online application form collecting necessary user information.
- Verification: FFGA provides basic validation that, during the production adoption phase, it may also be used in conjunction with offline processes to support the verification of the authenticity of materials produced by users. This is in addition to the checks that each member may carry out during the evaluation of a federation request.

Initial Screening

- Criteria Check: Verify if applicants meet basic criteria (e.g., company legal information, relevant scope for FAME, trustworthy, provenance).
- Background Check: Conduct basic background checks if necessary.

Federation Approval

- Review Process: FFGA sends the information about the organization/user to the FAME Governance Committee to review and approve applications.
- Notification: FFGA notifies users about their application status (accepted, rejected, or need more information).

Orientation

- Welcome Session: FFGA provides a virtual orientation session.
- Resources Access: FFGA provides access to necessary resources, tools, and introductory materials.

2.1.0.2 Step 3: Establish Consensus Criteria

Define Consensus Areas

- Policy: Identify areas requiring consensus (e.g., rule changes, major decisions).
- Scope: Define the scope of decisions needing consensus (strategic, operational).

Develop Consensus Mechanism

- Voting System: The FAME governance board (FGB) uses a voting system based on majority.
- Quorum: The FAME governance board (FGB) uses a quorum requirements for decision-making.

Consensus Building Process

- Proposal Submission: FAME project will outline how proposals are submitted (e.g., proposal templates, submission guidelines).
- Discussion Period: FAME project will define a discussion period before voting (e.g., forums, meetings).
- Amendment Process: Allow FAME project will consider for amendments to proposals during discussions.

Decision Implementation

- Vote Execution: FFGA uses an electronic voting, ballots.
- Result Announcement: FFGA Announce results transparently to all members.
- Implementation Plan: FFGA implements a plan to take decisions effectively.

2.1.0.3 Step 4: Continuous Engagement and Feedback

Regular Meetings

- Schedule: FAME Governance Board has regular meetings for updates and discussions.
- Agenda: FAME Governance Board prepares and shares agendas in advance.

Feedback Mechanism

- Surveys: FGB Conduct regular surveys to gather user feedback.
- Suggestion Box: FGB Create a suggestion box for continuous input.

Training and Development

- Workshops: FGB organizes training sessions and workshops.

- Mentorship: FGB implements mentorship programs for new members.

2.1.0.4 Step 5: Evaluation and Improvement

Performance Metrics

- KPIs: FGB defines key performance indicators (e.g., user participation rates, decision implementation time).
- Regular Review: FGB conducts regular reviews of KPIs.

Process Improvement

- Feedback Analysis: Analyze feedback for recurring issues.
- Adjustments: Make necessary adjustments to onboarding and consensus processes.

Reporting

- Transparency: Maintain transparency by sharing regular reports with all members.
- Documentation: Keep detailed records of all processes and changes.

2.1.1 Casual Navigation

User Role: Generic / Gender: Unknown / Name: Anonymous
Step-By-Step:

1. An anonymous user navigates with his/her web browser to the landing page
2. He/She browses through a series of pages with: Presentation, Search, FAQ, Other links

2.1.2 Organization Federation

User Role: Producer or consumer organization / Gender: female / Name: ACME
Step-By-Step:

1. A Business Organization (ACME) wants to federate to FAME
2. Alice on behalf of the organization lands with the browser on the FAME Platform or directly on the FAME federation governance application FFGA.
3. Alice is redirected to a new application that manages the process of FAME federation FFGA.
4. The FFGA presents the relevant information (what is FAME, what it means to federate, what is required…)
5. Alice moves to the application form and is forced to read all documentation (Terms of Service, Legal notes, etc.)
6. Alice provides in the form the company details, proves she operates as the ACME Legal Signatory and enters her contact information (company email alice@acme.com. If the ACME organization is Onboarding Authority, she must enter a decentralized identifier (DID) otherwise she can choose to delegate another organization to act as Onboarding Authority
7. FFGA organizes the information related to Alice and ACME and notifies the FGB
8. FGB have some time to inspect the application and are required to express their vote in the FFGA
9. FFGA at the time of expire of ACME/Alice application sums the votes and communicates to Alice the results
10. The FFGA communicates to the FAME Platform **FMAP** to register ACME as a trusted organization.
11. By accessing the FFGA control panel, Alice can individually invite users who will be able to access the marketplace
12. The fmap sends users an OTP code and a link from which to generate a qrcode
13. Through a FAME-compatible SSI Wallet app and by framing the qrcode just generated, the credentials to access the marketplace will be generated

This general process is simplified for EU private citizens and Institutions of EU or EU project in the step 6,7,8,9,10.

An external marketplace federates with the same process of an external business organization.

2.1.3 User Technical Onboarding

User Role: Generic / Gender: Female / Name: Alice
Step-By-Step:

- Alice, as representative of ACME organization which is a trusted member of the FAME business ecosystem, has identity managed by FFGA and is recognized by ACME/Alice DID
- FMAP has received from FFGA the ACME/Alice DID
- Alice installs a FAME-compatible Identity Wallet application on her personal device
- Alice can now operate on the **FMAP**
- **FMAP** admin onboards the received DID. (**FMAP** Admin does not know anything about Alice.)
- Now Alice can access the **FMAP**

2.1.4 Asset Preparation by Producer Application

User Role: Machine / Gender: Neutral / Name: Alice_Application
Step-By-Step:

1. Alice prepare the asset with APIs, tokens, etc… to enable access to the asset
2. Alice can develop an Alice_Application with FMAP API check. In this case at run time, for every use a check of availability and a record of usage is written into FMAP.
3. Alice provides the asset credentials for access (API Links, tokens, Connector, etc) to FMAP (the publishing)

2.1.5 Asset Publishing

User Role: Producer / Gender: Female / Name: Alice
Step -By Step:

1. Alice navigates with her web browser to the Asset Publishing Page
2. Alice fills in the form on the page with the asset's metadata, and submits it
3. Alice receives the ID assigned to their pending request and the input of a blockchain transaction
4. Alice, by means of her MetaMask wallet (browser plugin), signs the transaction with her trading account's private key and submits it to the P&T Tracing smart contract
5. Alice checks the request's status to get confirmation that it was processed successfully
6. Alice provides the requested metadata
7. Alice configures the access policy of the asset

2.1.6 Asset Publishing from other Marketplace (Application to FAMP - M2M)

User Role: Marketplace /Gender: Neutral / Name: Ext_MarketPlace
Step-By-Step:

1. Ext_MarketPlace use FMAP API /Batch (using supplied DID) to
    a. submit or update the asset metadata on Fame catalogue
    b. provides the requested metadata

2.1.7 Define Offering

User Role: Producer / Gender: Female / Name: Alice
Step-By-Step:

1. Alice navigates the asset catalogue on the FAME Marketplace, loads the page that displays the details of a previously published asset, and opts for adding a new commercial offering for that asset: the Offering definition web page is loaded into the user's browser
2. *Optionally*, Alice asks for advice on pricing and receives a list of questions that is displayed in a dialog window; the user then submits their answers to the questions
3. Alice receives a price suggestion that is displayed on the Offering definition page
4. Alice fills in the form on the Offering definition page, and submits it
5. Alice provides the Access Credential for sale of the asset in the form of offering
6. Alice receives the ID assigned to their pending request
7. Alice checks the request's status to get confirmation that it was processed successfully

2.1.8 Asset Discovery for Trading

User Role: Casual / Gender: Male / Name: Bob
Step-By-Step:

1. Bob is navigating to the 'home page' to locate assets of his interest
2. Bob chooses to search for an asset through the 'home page' (Alternatively, the user can search for an asset through the 'search page')
3. Bob is not able to view the existing assets without being onboarded
    a. Bob federates (Federation Process) is registered and logs-in
    b. Bob is not registered and proceeds to federate (FFGA followed by onboarding)
4. Bob finds an already indexed asset (quality assurance analytic)
5. The asset has already indexed to the FDAC
6. The asset has two (2) different offerings that have been added by producers

2.1.9 Asset Purchase

User Role: Casual / Gender: Male / Name: Bob
Step-By-Step:

1. Bob is federated and onboarded and access the FAME Marketplace with is ID
2. Bob selects a specific offering of the chosen asset
3. Bob completes the check-out
4. A separated process with Financial Transaction (Token, Credit Cards, etc.) shows up (*)
5. FAME gets transaction fee for purchase
6. Bob receives the credentials (NFT) of the asset purchased (containing the token to access the asset e.g. API Links, tokens, connector, etc) provided by Alice

(*): This operation and Further Billing, Transactions, Data flows happens outside FAME Platform

2.1.10 Asset Use by Consumer Application

User Role: Machine / Gender: Neutral / Name: Bob_Application /
Step-By-Step:

1. Bob _Application is implement to:
    a. Access the provider asset (API Links, tokens, Connector, etc.)
2. Bob_Application use the provided production credential to access the asset
3. Bob_Application use the asset (API Links, tokens, Connector, etc.) for example:
    a. Download DataSet (API Links, Connector, etc.)
    b. Download software (API Links, Connector, etc.)
    c. Execute provider API (API Links, Connector, etc.)

NB: All this operation happens outside FAME Platform

2.1.11 User Offboarding

Upon violation of conditions (Federation consensus agreement, Asset TOS) a user (any role) can be offboarded.

Ste--By Step:

1. This is initiated by FGB, upon user violation of TOS.
2. User is marked as non-fame and frozen
3. User log-in is prevented in (FFGA and FMAP)

3 FAME Frontend and Backend Integration

3.0 Introduction

To provide an overview of FAME, a conceptual view of the overall platform has been developed to highlight the benefits for both data consumers and data producers of the platform (Figure 1).

In both scenarios, as soon as data consumers and data producers are officially onboarded through their organizations to the FAME federation (following the process described in Section 3.2), they are then able to be registered as individual users to the platform, and authenticate themselves to access all the FAME's features. To this context, data consumers are able to search for and select data assets existing within the Federated Data Assets Catalogue (FDAC) of the platform, whereas data providers, like Data Space, Data Marketplaces and dataset owners, are able to connect to FAME to index their assets, allowing data consumers to further utilize them. This dual approach reflects FAME's bidirectional nature, fostering interaction between data providers and data consumers for a mutually beneficial ecosystem. Additional details on the constructed conceptual view can be located in the deliverable "D2.6 - Technical Specifications and Platform Architecture II".

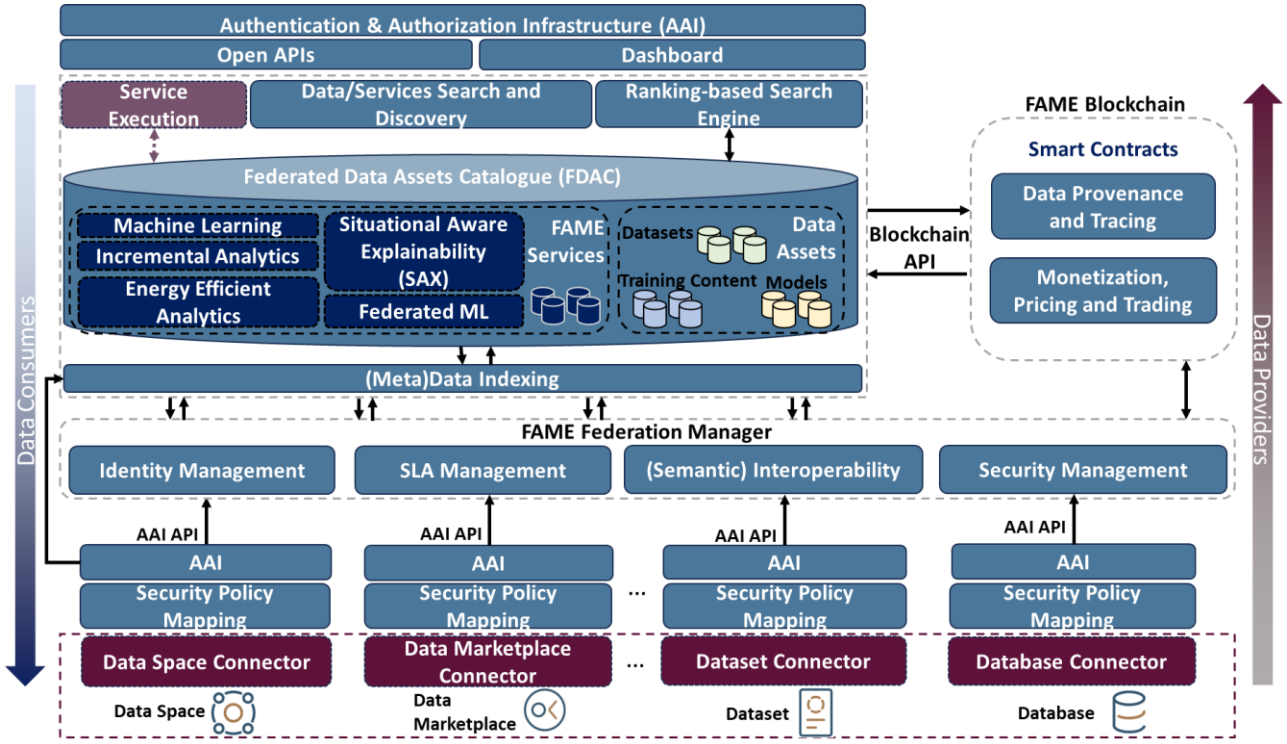


Figure 1 : Conceptual view of FAME

To further evolve this concept, this conceptual view has been used as the basis for the construction of the FAME Solution Architecture (SA) (Figure 2), containing specifically the following layers: (i) Dashboard, (ii) Open APIs, (iii) Federation Manager, (iv) Transactional Operations, and (v) Energy Efficient Analytics Services.

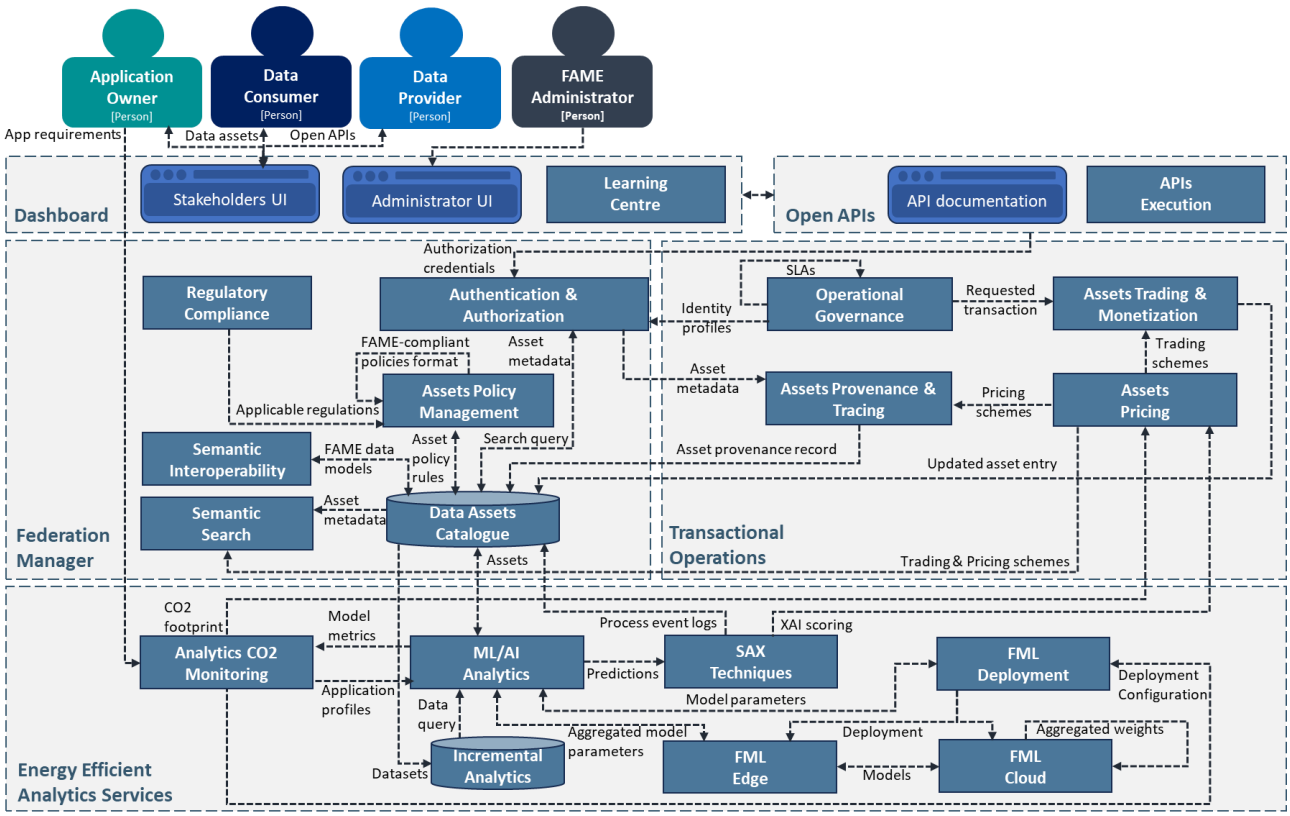


Figure 2 : FAME C4 architecture

Dashboard

The Dashboard layer contains all the components and User Interfaces (UIs) that are required for providing access to all the FAME stakeholders (i.e., end-users (data consumers, data providers) and administrators) regarding the supported functionalities of the FAME Federated Data Space.

Open APIs layer

The Open APIs layer is closely related to the Dashboard layer, acting as an additional interacting point with the FAME Federated Data Space. In essence it provides the interconnection and smooth integration between all the UIs and all the underlying FAME functionalities.

Federation Manager

The Federation Manager layer contains all the components that are needed for realizing the federated functionalities of FAME, ranging from the support of registration, authentication, and authorization of federated users and data sources, to the sharing and indexing of data assets deriving from such federated parties (i.e., external users and data sources).

Transactional Operations

The Transactional Operations layer contains all the components that are mostly related to the daily operation of a Data Space that supports trading – i.e., the online process through which the provider and the consumer of a given data asset can stipulate a legally-binding agreement that determines the terms and conditions of use, and possibly includes the contextual execution of a payment by means of a digital currency.

Energy Efficient Analytics Services

The Energy Efficient Analytics Services layer brings to FAME the necessary tools for implementing analytical functionalities for EmFi applications. This allows end-users to apply AI/ML models to their own data (or data coming from an external source), to obtain desired outputs (i.e., forecasting values, ranking systems, sentiment analysis), and to receive an explanation of the given results, thus adding business value to client's companies onboarded to the FAME Federated Data Space.

The FAME SA has been designed using the concept of a decoupled approach, separating the frontend from the backend that is a cornerstone of modern development, promoting flexibility, maintainability, and efficient development workflows. The decoupled approach is a way of separating the frontend (what users see and interact with or external applications interact with as already explained using a microservices architecture) from the backend (the data and logic that powers the application).

The advantages of the decoupled approach are:

- **Independent Development**: Frontend and backend teams can work more independently. Frontend developers can choose the most suitable technologies for the UIs' framework, while backend developers can focus on business logic without worrying about the specific UIs' implementation. This allows for parallel development and faster release cycles.
- **Flexibility and Scalability**: Each tier, frontend and backend, can be scaled independently based on its needs. This is especially useful for applications that experience traffic spikes. Resources can be allocated efficiently by scaling up the backend during high traffic periods and the frontend to handle increased user interactions.
- **Improved Maintainability**: With a clear separation of concerns, the codebase becomes easier to understand and maintain. It is simpler to identify issues and implement new features because the frontend and backend logic are not intertwined.
- **Teamwork and Communication**: Decoupled development fosters better communication and collaboration between frontend and backend teams. Since the codebases are separate, teams can work more autonomously while still needing to collaborate on Application Programming Interfaces (APIs) that define how the frontend and backend communicate.

3.1 FAME Federation Application

The FAME Federation Governance Application (FFGA) serves as the dedicated platform for managing the federation process within the FAME ecosystem. As outlined in Chapter 2, the federation process is a critical component that ensures only trusted and compliant organizations can participate in the FAME marketplace. The FFGA provides a structured, user-friendly interface for organizations to apply for federation membership while maintaining the highest standards of governance and security.

The FFGA operates as a complementary application to the main FAME Marketplace Platform (FMAP), specifically designed to handle the pre-marketplace activities related to organizational onboarding and federation management. This separation ensures that the federation governance processes remain independent and transparent while maintaining seamless integration with the broader FAME ecosystem.

3.1.0 FFGA design process

The design process for the FFGA followed a methodology similar to that employed for the FAME Dashboard, with particular emphasis on governance transparency and regulatory compliance. The design phase encompassed several key activities:

Requirements Collection: The FFGA requirements were derived from the federation process specifications detailed in Chapter 2, focusing on the need to manage organizational applications, voting mechanisms, and credential management. Key requirements included:

- Support for multi-step application processes with document verification
- Support for onboarding-authority selection: an applicant that is itself an Onboarding Authority (OA) must provide its Decentralized Identifier (DID) during federation; an applicant that is not an OA must designate an existing federated organization (FAME is the default, but any other OA member may be chosen) to act as OA on its behalf
- Implementation of democratic voting systems for federation members
- Integration capabilities with external authentication systems
- Compliance with GDPR and EU regulatory frameworks

User Journey Mapping: The design process centered around the primary user journey of organizational federation, as described in Section 2.2.3. This journey encompasses the complete lifecycle from initial application submission through voting and final approval/rejection.

Interface Design: The FFGA interface was designed with emphasis on:

- Clear presentation of federation rules and requirements
- Intuitive document upload and verification workflows
- Transparent voting status and results display
- Responsive design supporting various device types

Governance Framework Integration: The design incorporated the governance principles established by the FAME Governance Board (FGB), ensuring that all interface elements support democratic decision-making and transparent communication.

3.1.1 FFGA development process

3.1.1.0 FFGA technologies

The FFGA is built using modern web technologies optimized for reliability and security:

Backend Framework:

- **Flask** (Python 3.13): Chosen for its simplicity, flexibility, and strong security features
- **SQLAlchemy**: For robust database operations and ORM functionality
- **APScheduler**: For managing time-based voting processes and automated workflows

Database:

- **SQLite**: Lightweight database solution suitable for federation management workloads
- Support for future migration to PostgreSQL for enhanced scalability

Frontend Technologies:

- **HTML5/CSS3**: Modern web standards implementation
- **Tailwind CSS**: Utility-first CSS framework for responsive design
- **JavaScript**: For dynamic user interactions and form validation

Security and Validation:

- **Phonenumbers library**: For international phone number validation
- **Custom validation modules**: For document verification and data integrity
- **GDPR compliance modules**: Ensuring data protection standards

Deployment and CI/CD:

- **Docker**: Containerization for consistent deployment
- **Poetry**: Python dependency management
- **GitLab CI/CD**: Automated testing and deployment pipelines

3.1.1.1 FFGA source code

The FFGA follows a modular architecture with clear separation of concerns:

Project Structure:

```
ffga/
├── app/
│   ├── models/        # Data models (Organization, User, Vote, etc.)
│   ├── routes/        # URL routing and view logic
│   ├── services/      # Business logic and external integrations
│   ├── templates/     # HTML templates
│   ├── static/        # Static assets (CSS, images, documents)
│   └── utils/         # Utility functions and decorators
├── data/              # Database and uploads storage
├── logs/              # Application logging
```

```
└── configuration files
```

Key Components:

Data Models: The application implements a comprehensive data model supporting the federation workflow:

- **User**: Represents federation applicants and existing members
- **Organization**: Stores organizational details and federation status
- **Vote**: Manages individual voting records
- **Document**: Handles uploaded verification documents

Service Layer: Business logic is encapsulated in dedicated services:

- **AuthService**: User authentication and session management
- **OrganizationService**: Organization management and validation
- **VoteService**: Voting process coordination
- **EmailService**: Notification and communication systems

Security Features: The application implements multiple security layers:

- Password strength requirements and validation
- Document upload verification and sanitization
- Session management and CSRF protection
- Privacy-compliant data handling

The FFGA provides an intuitive interface for both applicant organizations and existing federation members:

3.1.1.2 FFGA user manual

Introduction

The FAME Federation Governance Application (FFGA) provides a unified web-based environment for organizations to apply for federation membership within the FAME Data Marketplace ecosystem. It is designed with transparency, security, and democratic governance in mind. This manual outlines the key elements of the federation interface, providing representative screenshots of the current version of the application, and offers an overview of the main functionalities available to users.

The FFGA serves as the dedicated platform for managing the federation process within the FAME ecosystem, ensuring that only trusted and compliant organizations can participate in the FAME marketplace. The application provides a structured, user-friendly interface for organizations to apply for federation membership while maintaining the highest standards of governance and security.

Home Page

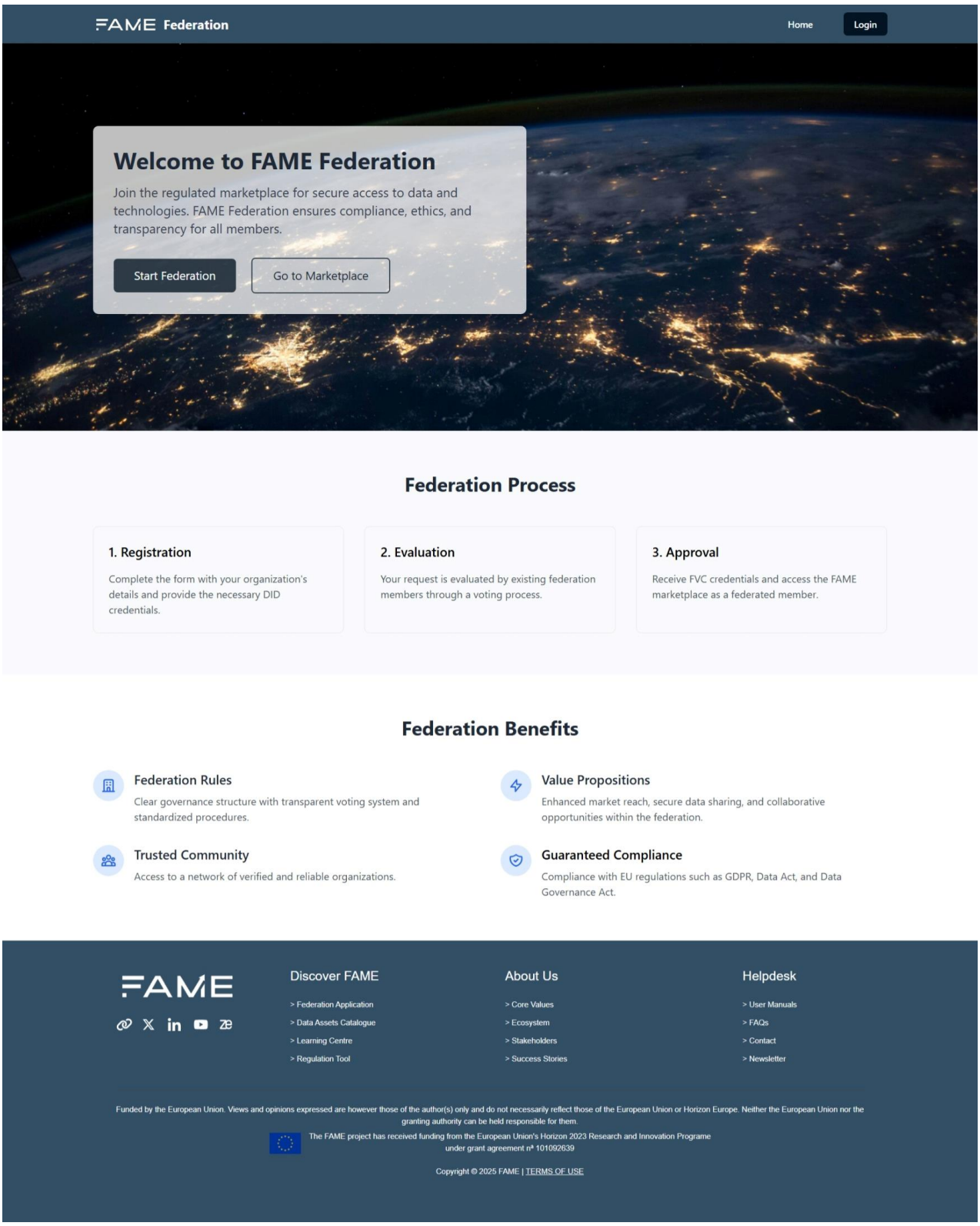


Figure 3 : Home page

The Home page of the FAME Federation Governance Application serves as the main entry point for both prospective federation members and existing members. The page is accessible by everyone and offers a comprehensive overview of the federation ecosystem and its purpose.

At the top, the header introduces the FAME federation identity through a clean navigation menu providing access to key functionalities, including user login, federation information, and navigation

to the Home. The navigation maintains consistency with the broader FAME ecosystem design while emphasizing the governance-focused nature of the application.

The hero section begins with a semi-transparent content area presents the main value proposition: "Join the regulated marketplace for secure access to data and technologies. FAME Federation ensures compliance, ethics, and transparency for all members." For non-authenticated users, a prominent "Start Federation" button provides direct access to the application process, while a "Go to Marketplace" button links to the main FAME marketplace platform.

Federation Process Section

Below the hero area, the "Federation Process" section outlines the three-step journey for organizations seeking federation membership:

1. **Registration**: Complete the form with your organization's details and provide the necessary DID credentials
2. **Evaluation**: Your request is evaluated by existing federation members through a voting process
3. **Approval**: Receive access the FAME marketplace as a federated member

This section provides transparency about the federation workflow, helping potential applicants understand what to expect throughout the process.

Federation Benefits Section

The "Federation Benefits" section highlights four key advantages of joining the FAME federation:

1. **Federation Rules**: Links to the FAME marketplace to provide detailed information about the clear governance structure with transparent voting system and standardized procedures
2. **Value Propositions**: Directs users to the marketplace's value proposition page, outlining the strategic benefits of federation membership
3. **Trusted Community**: Internal link to the trusted community page where users can view all current federated members and understand the quality of organizations within the federation
4. **EU Regulations Compliance**: Emphasizes the federation's commitment to compliance with European Union regulations, ensuring legal and ethical data sharing practices

Trusted Community Page

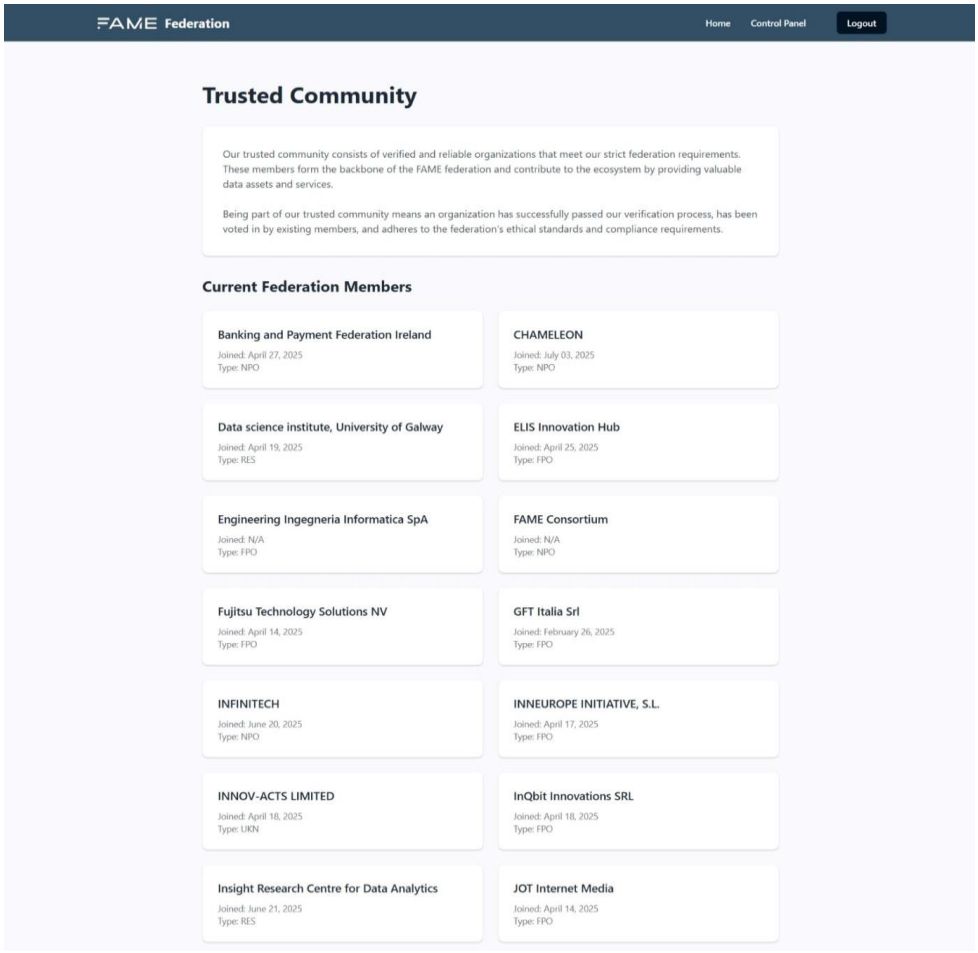


Figure 4: Trusted Community page

The Trusted Community page showcases all approved federation members, providing transparency about the organizations that comprise the FAME federation. This page is accessible to everyone and serves as a public directory of trusted partners within the ecosystem.

The page displays federated organizations in an organized, searchable format:

- Organization legal names
- Federation approval dates showing membership durationOrganization types and primary business sectors

This transparency feature builds trust in the federation by allowing stakeholders to see exactly which organizations have been approved through the democratic voting process. It also helps potential applicants understand the types of organizations that are typically accepted into the federation. The page updates in real-time as new organizations are approved, ensuring that the community directory always reflects the current state of the federation membership.

Footer Navigation

The footer provides comprehensive navigation to additional resources and information, including links to user manuals, FAQs, contact information, newsletter subscription, core values, ecosystem information, stakeholders details, and success stories. The footer also displays the European Union funding acknowledgment and includes links to terms of use, reflecting the project's commitment to transparency and regulatory compliance.

Federation Application Page

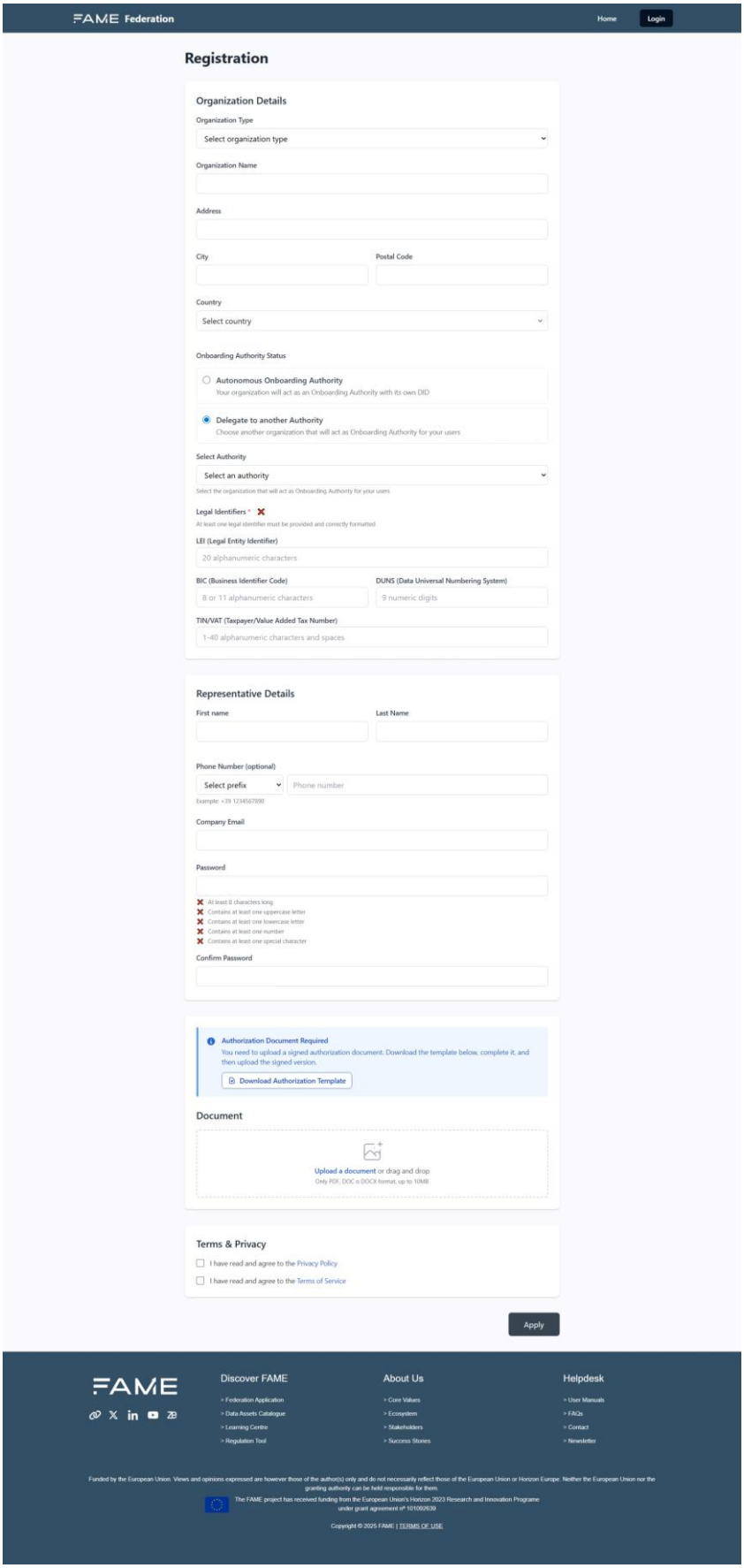


Figure 5: Federation page

The Federation Application page is the core interface where organizations submit their requests to join the FAME federation. This page is accessible to non-authenticated users and provides a comprehensive multi-step form for collecting all necessary organizational and representative information.

The application form is structured in logical sections to guide applicants through the federation process:

Organization Information Section

This section collects fundamental details about the organization including:

- Legal organization name
- Legal entity identifier and registration country
- Complete postal address details with automated country selection
- Organization type classification (corporation, partnership, non-profit, etc.)

Onboarding Authority Selection

A critical component of the federation process is the selection of an Onboarding Authority (OA):

- Organizations can select FAME as their default onboarding authority
- Existing federated organizations may serve as onboarding authorities for new applicants
- Self-onboarding is available for organizations that are themselves onboarding authorities

This section includes explanations of each option and its implications for the user management process post-federation.

The form implements real-time validation to ensure data quality and completeness, with clear error messages guiding users to correct any issues before proceeding.

Authorized Representative Information

The second major section focuses on the individual who will serve as the organization's primary contact and representative within the federation:

- Full name
- Direct contact information including company email and phone number
- Password creation for future access to the federation management system

Phone number validation includes international format checking to ensure communication reliability across global federation members.

Document Upload and Verification

The document section handles the secure upload of required authorization documents with a prominent information panel titled **"Authorization Document Required"**. This blue-bordered information box explains that applicants need to upload a signed authorization document and provides a direct link to download the template document.

The template must be completed with organizational details, signed by the representative user (the authorized representative of the organization), and uploaded in the secure file upload box below.

The interface features:

- Support for PDF document uploads with file size and format validation
- Drag-and-drop upload functionality with visual file selection interface
- Clear guidelines requiring official signatures and organizational authorization
- Secure file handling with virus scanning and document verification
- Document privacy protection in accordance with GDPR requirements

Terms and Privacy Acceptance

Before submitting the application, users must review and accept the platform's legal agreements through a dedicated "Terms & Privacy" section. This section requires two mandatory checkbox confirmations:

- **"I have read and agree to the Privacy Policy"** - Links to the comprehensive Privacy Policy page which outlines how FFGA handles personal and organizational data in compliance with EU regulations, including data collection practices, processing purposes, security measures, user rights under GDPR, and contact information for privacy-related inquiries.
- **"I have read and agree to the Terms of Service"** - Links to the Terms of Service page which establishes the legal framework for using the FFGA platform, covering the federation process obligations, user responsibilities, intellectual property rights, platform governance rules, and compliance requirements for EU regulations.

Both documents open in new tabs to allow users to review the complete terms while maintaining their application progress. Upon form completion, applicants receive immediate confirmation of their submission along with a unique request identifier for tracking purposes.

Login Page

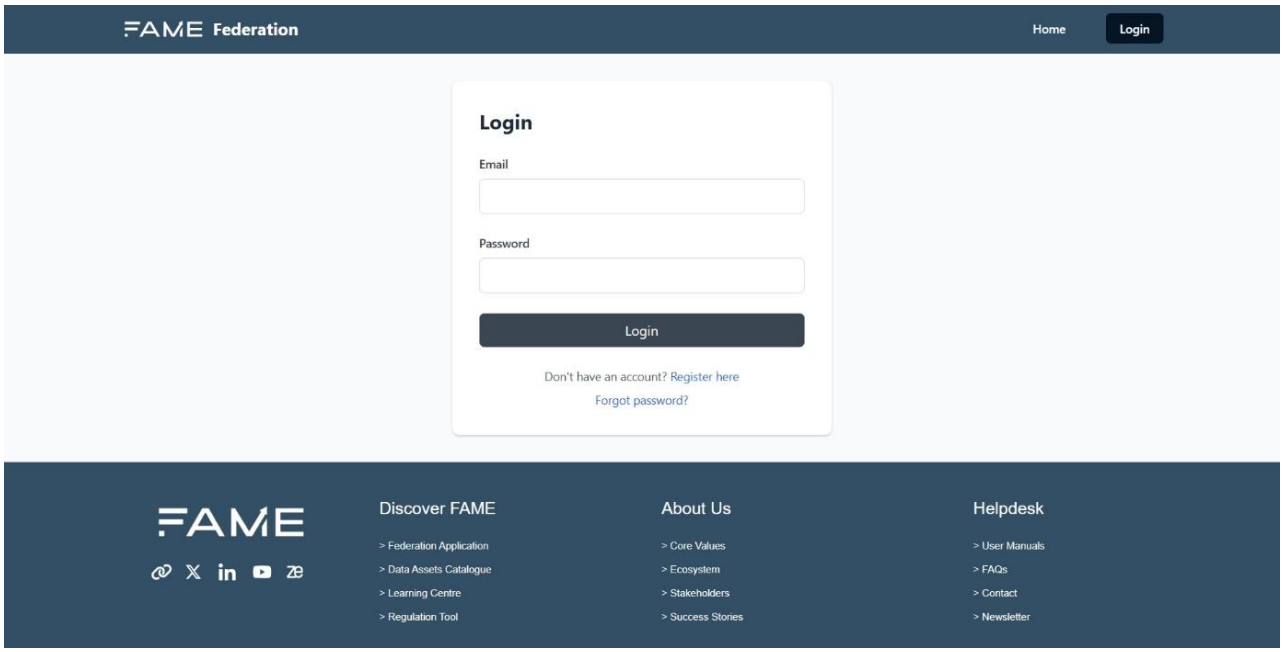


Figure 6: Login page

The Login page provides secure access for existing federation members and organization representatives to access their dashboard and participate in federation governance activities. The page implements modern authentication standards while maintaining ease of use.

The login interface features:

- Email-based authentication with case-insensitive handling
- Secure password input with strength requirements clearly displayed
- Session management with appropriate timeout policies
- Password recovery options for users who have forgotten their credentials

The page also provides clear guidance for new users on how to access the federation application process and understand the difference between public information access and authenticated member features.

For users who have not yet confirmed their email addresses, the page provides options to resend confirmation emails and links to support resources for resolving access issues.

Control Panel

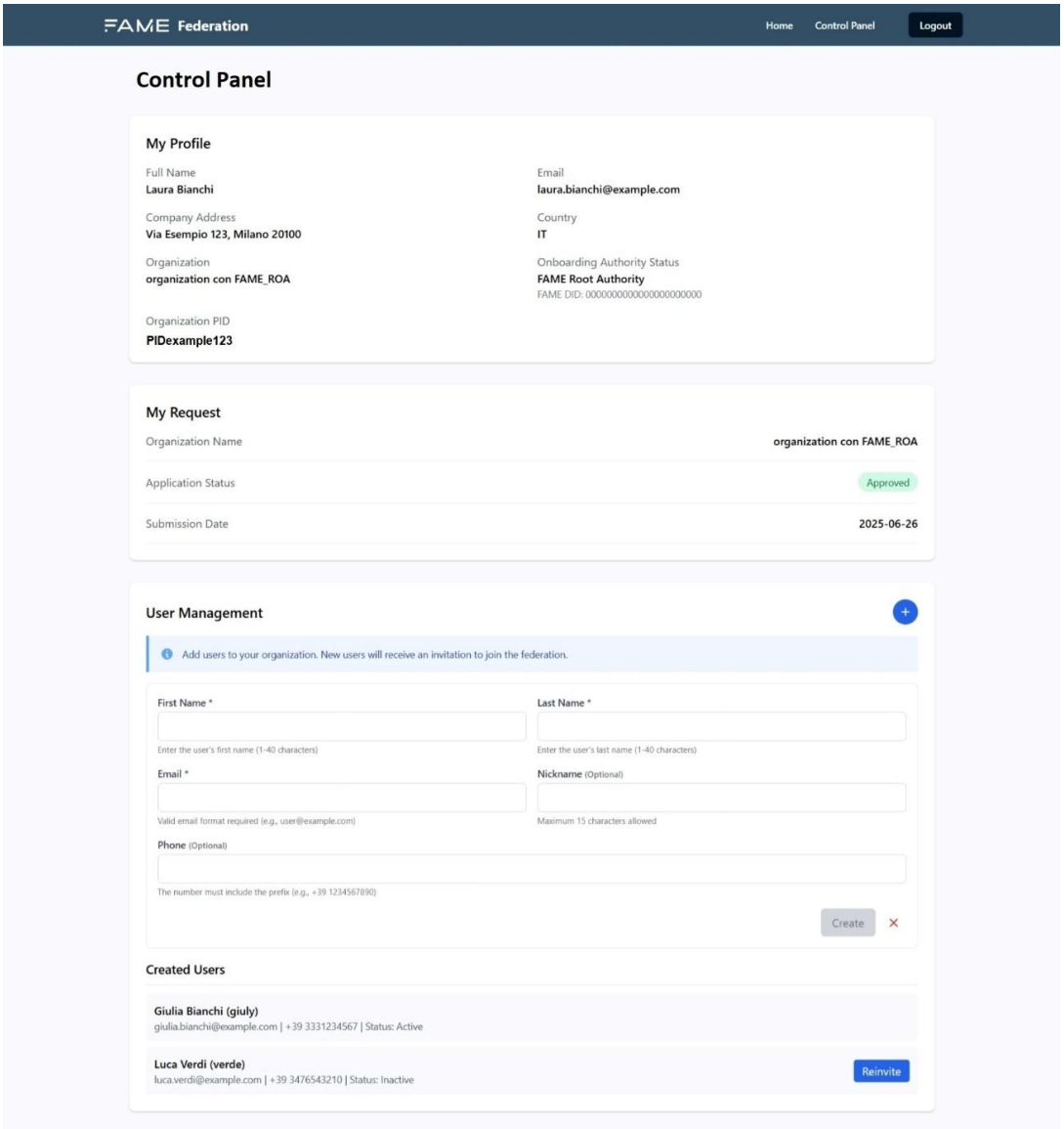


Figure 7: Control panel page

This figure shows the control panel page of the FAME Federation portal, a dedicated space where registered users manage their organization's presence and administrative activities within the FAME ecosystem. This view is customized and available only to authenticated users, serving as a central management hub.

Immediately below, the page is organized into modular blocks of information, starting with **My Profile**, which summarizes the logged-in user's credentials and organizational affiliation. This section clearly displays the user's full name, email address, company address, country, and organization name. It is important to note that it also highlights the authority type within the onboarding process, indicating their onboarding authority status, chosen during the federation phase (see the previous section "Federation Application Page"), thus granting them elevated permissions within the ecosystem. In particular, only users who have chosen FAME root authority will be able to manage marketplace users on this page in the User Management section. In addition, the user is also assigned a unique organization PID, ensuring traceability and secure identity management throughout the Fame ecosystem.

Next, the **My Request** module concisely communicates the status of the user's application for inclusion of the organization in the federation. The panel shows the name of the organization and the date it was approved. This status with date and time ensures transparency for administrative monitoring and is essential for audit and compliance functions.

The **User Management** section, available only to those who have chosen Fame as their onboarding authority, further emphasizes the administrative capabilities of the logged-in user, allowing them to add or manage users who can access the marketplace. An information banner reminds administrators that new users must receive invitations to join the federation and access the marketplace. Using the "+" button, you can invite new users by entering the following information: first name, last name, and email address, as well as optional nickname and phone number. Invited users will then receive an email with instructions on how to activate their account and access the marketplace. Below is a list of user accounts that have already been created, including their status (active or inactive) along with their associated email addresses. Action buttons such as "Reinvite" offer dynamic options for re-engaging users, helping to simplify onboarding workflows and organizational maintenance. Buttons for editing, offboarding and deleting users are also planned in the future.

The **Federation Requests** section concludes the main body of content, clearly indicating new federation requests. You can click on the request to view all the details.

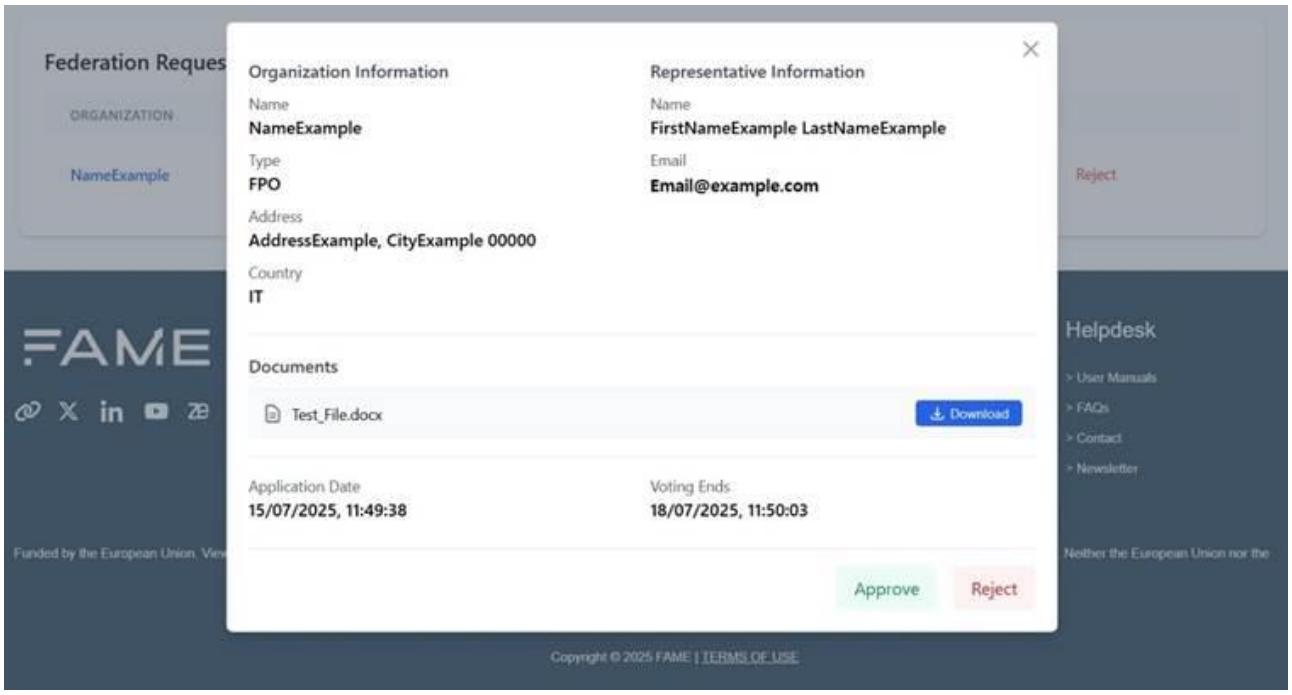


Figure 8: Request details

All the details of the organization and the legal representative who submitted the request are displayed. In addition, you can download and view the Authorization Document. The date of submission of the request and the closing date of the vote are shown. Finally, there are buttons to cast your vote, accept or reject the request.

3.2 FAME Dashboard

Within its complete implemented environment, FAME provides an integrated user-friendly end-to-end User Interface (UI) facilitating the interaction of the FAME stakeholders with all the involved FAME backend services, components, and processes. Such UI, namely the FAME Dashboard, has been already modelled and depicted within the overall FAME SA as described in D2.6 - Technical Specifications and Platform Architecture II [5], and is also illustrated in the figure below. As it can be observed, various types of stakeholders can have access to the FAME Dashboard (i.e., FAME Administrator, Data Provider, Data Consumer), where in the cases that an external stakeholder may be an end-user representing either a data consumer or a data provider intending to connect and interact with FAME, he/she has access to the Stakeholders UI. Otherwise, in the cases that the external stakeholder is a FAME administrator needing to access FAME for administration purposes or for any system parameterization/maintenance, he/she is navigated through the Administrator UI that offers a related UI for such purposes.

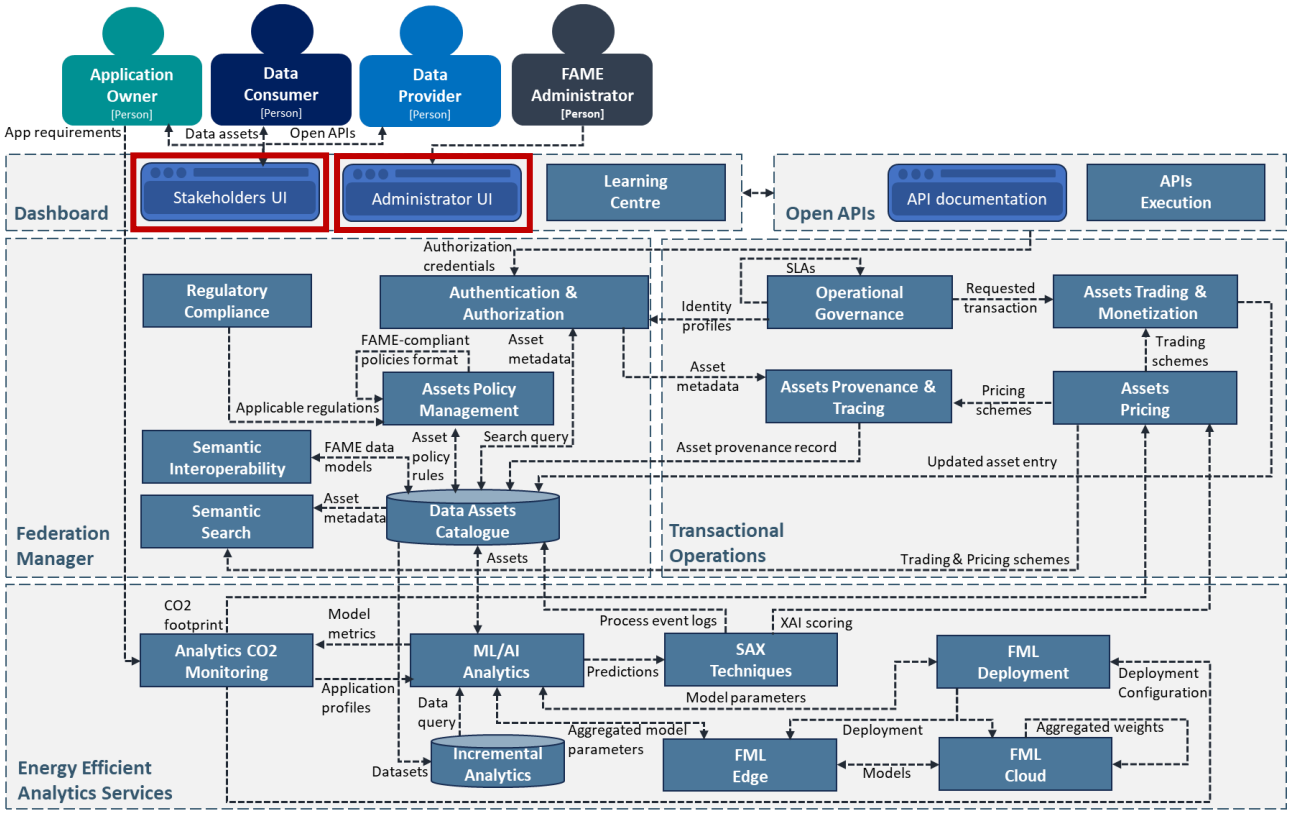


Figure 9 : Positioning of FAME Dashboard within FAME SA

Hence, the main reason behind the FAME Dashboard is to provide to all of its stakeholders (both technical and non-technical ones), a user-friendly end-to-end UI that will facilitate their interaction with all the involved FAME services, components, and processes. Towards this direction, the FAME Dashboard has been designed and implemented for providing an interface that will be aesthetically pleasing and simple to use, inviting all the FAME stakeholders (both end-users and administrators) to further explore all the associated capabilities and components of the overall platform. As a result,

User Experience (UX) has been enhanced with an emphasis on streamlining complicated FAME-related activities (such as searching, monetization, and trading), guaranteeing fluid navigation, and ensuring that end-users can perform their tasks with ease. Thus, a strong brand identity will be built, helping towards transforming FAME to a widely known single-entry point of data assets and functionalities related with EmFi applications.

3.2.0 Dashboard design process

Prior to initiating the overall design process of the FAME Dashboard, a specific methodology was followed to fulfill the different FAME platform requirements and facilitate the final integration. Such methodology is traditionally followed when designing applications, including a variety of steps for capturing the different needs and points of views. This methodology considers both the UI and the UX aspects, considered in general critical when designing digital products. While they are often used interchangeably, they have distinct roles and responsibilities. The key difference between UI and UX is that the UI considers the visual and interactive elements of a digital product that facilitate user interaction, whereas the UX deals with the overall experience that users have while interacting with a digital product, including their emotions, perceptions, and responses. Considering the latter, the steps followed for the FAME Dashboard design process are depicted below:

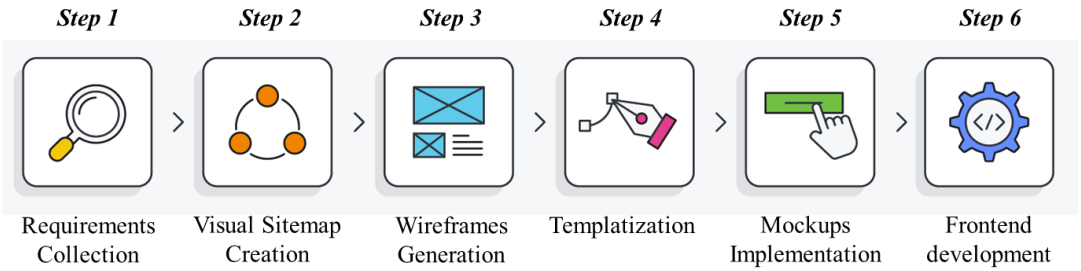


Figure 10 : FAME Dashboard design process

3.2.0.0 Step 1: Requirements Collection

To make the FAME Dashboard interface more engaging to the end-users, it has been considered that the UI design should be a top priority for the end-users. That is the reason why deep research was followed on the potential FAME end-users by the different designer and development teams, focusing primarily on the set requirements. Since the beginning of FAME, a plethora of Business and Technical requirements have been identified for capturing the different needs of the FAME pilots, being available in D2.5 - Requirements Analysis, Specifications and Co-Creation II [6]. These requirements have guided and facilitated the implementation process of the FAME components, whereas also boosting the design, specification, and implementation of the FAME Dashboard. Apart from the consideration of all these requirements, a questionnaire has been circulated (see the figures below) to the different FAME technical components' leaders, to better identify their needs in terms of UI design requirements, facilitations, or potential adaptations (e.g., requirement for a specific number of input and output fields or specific actions' triggering buttons). The questionnaire included several questions covering the topics of the: (i) personal information of the different technical components' leader, (ii) name/description of the technical components, (iii) requirements of the UI, (iv) details of the interacting technical components, (v) way of interaction of the technical components, (vi) totally required input/output UI fields, (vii) preferred way of communication with the FAME Dashboard, and (viii) technologies/programming languages that are used for the technical components. At this point, it should be mentioned that for the updated version of the current deliverable, the same questionnaire has been circulated again to gather a potentially new set of requirements from the partners. No major updates were provided to these questionnaires, apart from the need for improved

UX and more efficient end-user navigation, thus adjusting the needed effort for restructuring specific parts of the overall UI and UX. In particular, towards re-assessing the ambition and further increase usability, accessibility, performance, and overall aesthetics of the FAME Dashboard's, and overall the FAME Data Marketplace's UI and UX, a list of specific actions was applied:

Enhancing Navigation & Information Architecture by redesigning the navigation experience and rebuilding the search functionality that acts as the core of the FAME Data Marketplace. In this context, the number of clicks required to reach important content was reduced, offering in just a few clicks the required content to the end-users.

Optimizing Performance by optimizing images and using lazy loading in pages related to providing content, without needing time-consuming back-end functionalities, following the related best practices that have been already identified in a related conference publication of the FAME consortium. Minification of front-end languages (e.g., HTML, SCSS, TypeScript) was another measure that was considered for enhancing UX.

Enhancing Accessibility by providing the ability to end-users with disabilities to efficiently navigate within all the UIs of the FAME Data Marketplace, ensuring that forms and controls are easily navigable for all the stakeholders.

Improving UI Design & Aesthetics by using consistent system design in all the UIs involved, ensuring that all of them follow the same colors' patterns, typography, and spacing. The UX was enhanced by also ensuring responsive design for mobile-first experiences, facilitating wider usage of the FAME Data Marketplace from a plethora of devices and screen sizes.

Streamlining User Workflows by minimizing form fields and using autofill where possible, even showcasing to the end-users (e.g., videos, hover effects) of what and how should be written in specific forms. Another option was to provide progress indicators in multi-step processes, especially for the indexing of data assets and their related offerings that may be usually a long process.

Implementing Clear Feedback Mechanisms by providing real-time validation on forms and using success/error messages that are both helpful and clear (e.g., when searching the Federated Data Assets Catalogue).

Personalization & Customization by offering font adjustments and related UI preferences, also available through the accessibility feature.

Improving Security & Trust by enhancing and updating the terms of use of the FAME Data Marketplace, clearly communicating its privacy policies and data usage.

Maintaining Consistency & Updates by regularly updating the FAME Data Marketplace following user feedback, ensuring that all the updates are firstly tested and experimented in an offline version of the platform, thus avoiding breaking the user flow.

The figures below illustrate some indicative responses from the overall components' leader feedback.

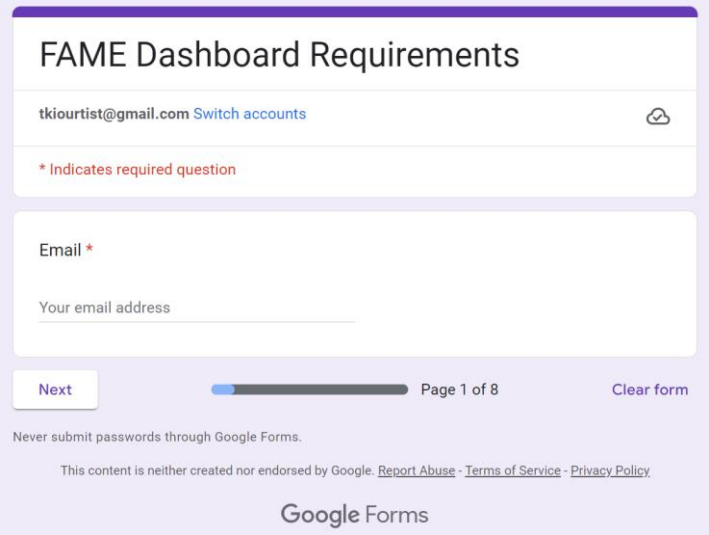


Figure 11 : FAME Dashboard components' requirements collection

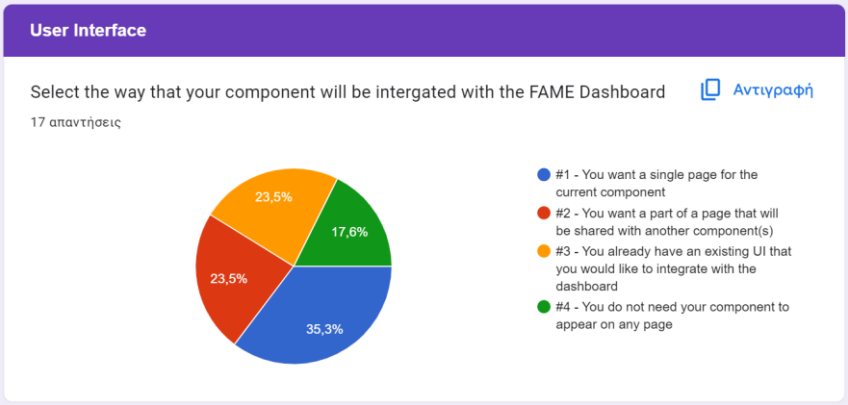


Figure 12 : FAME Dashboard components' needed UIs feedback

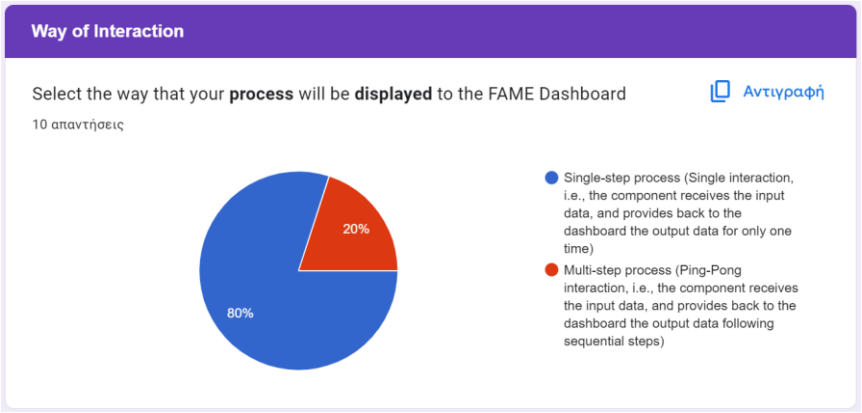


Figure 13 : FAME Dashboard components' UIs interactions feedback

It should be highlighted that through this questionnaire, the interactions among the different backend services were restructured (where needed) through the proper questions, as well as specific UIs were re-designed. The collected results can be seen in Table 2 : Status of backend services UIs that includes the list of all the available backend services of the FAME Data Marketplace, and their related requirements for a specific UI or not.

It should be mentioned that *Step 3 - Step 7* of the design process described below was followed only for the backend services and content that did not have any implemented UI and were implemented in the context of the FAME Dashboard (i.e., to-be-implemented - *TBI*). For the rest of the backend services (i.e., the ones that already had a UI (*Implemented*) outside of the context of the FAME Dashboard), a dedicated UI design methodology was followed as described in the corresponding backend services' deliverables (*Deliverable* column in Table 2).

Table 2 : Status of backend services UIs

FAME Stakeholder	Backend Service/Content	UI Status	Deliverable
End-user	Federation application	Implemented	D2.4 (current deliverable)
	Home page	TBI	D2.4 (current deliverable)
	About us	TBI	D2.4 (current deliverable)
	Technological framework	TBI	D2.4 (current deliverable)
	Business framework	TBI	D2.4 (current deliverable)
	Legal framework	TBI	D2.4 (current deliverable)
	Terms of use	TBI	D2.4 (current deliverable)
	Helpdesk	TBI	D2.4 (current deliverable)
	User login	Implemented	D3.4 [32]
	User profile	TBI	D2.4 (current deliverable)
	Trading account enrollment	Implemented	D4.5 [9]
	Asset publishment	Implemented	D4.4 [8]
	Data assets catalogue	Implemented	D3.5 [7]
	Asset details	Implemented	D3.5 [7]
	Asset policy manager	Implemented	D3.4 [32]
	Offering definition	Implemented	D4.4 [8]
	Offering details	Implemented	D4.4 [8]
	Pricing advisor	Implemented	D4.5 [9]
	Learning centre	Implemented	D7.4 (upcoming deliverable)
	Course details	Implemented	D7.4 (upcoming deliverable)
	Regulation tool	Implemented	D3.6 [10]
Administrator	Home page	TBI	D2.4 (current deliverable)
	Admins	TBI	D2.4 (current deliverable)

	Admin onboarding	TBI	D2.4 (current deliverable)
	Admin details	TBI	D2.4 (current deliverable)
	Members	TBI	D2.4 (current deliverable)
	Member onboarding	TBI	D2.4 (current deliverable)
	Member details	TBI	D2.4 (current deliverable)
	Users	TBI	D2.4 (current deliverable)
	User onboarding	TBI	D2.4 (current deliverable)
	User details	TBI	D2.4 (current deliverable)
	User tradings	TBI	D2.4 (current deliverable)

This collected knowledge led to the next step of the FAME Dashboard design process - namely the creation of the Visual Sitemap - considering all the existing backend services of the FAME platform.

3.2.0.1 Step 2: Visual Sitemap Creation

A Visual Sitemap is an essential information architecture step that can visually expose difficult-to-crawl content and help refine a website's navigation and overall structure before and after it is built. Hence, its purpose is focusing on illustrating the relationships among all the pages of a website, and providing a clear picture of the website's content, organization, and navigation. In essence, a Visual Sitemap represents the structure of a website in an easy-to-understand visual diagram. In the context of the FAME Dashboard, such concept was quite crucial for plotting out the whole platform's frontend and backend functionalities, capturing the current state of the platform visualized outcomes and results, planning in parallel for future needs and updates. Through the FAME Dashboard Visual Sitemap, it has been easily ascertained how many pages are needed in place to cover all the underlying services/functionalities, along with the predefined requirements from the previous step, whereas it was also identified whether there are missing any crucial elements or not, how fast the FAME Dashboard could deliver the desired content to its users, as well as how efficiently its users could navigate themselves across the entire environment. The Visual Sitemap design was performed using the Octopus Visual Sitemap tool [14], a web-based platform that facilitates the creation of interactive and collaborative sitemaps through a user-friendly drag-and-drop interface. This tool supports real-time editing, version control, and hierarchical structuring, making it ideal for both design iteration and stakeholder engagement. The sitemap design process carefully considered the needs of both end-users and FAME Data Marketplace administrators, resulting in the development of two (2) distinct Visual Sitemaps tailored to their respective navigation flows and functional priorities.

As for the end-users Visual Sitemap, illustrated in the figure below, it can be seen that the landing page of the end-user is the Dashboard Home from which the end-user can navigate to the different 1st level backend services/functionalities of the FAME Data Marketplace, referring to the Federation Application, the Login functionality, the Profile, the FDAC, the Publish functionality, the Learning Centre, the About Us page, the Helpdesk, the Terms of Use, and the Regulation Tool, which are directly accessible either from the Dashboard Home menu bar or the provided call-for-action buttons residing within the Home page. The rest of the backend services (i.e., Trading Account Enrollment, Asset, Offering Definition, Offering Details, Asset Policy Manager, Pricing Advisor, Course details, FAME Technological Framework, FAME Business Framework, FAME Legal Framework) are also accessible through the FAME Dashboard, with the difference that they can be reached through the

respective internal pages of the respective 1st level services, since their functionalities are implemented as part of those 1st level services' procedures. To this end, it should be noted that in the figure below, are also depicted the actions that can be performed to each page from the end-user side, as well as the connections among the different pages that interact with the underlying backend services. Further information on the detailed content of each page depicted in the end-users' Visual Sitemap is provided in Section 3.3.2.4, where the user manual of the whole implementation is provided.

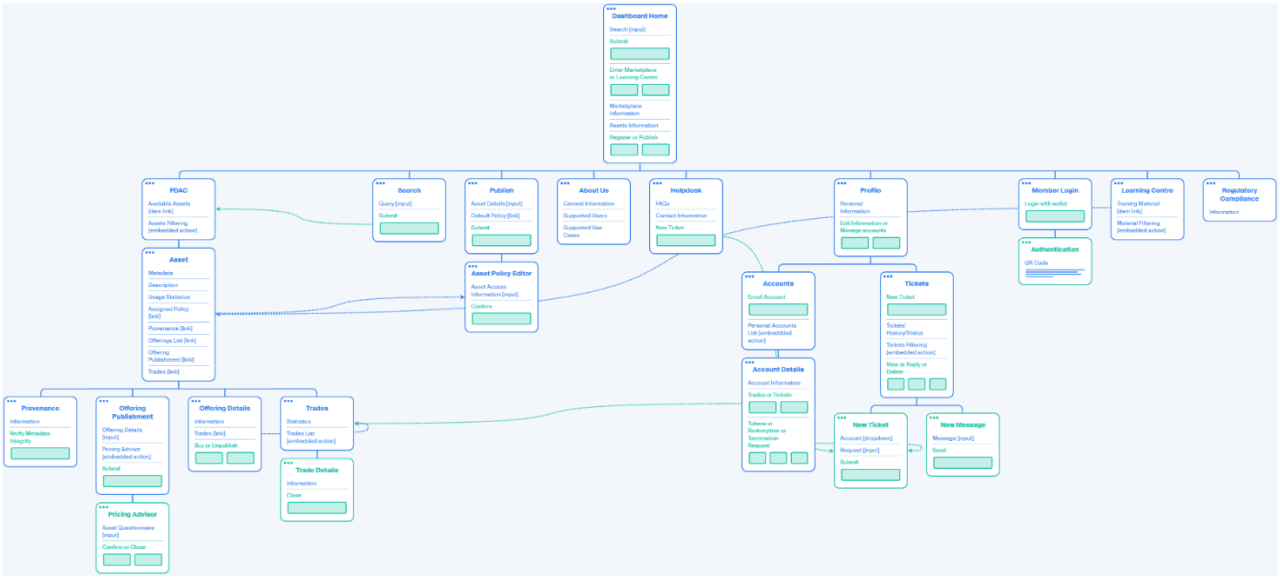


Figure 14 : FAME Dashboard end-users' Visual Sitemap

As for the administrators' Visual Sitemap, illustrated in the figure below, it can be seen that the landing page of the administrator is the Admin Home from which the administrator can monitor the platform general statistics (e.g., federation members, trusted sources, catalogue entries) and navigate to the different backend services/functionalities of the administrative part of the FAME platform, referring to the Administrators', the Members', the Trusted Sources', and the Traders' management pages, as well as the related Tickets' overview page. To this end, it should be also stated that in the figure below, there are additionally depicted the actions that can be performed to each page from the administrator side, as well as the interactions among the different pages. Further information on the detailed content of each page illustrated in the administrators' Visual Sitemap are provided in *Step 6*, where the actual mockups of all these pages are provided.

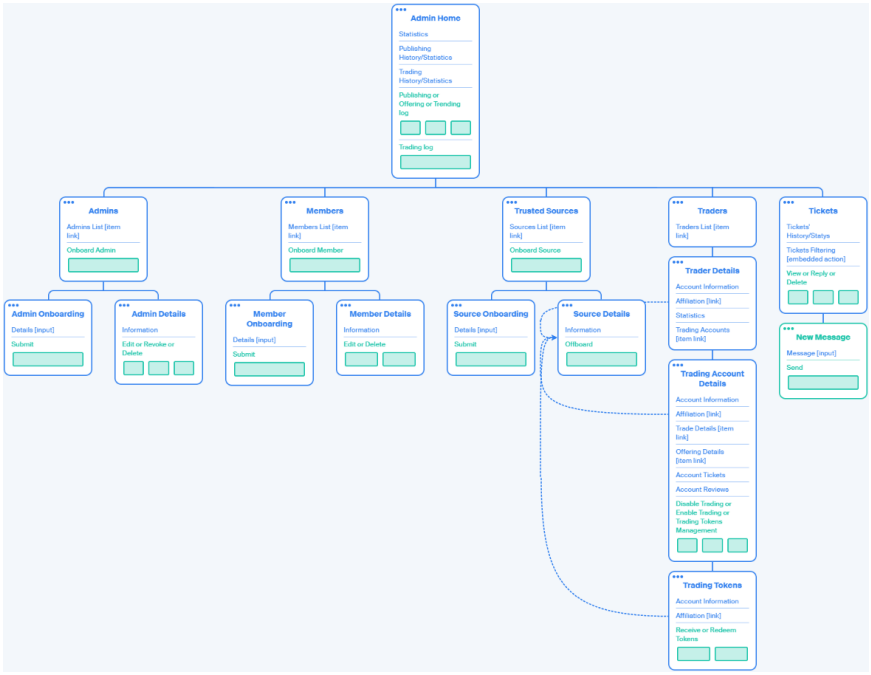


Figure 15 : FAME Dashboard administrators' Visual Sitemap

The fulfilment of this part led to the initiation of the next step, regarding the Wireframes generation.

3.2.0.2 Step 3: Wireframes Generation

After the creation of the Visual Sitemaps, a plethora of ideas had been generated around the structure and functionality of the overall solution, and that is where the concept of Wireframes was developed before moving to the final UI design. This step of the process was a crucial intermediary step between the conceptual design and the final UI, which focused on giving shapes to the rough ideas that were collected from the previous steps. While wireframing, text was being added, as well as sample relatable images, and dummy content (i.e., lorem ipsum text) to simulate content flow and user interaction, ensuring that layout, hierarchy, and usability were effectively addressed before proceeding to high-fidelity UI design. In the context of the FAME Dashboard, the wireframes developed through Microsoft PowerPoint involving the generation of low-fidelity outlines of the layout and structure of the complete UI. These Wireframes served as a blueprint for the design, helping into clearly identifying the placement of elements, information hierarchy, and overall flow before diving into the visual design. Moreover, Wireframes also facilitated into validating the design with all the FAME Data Marketplace key stakeholders, allowing everyone to comment about what element and text goes where before anything gets finalized, and to achieve high quality results. Since multiple Wireframes were generated, being considered as primary steps of the Mockups that were generated afterwards (explained in detail in Section 3.3.1.5), only an example of a Wireframe designed for the administrator Home page of the FAME platform is being depicted in this deliverable (see the figure below) to minimize its content and total number of pages. It should be mentioned that a similar Wireframe design concept has been followed for all the pages that can be identified in the Visual Sitemaps, resulting into a plethora of Wireframes.

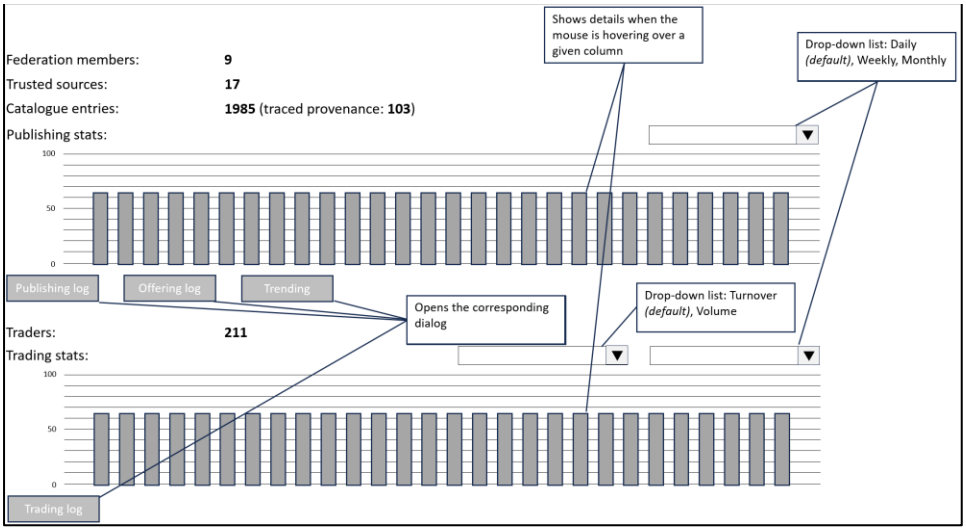


Figure 16 : FAME Dashboard administrators' home page Wireframe

The finalization of the Wireframes gave the green light to the next step related with the concept of templatization.

3.2.0.3 Step 4: Templatization

The process of templatization included the creation of reusable components across all the platform's UIs, for multiple atomic elements like buttons, input fields, or texts, among others. In general, templatization and component creation in the context of the FAME Dashboard was performed to create consistency among elements, and for all the design and implementation process to follow a common design pattern and paradigm. All these components were created by adding images, graphics text, and related placeholders, leading into a homogeneous format. The figure below illustrates a snapshot of the templatization results, showcasing the way that the navigation bar should be designed, and the related colorings of the texts and buttons that each UI designer should comply with.

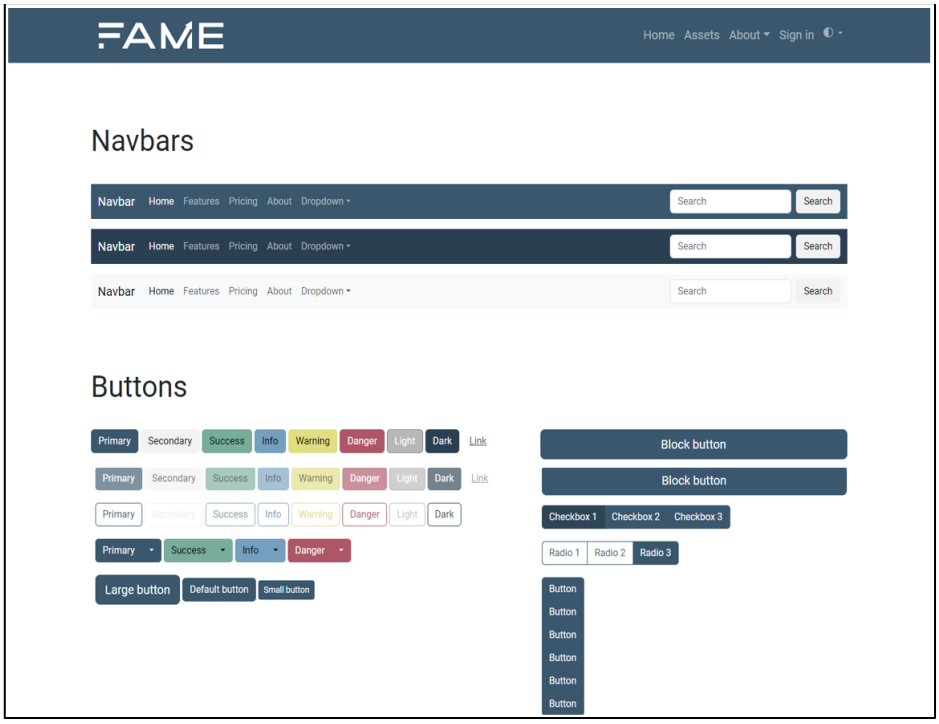


Figure 17 : FAME Dashboard templatized visual elements

When all the aforementioned steps have been completed, the Mockups creation phase initiated.

3.2.0.4 Step 5: Mockups Implementation

This step was to design the screens as they would appear in the final product. Close to the pixel-perfect design, using all the work and data collected from the previous steps, adding all the text copies, graphics, icons, and images to the design. The mockups implementation phase is the step where all the colors, fonts, and aesthetics are added to the screens. For the FAME Dashboard, the latter process was conducted through FIGMA [15], a collaborative interface design tool known for its real-time co-editing and prototyping capabilities, whereas once the design was finished, the related screens were shown to the end-users to gather related feedback. A set of refinements and a/b testing variants were decided based on this review to explore alternative design solutions and improve UX, whereas as the designs were finalized, assets like images, text content, translations, icons, and illustrations, were systematically prepared and handed over to the development team for implementation.

3.2.0.4.1 End-user Mockups

The Mockups that have been designed from the side of the end-users are provided in this sub-Section. It should be noted that the purpose of this Section is not to provide an end-user manual, since this is depicted in Section 3.3.2.4. As a result, since the end-user mockups and the various screens presented in the end-user manual represent overlapping content, this sub-Section includes only a selected subset of the mockups. This approach is intended to avoid redundancy, while still providing sufficient proof-of-concept visuals to demonstrate the design intent and UI structure.

In this context, it should be also noted that all the different Mockups are based on a specific skeleton page (Figure below) that includes all the necessary information that should always exist in each different page of the end-users FAME Dashboard, whereas its main content is the one that is being adapted based on the purpose of each page and its related backend service. This necessary information includes the navigation menu bar from which the end-users can login and access to the different core pages of the FAME Data Marketplace, and the widgets through which the end-users can have immediate access to the FDAC, the Learning Centre, the Regulation tool, and the FAME Chatbot. It also includes the footer of the page containing information about the FAME interconnected core applications (i.e., Federation Application, FDAC, Learning Centre, Regulation Tool), general information (i.e., FAME Core Values, Ecosystem, Stakeholders, Success Stories), end-users' support information (i.e., User Manuals, FAQs, Contact, Newsletter), and related Communication/Social Media/Terms of Use information. As a result, all the pages that are depicted below are built based on this skeleton page, equally adapting their content to their purpose of use.

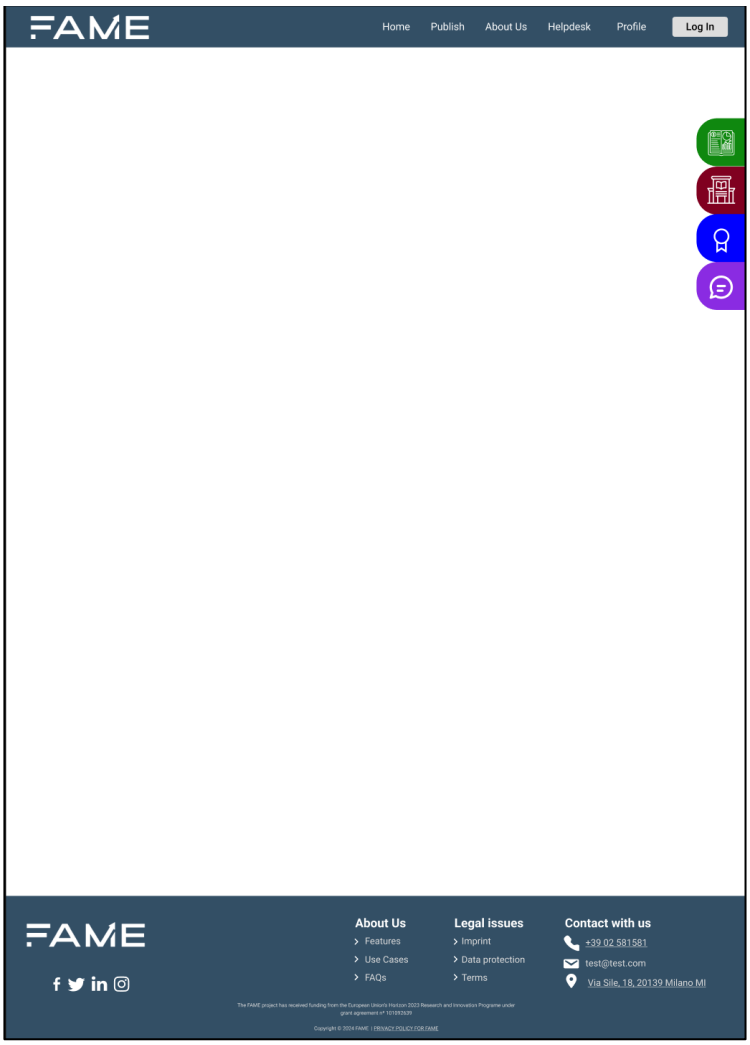


Figure 18 : FAME Dashboard end-users' skeleton page

With that in mind, the rest of the Mockups follow. The Home page of the end-users can be seen in the figure below, showcasing the exact way that the FAME Dashboard Home page will be finally implemented.

Figure 19 : End-users' home page Mockup

Respectively, a similar process has been repeated for the rest of the end-users' pages, with all this information being thoroughly explained in Section 3.3.2.4.

3.2.0.4.2 Administrator Mockups

The Mockups that have been designed from the side of the administrator are provided accordingly in this sub-Section. It should be noted that, as in the case of the end-users' side, the purpose of this sub-Section is not to provide an administrator manual, since the implementation of the administrator's UI falls outside of the defined scope of the FAME project, and thus it will not be considered in the context of this deliverable. This decision was based on several factors, including prioritization of core user-facing features, limited project resources and timeline constraints, and a focus on validating end-user interactions first, which were considered critical for the proof-of-concept phase. Nevertheless, for the sake of completeness and future extensibility, a set of reference mockups has been provided since the FAME administrator back-end functionalities are already implemented, without however being integrated into any UI implementation.

In this stage, it should be also noted that as in the case of the end-users, all the different Mockups of the administrator side are based on a specific skeleton page (Figure 20) that includes all the necessary information that should always exist in each different page of the administrator FAME Dashboard, whereas its main content is the one that is adapted based on the purpose of each page and its related backend service. This necessary information includes the horizontal navigation menu bar from which the administrator can logout, as well as the vertical navigation menu bar from which the administrator can access the different pages of the administrator FAME Dashboard for managing the administrators, the members, the users, the traders, and the platform tickets. The footer of the skeleton page also contains information about the FAME interconnected core applications (i.e., Federation Application, FDAC, Learning Centre, Regulation Tool), general information (i.e., FAME Core Values, Ecosystem, Stakeholders, Success Stories), end-users' support information (i.e., User Manuals, FAQs, Contact, Newsletter), and related Communication/Social Media/Terms of Use information. As a result, all the pages that are depicted below are built based on this skeleton page, equally adapting their content to their purpose of use.

Figure 20 : FAME Dashboard administrators skeleton page

Following the latter, the Home page of the administrator is offered through the figure below.

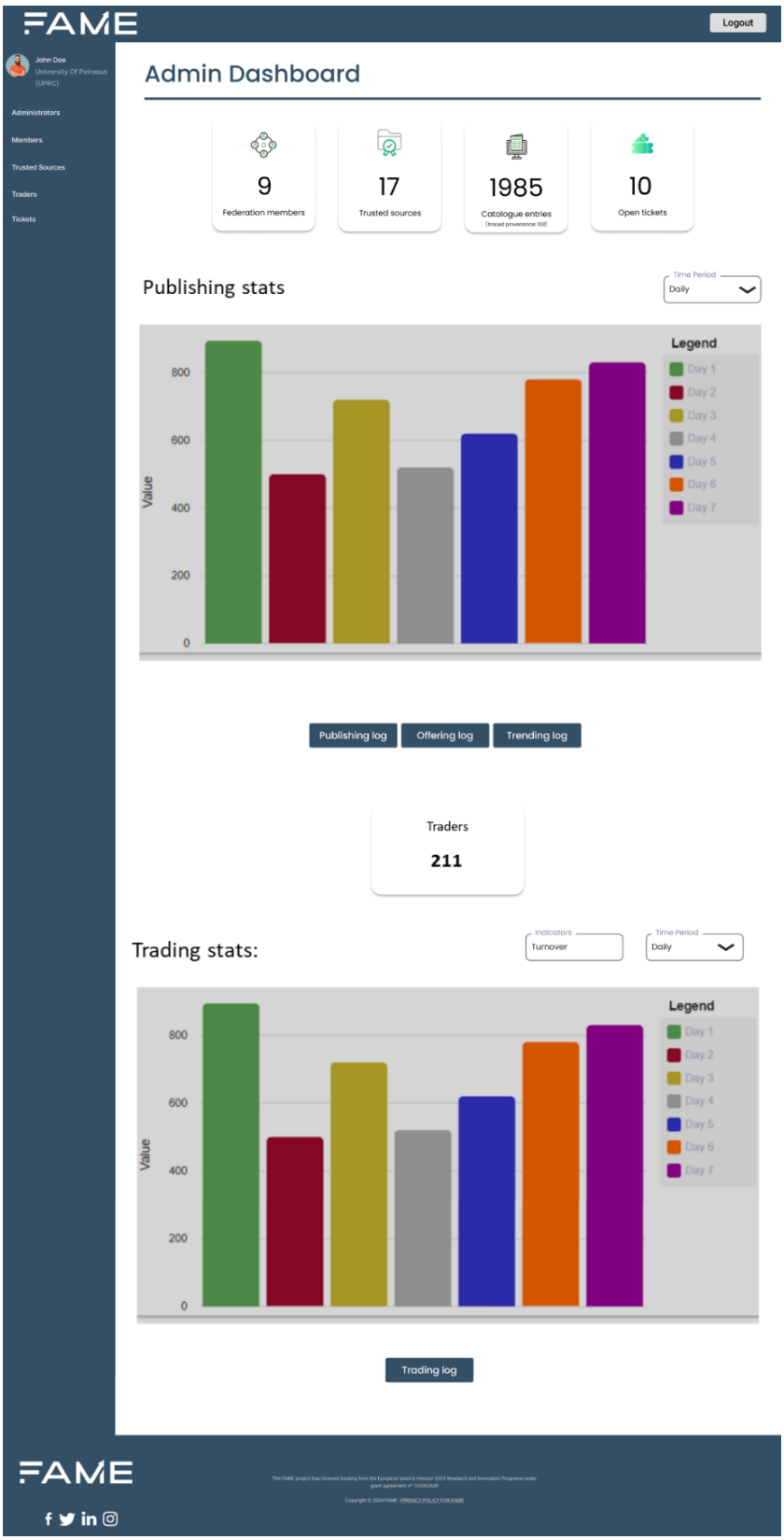


Figure 21 : Administrators' home page Mockup

The rest of the mockups (Administrators management page, Administrator onboarding page, Administrator details page, Members page, Users page, User details page, Members page, Member details page, Traders page, Trader details page, Trading accounts details page, Trading tokens page) are following a similar pattern as already described in the previous version of this deliverable (D2.3 - Integrated FAME Data Marketplace I) with slight updates and enhancements in the overall UI, following the FAME Dashboard administrators' skeleton page.

By the time that the Mockups were approved and finetuned based on all the stakeholders and FAME technical components developers' needs, the last phase of the design and development of the FAME Dashboard initiated, namely the frontend development.

3.2.0.5 Step 6: Frontend Development

By the time that the Mockups design was finalized, the actual frontend development phase started, which had to deal with the preparation of the detailed design specifications and assets for the developers, so they could ensure that the final product closely matches the design vision. For the FAME Dashboard, since the complexity of the design was high, specific interaction guidelines were additionally provided that explained how interactions like hover effects, transitions, or animations should behave. Moreover, related style guides were generated, which served as comprehensive documentation of the visual and interaction design guidelines, branding elements, and UI components used in the design. All in all, these style guides also helped in establishing and maintaining a consistent visual language across the FAME Dashboard, and as a result across the overall FAME Data Marketplace UIs. Finally, it should be noted that during the development phase some further updates were performed in the already provided Mockups (*Step 5*), to offer a fully compliant UI based on the relevant feedback. At this stage, all the different Mockups were already implemented, whereas their integration with the different backend services was taking place. Related information on how the FAME Dashboard UIs can be executed and experimented by any stakeholder is provided in Section 3.3.2.3.

3.2.0.6 Step 7: Testing & Iteration

The Testing and Iteration phase was performed as a final step, in which multiple internal testing activities took place in the context of the different FAME Dashboard design phases (as explained in the previous Sections) including a plethora of well-organized and targeted user testing. The overall idea behind these tests was that the designs might look ideal from where the developers' side, but it is important to remember that the developer is not the end-user. As a result, testing the prototype with real end-users helped into identifying any usability issues or pain-points that might have been missed along the way. Furthermore, it helped into getting a sense of how the visual design strategy was resonating with real end-users and how it met their expectations.

To this end, there were conducted multiple usability testing sessions with target end-users representing both technical and non-technical personas. These testing sessions included:

- Click-through prototype experimentation to evaluate user navigation pathways.
- Task-based assessments, in which the different end-users had to complete specific actions (e.g., locating data visualizations, accessing reports, customizing dashboard widgets) to gauge ease of use and efficiency.
- A/B testing of alternative design variations to perform comparisons on user preferences and performance metrics.
- Think-aloud protocols through online meetings, physical training sessions, and the face-to-face General Assembly (GA) meetings, where participants expressed their thoughts, while using the prototype, and revealing areas of confusion.

- Feedback online surveys to capture qualitative outcomes about the overall UI/UX, clarity of information, and overall satisfaction of the users.
- Use of heatmaps and session recording – after consent approval – to analyze user behavior on specific functionalities of the platform, improving the overall UX in a continuous way.

The results of the testing sessions identified usability issues (e.g., unclear icons, inconsistent terminology, unnecessary complexities in workflows), providing insights into how well the visual design strategy aligned with the end-users' requirements. Iterative refinements were made accordingly, ensuring the final interface was both intuitive and user-centered.

3.2.1 Dashboard development process

3.2.1.0 Dashboard technologies

The FAME Dashboard has been released as a responsive web application, whose frontend part primarily utilizes **Angular** [16] (version: Angular v16.2.12) along with the following technologies:

- **Docker** [17]: Docker is used to containerize the Angular application, facilitating easy deployment and scalability, ensuring consistency across different environments, and simplifying the deployment process.
- **HTML** [18]: HTML is used for structuring the content of the web application, providing a solid foundation for the presentation layer (version: HTML5).
- **SCSS** [19]: SCSS (Sass) is employed for styling the web application. Its powerful features such as variables, nesting, and mixings aid in maintaining scalable and maintainable CSS code (version: SCSS (Sass) v1.64.1).
- **TypeScript** [20]: TypeScript serves as the primary programming language for building the frontend of the web application. Its static typing and modern syntax help in writing cleaner and more structured code (version: TypeScript v5.1.3).
- **Angular Material** [21]: Angular Material is utilized for styling and theming the web application. Angular Material provides a set of pre-built UI components following the Material Design principles, enabling the development team to create a visually appealing and consistent user interface (version: Material v16).
- **Npm Packages**: Additional libraries and frameworks have been exploited depending on the web application's requirements, ensuring the efficient development and delivery of the application. Such libraries or frameworks include:
    - HttpClient: Angular's HttpClientModule for making Hypertext Transfer Protocol (HTTP) requests to a server. This package provides a simplified API for interacting with RESTful services.
    - RxJS: Reactive Extensions for JavaScript (RxJS) for handling asynchronous operations and managing streams of data.
    - Angular Router: While Angular's router comes built-in, additional features are achieved by utilizing npm packages related to routing and navigation.
    - Angular Forms Modules: Alongside the core forms' functionality, additional npm packages have been considered for form validation, custom form controls, and form layout management.
    - Angular CLI: Although not strictly a package, Angular Command Line Interface (CLI) provides various commands and utilities for scaffolding, building, and managing Angular applications. It is essential for efficient development workflows.
    - Testing Frameworks: Npm packages for testing frameworks have been utilized for writing and running tests for Angular applications.

- Lodash: A utility library has been exploited for aiding data manipulation in a functional programming style.
- angular-auth-oidc-client: A fully featured OpenID Connect (OIDC) and OAuth 2.0 library for Angular applications, enabling secure authentication flows with Identity Providers (IdPs), supporting silent refresh, access token renewal, and role-based access control. This is essential for integrating the Dashboard with external SSO (Single Sign-On) systems or identity federation infrastructures (version: v16.0.0).
- Base-x: A lightweight and fast base encoding/decoding library, used for encoding identifiers and handling data formats commonly found in blockchain and decentralized systems (version: v4.0.0).
- Buffer: A Node.js-compatible buffer implementation for browsers, allowing binary data manipulation. It is especially useful for dealing with cryptographic operations and binary data streams (version: v6.0.3).
- Viem: A TypeScript-first Ethereum library for building decentralized applications (dApps) and interacting with Ethereum-compatible blockchains. It is used to interface with smart contracts and support possible integration scenarios with decentralized data sources or Web3 components (version: v2.17.4).

Furthermore, for providing a fully functional Dashboard, the latter has been integrated with technologies from external sources seamlessly. The following key features of Angular play a significant role in achieving this integration:

- **Two-way Data Binding**: Angular's two-way data binding feature facilitates the automatic synchronization of data between the model and the view. This reduces the need for manual DOM manipulation, enhancing the efficiency of the developed application.
- **Component-Based Architecture**: Angular promotes the use of reusable and encapsulated components. This architecture makes it easier to manage and maintain complex UIs by breaking them down into smaller, manageable pieces.
- **Dependency Injection**: Angular boasts a powerful dependency injection system. It helps managing component dependencies, making the developed application more modular and testable, while enhancing code maintainability.
- **Directives**: Angular provides both built-in and custom directives. These directives enable developers to create custom behaviors and manipulate the DOM effectively, enhancing the flexibility of our application.
- **Routing**: Angular offers a robust routing module to handle navigation between views and pages in a single-page application. This feature ensures smooth navigation and enhances the UX.
- **Forms**: Angular provides robust support for building both template-driven and reactive forms. This makes it easy to handle user input and validation, ensuring data integrity and enhancing user interaction.
- **Services**: Angular allows the creation of services to encapsulate and share functionality across components. This feature facilitates better management of data, API calls, and other common tasks, enhancing the overall efficiency of our application.
- **Authentication Module**: A dedicated auth module is used to manage user authentication and authorization mechanisms. It supports login flows, token storage, role-based access control (RBAC), and integration with OIDC-compliant identity providers using libraries such as angular-auth-oidc-client.
- **Interfaces for API Models**: The frontend defines a consistent set of interfaces to represent the structure of the data received from the various backend APIs. This improves type safety,

code clarity, and the maintainability of the data-handling logic in Angular components and services.

- **Custom Validators for Blockchain Integration**: Custom validators are implemented to validate blockchain addresses (e.g., Ethereum format) within user inputs. This ensures data correctness and avoids invalid operations in Web3-related functionalities.
- **Environment-Specific Configuration**: The developed application leverages Angular's environments system to manage configuration settings across different deployment phases (e.g., development, staging, production). Each environment contains specific variables such as API endpoints, logging flags, and authentication settings, enabling smooth CI/CD and safe environment switching.

3.2.1.1 Dashboard design principles

Adhering to ***fundamental design principles*** is also quite crucial for the platform's success and users' satisfaction. The essential core principles that are followed by all the FAME Dashboard pages are:

Simplicity: Aim for a straightforward design, reducing complexity of the platform's content.
Consistency: Maintain both visual and operational uniformity across all platform pages to aid in user interaction.
Efficiency: Structure the pages' information thoughtfully to facilitate effective navigation and use.
Accessibility: Embrace user diversity by designing inclusive pages accessible to individuals of varying abilities.
Scalability: Design with an eye toward future expansion and adaptability, streamlining future adjustments and maintaining platform coherence.
Memorable Design: Create an engaging and memorable experience, trying to foster an emotional connection with the user.
Aesthetic and Minimalist Design: Favor a design that is both visually appealing and minimalist, evoking a positive reaction and enhancing the perceived effectiveness of the platform.

The FAME Dashboard also adopts key principles of ***SEO (Search Engine Optimization)*** to enhance its digital footprint and ensure unparalleled visibility across search engines, prioritizing:

- **PProper Response Codes**: Ensure web pages return a 200 HTTP status code for successful rendering and update or remove unnecessary redirects to optimize user and crawler access.
- **URL Structure**: Craft URLs that are concise, descriptive, lowercase, and free of unnecessary characters, reflecting the platform's content hierarchy.
- **Mobile-Friendliness**: Optimize the pages' content for mobile viewing and Google's mobile-first indexing, ensuring the mobile site contains all the critical content.
- **Site Speed Optimization**: Focus on reducing page load times through efficient code, optimized images, and minimized use of heavy plugins.

In FAME's Dashboard development, adhering to ***green principles*** not only supports environmental sustainability but also enhances user experience. To incorporate these values, all of its pages support:

- **Minimalist Design**: Streamline interfaces to include only essential elements, reducing cognitive load and energy consumption.
- **Dark Colors**: Utilize darker color palettes to reduce energy usage on devices with OLED screens, promoting longer battery life.
- **Compressed Media Files**: Ensure media files are optimized for quick loading, conserving bandwidth, and improving access times.
- **Full-responsive Website**: Craft a design that adapts seamlessly to any screen size, minimizing the need for redundant resources across devices.

- **Clear and Concise Code**: Write an efficient code that performs tasks without unnecessary complexity, reducing processing time and energy usage.

3.2.1.2 Dashboard source code

As stated in the previous Sections, the FAME Dashboard is a crucial component, providing users with an intuitive interface to interact with the platform's functionalities. This Section details the source code and the deployment procedures for exploiting the FAME Dashboard, catering to both non-Docker and Docker environments. To this context, the users should follow the below instructions carefully to set up the FAME Dashboard correctly.

Initially, the following programs and libraries are required to be installed into the user's environment to be able to perform the rest of the subsequent instructions:

- WSL [22]
- Git [23]
- Node.js [24]
- Docker (non-required for section ***Non-Docker Environment***) [25]

Moreover, in order for someone to be able access and install the FAME Dashboard, related access credentials should be granted to the following repository: https://gitlab.gftinnovation.eu/

The versions of the required programs and libraries currently installed on the machine where the development takes place are the following:

```
C:\Users\user_fame> git --version
git version 2.32.0.windows.2
C:\Users\user_fame> node -v
v18.14.2
C:\Users\user_fame> npm -v
9.5.0
C:\Users\user_fame> nvm list
  * 18.14.2 (Currently using 64-bit executable)
    16.14.0
    16.13.1
C:\Users\ user_fame > docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Build with BuildKit (Docker Inc., v0.6.1-docker)
  compose: Docker Compose (Docker Inc., v2.0.0-rc.1)
  scan: Docker Scan (Docker Inc., v0.8.0)
Server:
 Containers: 17
  Running: 5
  Paused: 5
  Stopped: 7
 Images: 19
 Server Version: 20.10.8
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
```

```
Default Runtime: runc
Init Binary: docker-init
containerd version: e25210fe30a0a703442421b0f60afac609f950a3
runc version: v1.0.1-0-g4144b63
init version: de40ad0
Security Options:
 seccomp
  Profile: default
Kernel Version: 5.10.16.3-microsoft-standard-WSL2
Operating System: Docker Desktop
OSType: linux
Architecture: x86_64
CPUs: 12
Total Memory: 12.36GiB
Name: docker-desktop
```

Non-Docker Environment

To execute the FAME Dashboard without using Docker, the user should ensure that Node.js version 18 is installed. If not, the Node Version Manager (NVM) [26] packager can be used to download and manage Node.js versions on the preferred machine. The related steps follow:

Step 1: Install Node.js

- In case that Node.js 18 is not installed, NVM should be used to install it. The installation guide for Node.js and NPM should be followed on the desired operating system [27].
    - To install NVM on Windows, the instructions are as follows:
        - Download the NVM Windows installer from the NVM for Windows GitHub repository [26].
        - Execute the installer and adhere to the on-screen prompts.
        - After the installation finishes, open a new command prompt or PowerShell window.
        - Confirm the installation by running the following command: `nvm --version`
        - If everything is set up correctly, you should see the NVM version number.

Step 2: Clone the repository

The terminal should be opened to run the following command: `git clone https://gitlab.gftinnovation.eu/fame/fame-dashboard.git`

- This command will clone the FAME Dashboard code into the local machine.

Step 3: Navigate to the project folder

- The directory must be changed to the project folder: `cd fame-app`

Step 4: Install the required dependencies

- The following command should be run to install all the necessary Node modules: `npm install`
- In some cases, particularly when dealing with strict or incompatible peer dependencies (e.g., during CI/CD in containerized environments or when installing legacy packages), the following alternative command may be used to bypass peer dependency conflicts: `npm install --legacy-peer-deps`

Step 5: Serve the application

- The application can start with the command: `ng serve --open`
- This command will automatically open the default browser, and it will accordingly be navigated to ***localhost:4200***, where the Dashboard will be running.
- The `--open` flag is optional and will automatically open the default browser to ***localhost:4200***. If it is preferred for the browser to open manually, the following command can be executed: `ng serve`

Docker Environment

In the case that Docker is preferred to be used, the following steps outline how to set up and execute the FAME Dashboard using Docker containers.

Step 1: Clone the repository

- As in the case of the non-Docker setup, the repository must be cloned into the local machine: `git clone https://gitlab.gftinnovation.eu/fame/fame-dashboard.git`

Step 2: Add docker configuration

- The `docker-compose.yml` file must be added to the root level of the cloned repository. The structure should look like the following one:

```
├── fame-app
├── docker-compose.yml
```

Step 3: Run docker compose

- The user should navigate to the root level of the repository where the docker-compose.yml file is located, and execute the command: `docker-compose up -d`
- This command will build and start the Docker containers in detached mode.

Step 4: Access the Dashboard

- The browser must be opened, and the user should navigate to ***localhost:4200***. The FAME Dashboard will be running, being fully accessible.

By following these steps, a user can set up and run the FAME Dashboard either with or without exploiting Docker, depending on the environment preferences. This flexibility ensures that anyone can work with the FAME Dashboard in a way that best fits the overall development setup.

3.2.1.3 Dashboard user manual

The FAME Dashboard provides a unified web-based environment for end-users to access, publish, explore, and manage data assets within the FAME Data Marketplace (i.e., the core technological component of the FAME Federated Data Space). It is designed with accessibility, modularity, and user guidance in mind. This manual outlines the key elements of the overall built user interface, providing representative screenshots of the current version of the Data Marketplace, and provides an overview of the main functionalities that are available to the users.

Figure 22 presents the **Home page** of the FAME Data Marketplace that is accessible by everyone, offering a high-level overview of the marketplace's ecosystem and purpose. At the top, the header introduces the marketplace's identity and through a dedicated menu bar provides access to key functionalities, referring to user's login, data assets' search, and user's navigation to the Home, About Us, Helpdesk, Publish, and Profile pages. Along with this menu bar, a sticky icon menu bar exists in the right part of the Home page, including four (4) discrete widgets via which the end-users can have immediate access to the FDAC, the Learning Centre, the Regulation Tool, and the FAME Chatbot. In the following paragraphs of this sub-Section, the content of all those pages is fully detailed. Below the menu bar, a central search bar exists that allows end-users to explore the data assets existing within the FAME Data Marketplace, while the "Discover FAME" section showcases the four (4) core technological modules of the marketplace: the Federation Application, the Data Assets Catalogue, the Learning Centre, and the Regulation Tool. Each module is supported by a minimal description and a link for direct access to the respective module. Further down, the "FAME Benefits" area outlines the main advantages of the FAME Data Marketplace, whilst the "Be Part of Our Community" section presents real-time platform statistics such as the number of federated members, active users, published data assets, and other community metrics, reflecting engagement and ecosystem growth. The "Publish with Us" area categorizes the type of data assets that the users can find and share with

the marketplace, having a variation between datasets, services, software, multimedia content, training courses, and others.

Finally, the “FAME Data Marketplace” section concludes the page with an embedded video tour of the marketplace, while the footer offers navigation to informative sections like Discover, About Us, Helpdesk, the Social Media, and the Terms of Use of FAME. As a final note on this page, there exists an accessibility icon positioned discreetly within the bottom left of this interface, which provides end-users with options to customize visual settings, such as contrast and font size, or to enable other accessibility enhancements. This feature ensures that the FAME Data Marketplace is inclusive and remains fully navigable and user-friendly for individuals with varying visual or cognitive needs.

Figure 22 : Home page

Figure 23 illustrates the content of the **About Us page** that is accessible by everyone, where end-users can locate detailed information of the usage and offerings of FAME. Initially, the "Join & Access FAME" section provides the end-users with step-by-step manuals with the processes of how to onboard organizations and users to the FAME Data Marketplace, how to enable trading functionalities, and how to publish or purchase data assets. Below, the "FAME in a Nutshell" area presents the core components of the FAME Data Space ecosystem, categorized into the three (3)

frameworks of technological, business, and legal, each one having its own dedicated page for locating additional detail (Figure 24). This section also emphasizes the core benefits of the FAME Data Space ecosystem, accompanied by the various stakeholder categories that can benefit from FAME, referring to businesses, consumers, scientists and government entities. The rest of the page is then dedicated to "FAME Success Stories", where short descriptions and accompanying images illustrate real-world applications of FAME, including financial services, ESG scoring, funding facilitation and climate prediction for insurance, thus highlighting the impact and adaptability of FAME in diverse sectors.

Figure 23 : About Us page

(a)

(b)

(c)

Detailed pages of (a) FAME Technological Framework, (b) FAME Business Framework, (c) FAME Legal Framework

Figure 24 :Detailed pages

Figure 25 displays the **Helpdesk page** that is accessible by everyone, where end-users can locate further assistance on how to exploit the marketplace functionalities and offerings. Initially, it includes the FAQs section of the FAME Data Marketplace, which is organized into the collapsible categories of: General, Target Audience, Federation and Transactions, each one containing a list of commonly asked questions, such as "What is FAME?", "Who can federate?", and "How are transactions managed?", offering clarity to both new and experienced end-users. Just below the FAQs, a section titled "Still have questions?" encourages end-users to contact support via email for further assistance. This is followed by a subscription prompt where end-users can join the FAME newsletter. At the bottom, the "Find Us Online" area provides quick links to FAME's official website and social media channels (i.e., X, LinkedIn, YouTube, Zenodo), reinforcing the platform's accessibility and outreach.

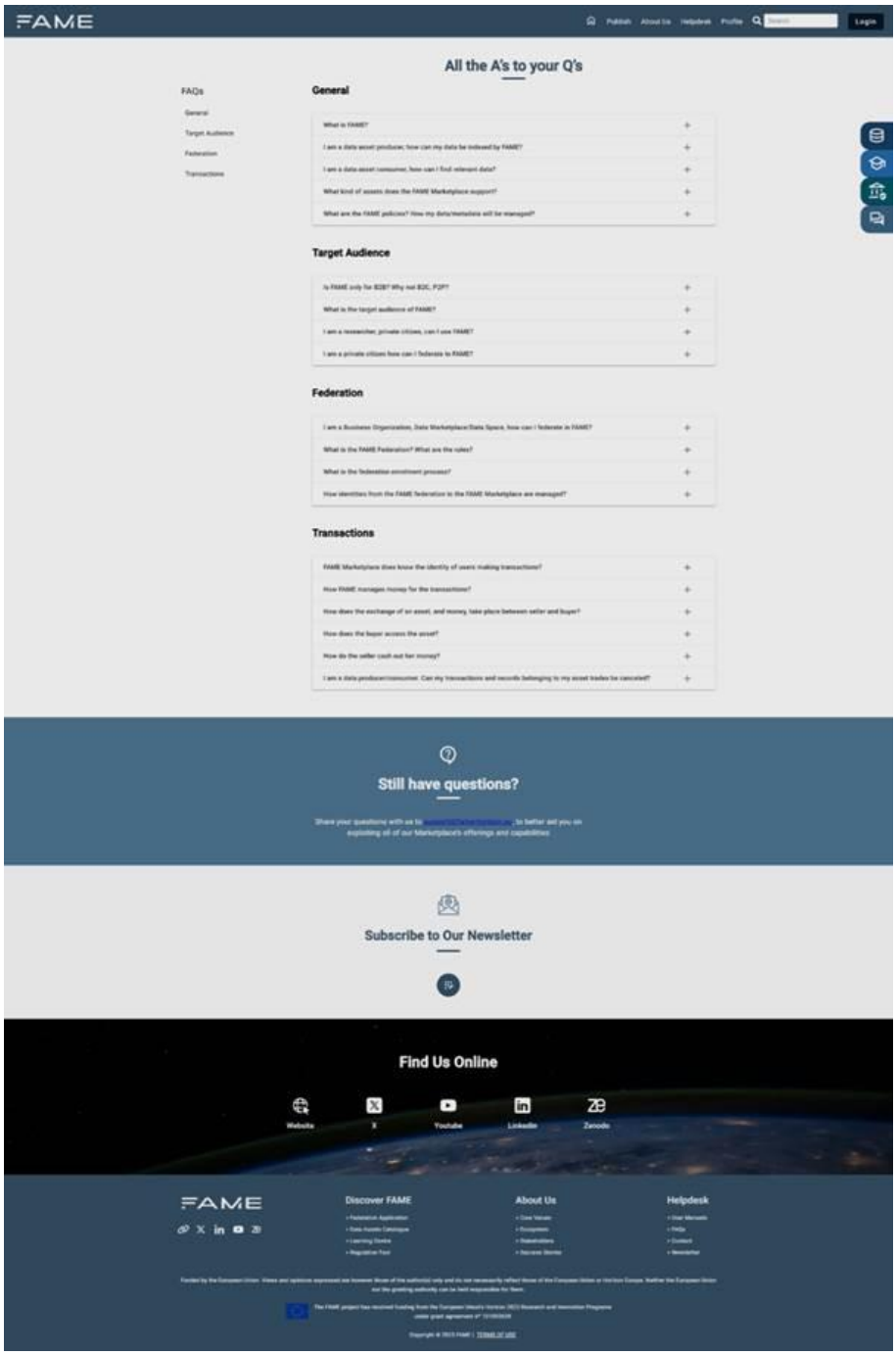


Figure 25 : Helpdesk page

Figure 26 showcases the **Login page** of the FAME Data Marketplace that is accessible by everyone, through which the end-users are able to be authenticated and granted access to all the functionalities of the marketplace. At the center of the screen, end-users are prompted to be authenticated by scanning a QR code using their mobile identity wallet application (i.e., Sphereon Wallet). This QR code facilitates secure identity verification and login into the system. Under the QR code, specific buttons are provided to download the identity wallet application from either the Google Play or the App Store, making the login process accessible for all mobile users. A small text link leads to the detailed step-by-step manual (located in the About Us page – Figure 2) for the end-users that may need any further assistance on this procedure, whereas another text links prompts end-users to proceed with completing the onboarding process (if not done yet), since the completion of this process is a prerequisite for successfully passing through the authentication mechanism of the marketplace.

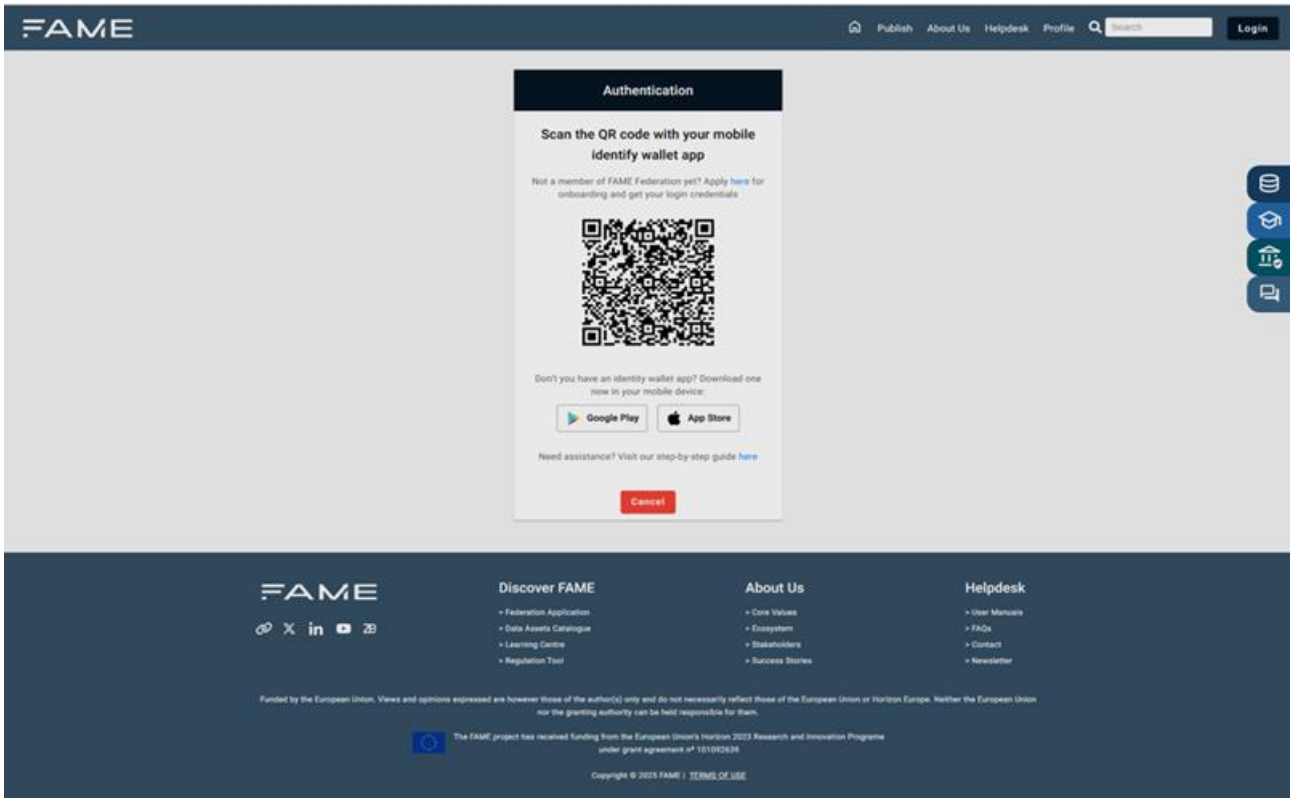


Figure 26 : Login page

Figure 27 presents the **Profile page** that is accessible only by logged in end-users, who can access further information on their created profile and associated accounts. On the left, the user's "Profile" information is displayed, including the user ID, role, affiliated institution and region. On the right, the "Accounts" section provides the user's wallet details, including the associated blockchain account address, current FDE token balance and the wallet connection status. A button for enrolling a new account is also present, allowing end-users to enroll their blockchain accounts in the marketplace. Figure 28 depicts the **Trading Account Enrollment page**, where end-users can register a blockchain account to the FAME Data Marketplace, to be able to participate in the existing trading activities. On the top of this page end-users are prompted to enter the public address of their blockchain account to be enrolled, while the central section outlines the terms and conditions required for the setup of the account, which have to be accepted by the end-users to activate the "Enroll" button. It should be noted that the end-users must already have installed in their web browser the Metamask Wallet, have in their position a blockchain account (existing in the Metamask Wallet), and also being connected to it to successfully complete the abovementioned enrollment process. As a final note, this page contains a breadcrumb navigation at the top left, helping end-users to track their location within their Profile.

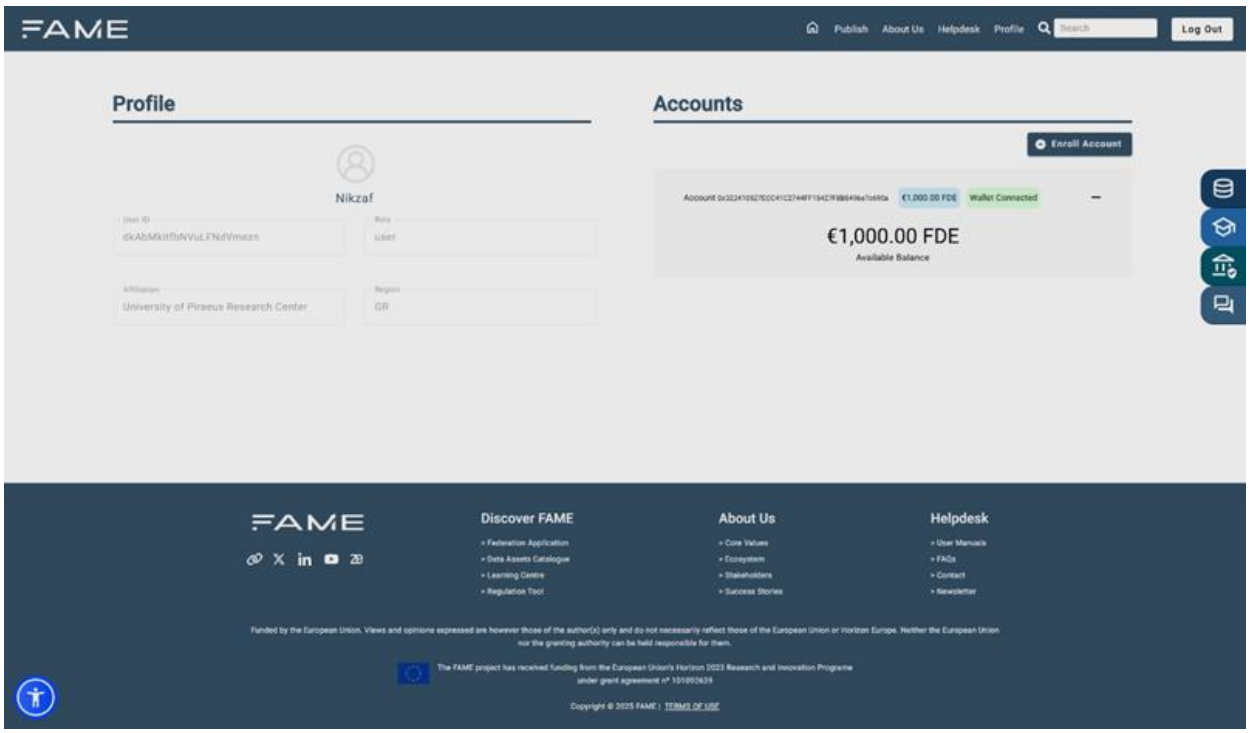


Figure 27 : Profile page

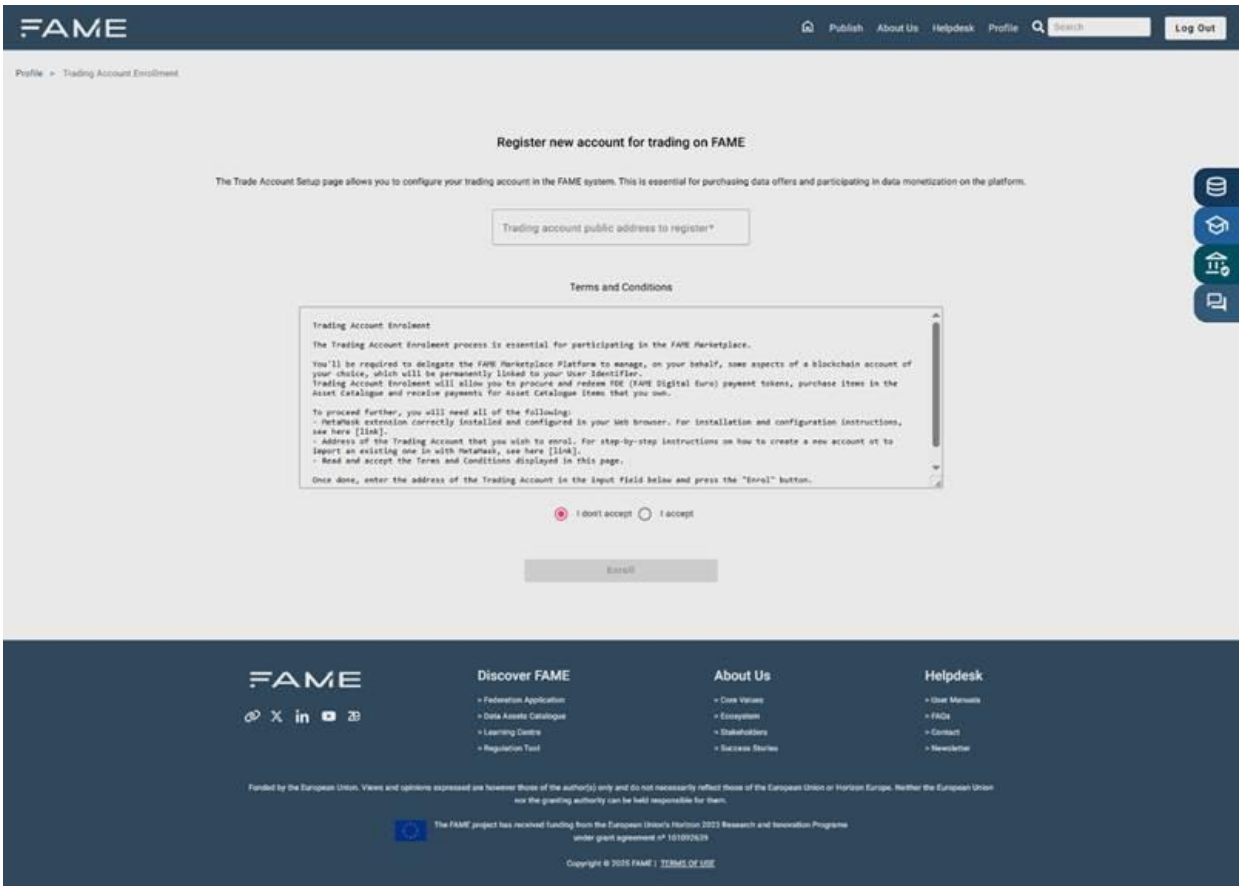


Figure 28 : Trading Account Enrollment page

Figure 29 illustrates the **Publish page** that is accessible only by logged in end-users, where the latter can initiate the data assets' indexing process, by providing detailed information on the data asset to be indexed to the FAME Data Marketplace. The initial step in this page prompts the end-users to connect to their Metamask Wallet, since this is a prerequisite for initiating the publishing process. A

secondary area also exists to offer guidance for end-users who have not installed yet the Metamask Wallet extension to their web browsers, which is crucial to enable blockchain-based data assets publishment on the FAME Data Marketplace, establishing secure interaction between the user and the platform. To this part, a small text link exists that leads to the detailed step-by-step manual (located in the About Us page – Figure 23) for the end-users that may need any further assistance on this procedure.

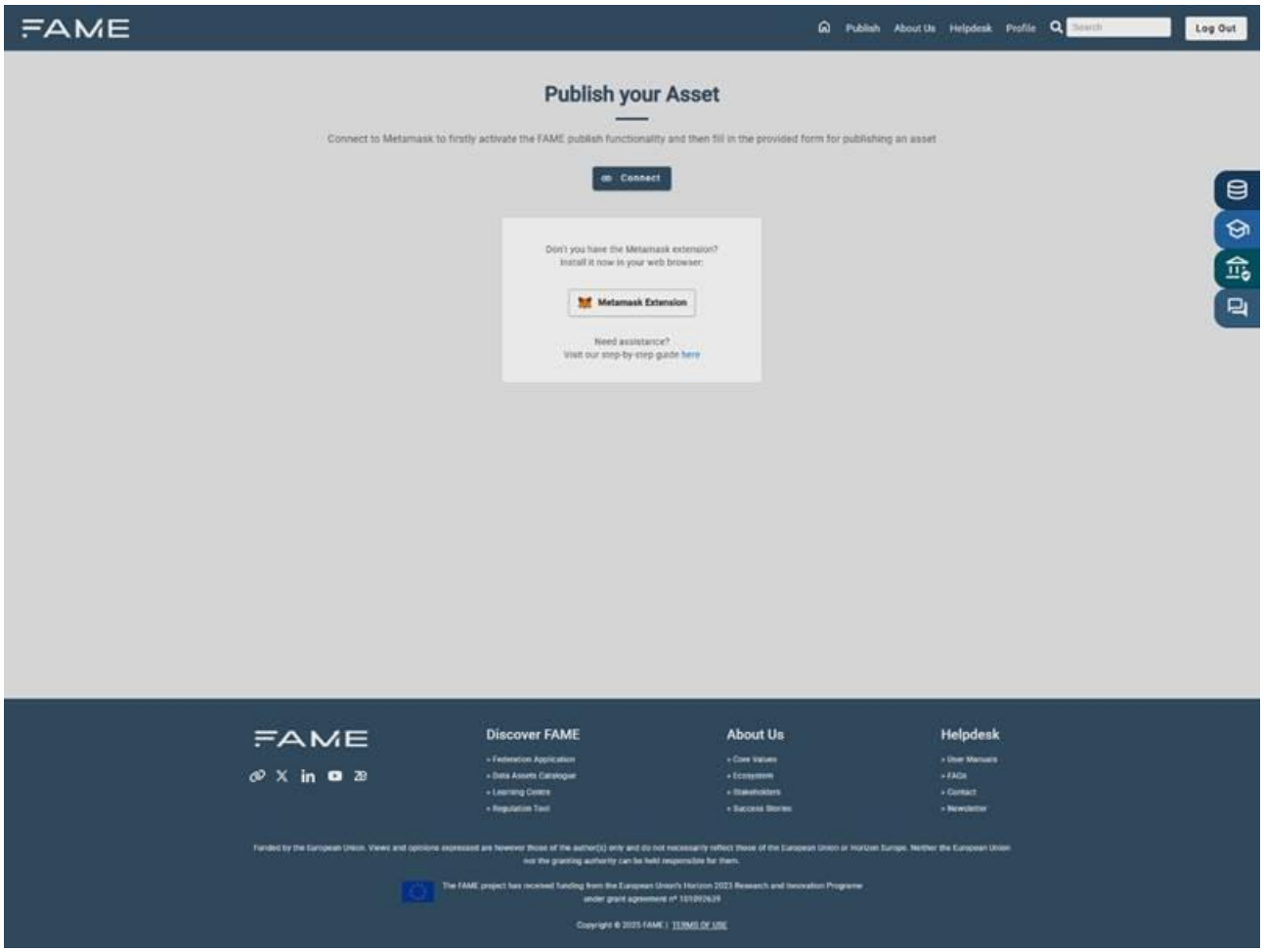


Figure 29 : Asset Publishment page (1)

After successfully logging in and connecting to their Metamask Wallet, end-users have access to the rest of the page's content. Figure 30 presents this content of the Publish page, showcasing the data asset's submission form that appears, so as to enable end-users to contribute new content to the FAME Data Marketplace. The form includes various fields, including the title, type, description, logo, web site, standards, and associated content delivery service of the data asset to be published. Furthermore, information icons and warning messages are integrated into the form to assist end-users by clarifying each field's requirements and alerting them for any potential issues. Positioned centrally, the form reflects a clean and structured layout, guiding end-users through the publishing process to finally either "Discard" or "Confirm" the provided information.

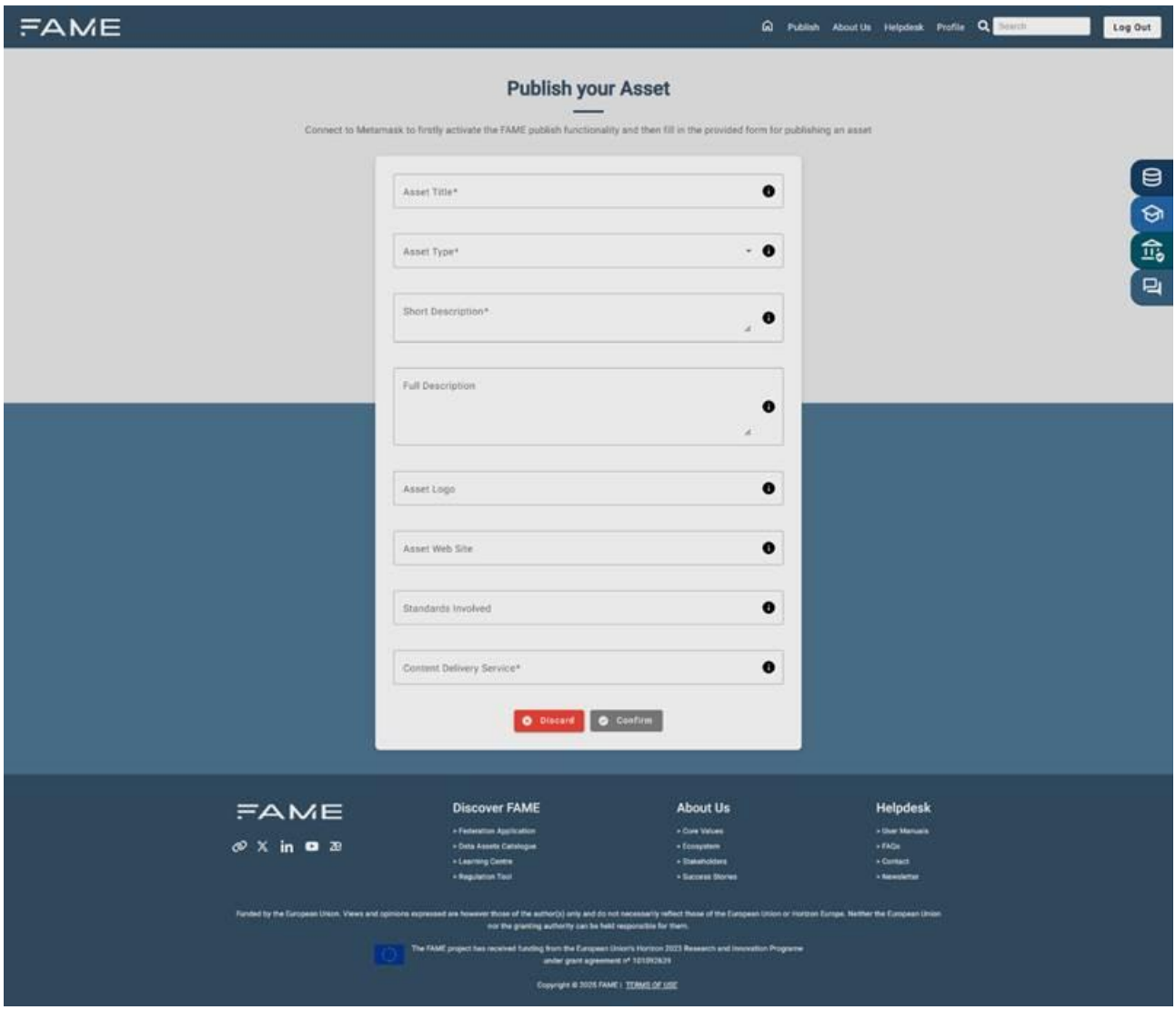


Figure 30 : Asset Publishment page (2)

Once the form is completed and submitted, it triggers an on-screen confirmation with the newly indexed asset's ID, and a direct link through the "Go to entry" button to view the asset entry in the FDAC, reinforcing transparency and immediate feedback for the data asset's creator (Figure 31).

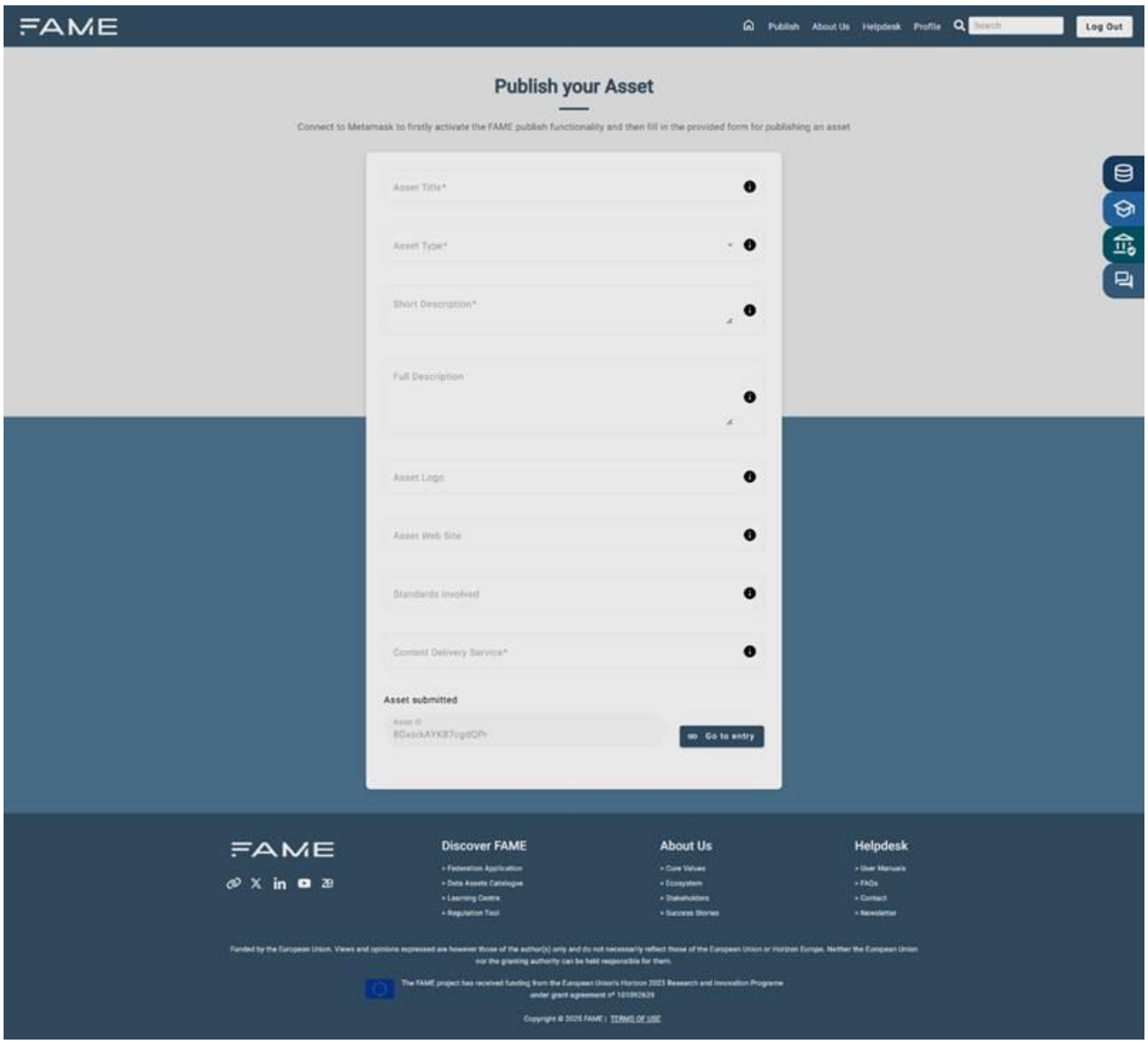


Figure 31 :Asset Publishment page (3)

Figure 32 depicts the **Data Assets Catalogue (FDAC) page** that is fully accessible by logged in end-users, displaying the main catalogue of all the available data assets within the FAME Data Marketplace. On the left panel, end-users can filter the contents of the FDAC by name, type (e.g., dataset, model, documentation) and date, where actionable "Apply" and "Clear" buttons exist for either applying such actions based on the chosen criteria or cleaning any specified filter, correspondingly. On the right part of the page, a grid presents each data asset as a tile with key metadata and a "Certified Federation" badge. To this end, it should be mentioned that non-logged in end-users can also access this page, with the difference that the FDAC will depict only the data assets that have been assigned with a "public" policy access type, thus being fully visible to any external end-user. As a final note for this page, the filtering system is designed to support future enhancements, including the ability to select price ranges for the data assets, filter them by organization, and choose among a variety of supported business models, referring to Pay-as-you-go, Pay-as-you-use and Subscription.

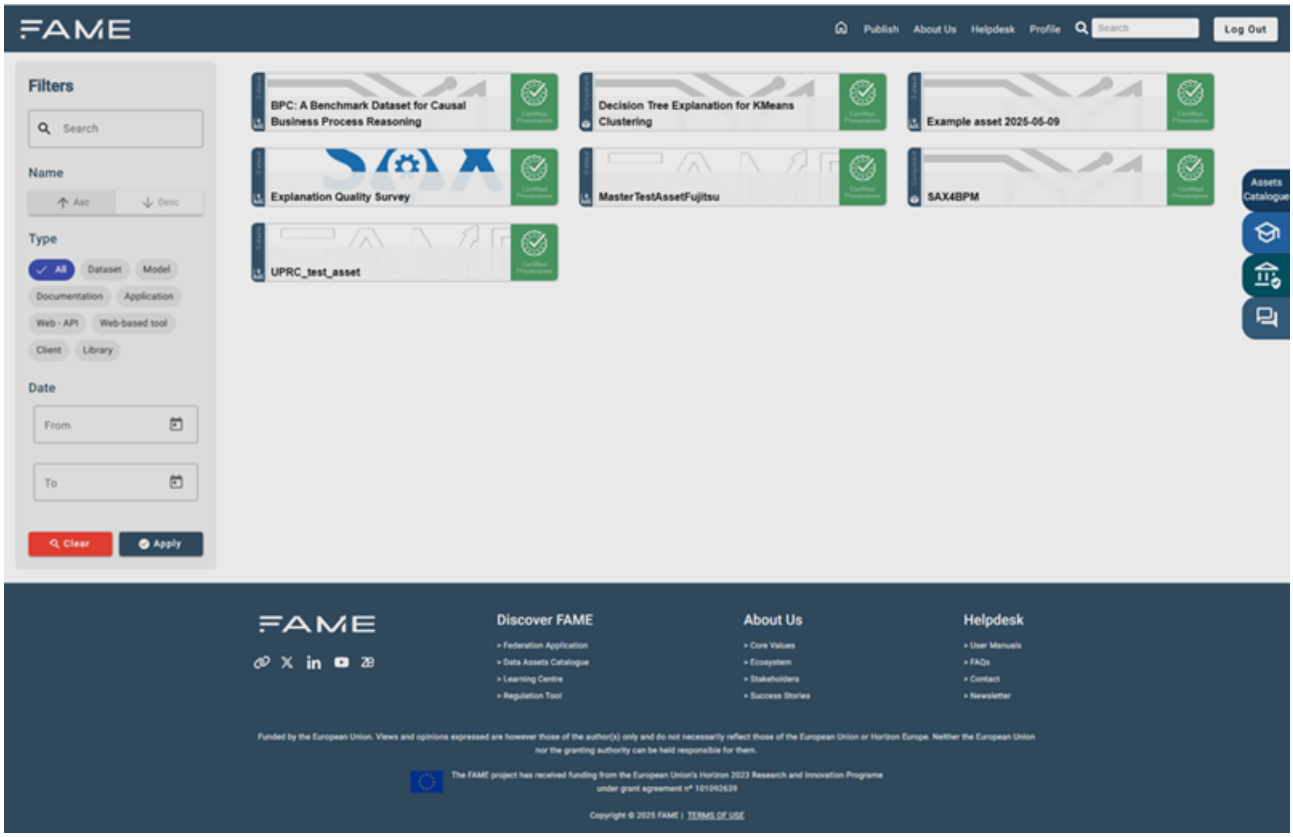


Figure 32 : Data Assets Catalogue page

Figure 33 includes the **Asset Details page** that is fully accessible by logged in end-users, which includes all the details of each data asset. Upon selecting a data asset from the FDAC page, end-users are redirected to the chosen asset's details view, which includes four (4) main collapsible sections, referring to: (i) characterization, which shows core metadata of the asset such as organization, description, and type, (ii) description, which offers a brief text summary of the asset, (iii) offerings, which lists any associated offerings, each linked with a specific price and business model to be applied in case that an end-user decides to "Buy" this offering, and (iv) media gallery, which displays related images or visual references of the asset. At the top of the page, a breadcrumb navigation also exists to help end-users trace their current path within the FDAC. To this end, it should be mentioned that non-logged in end-users can also access this page, with the difference that they only see public assets, whereas logged-in users additionally see assets belonging to their organization.

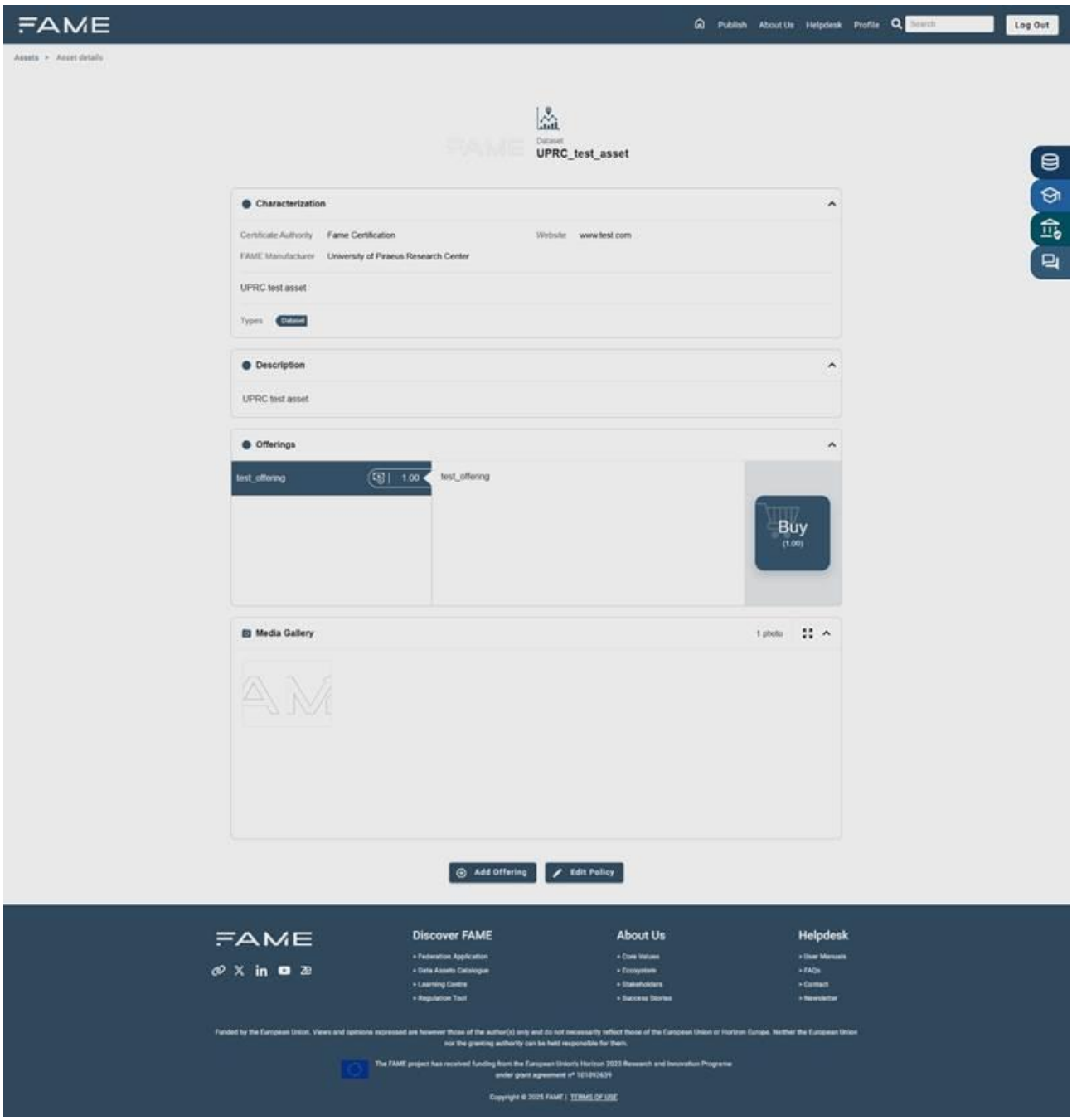


Figure 33 : Asset Details page

Figure 34 displays the **Offering Details page** that is accessible only by logged in end-users, which appears when an end-user clicks to "Buy" a specific offering from the Asset Details page. At the top of the page, a breadcrumb navigation is provided to help end-users trace their current path within the FDAC as in the case of the previous page. In the rest of the page's content, end-users can locate key information about the chosen data asset's offering, referring to the offering ID and the linked data asset, along with details about the offering's business model, duration, and price in FDE units. It also includes relevant terms of use such as limitations, service level agreements (SLA), and terms and conditions of the offering, whereas a prominently placed "Purchase" button allows end-users to proceed with the transaction of purchasing this specific offering. To this end, it should be noted that the end-users must have already enrolled a blockchain account to the FAME Data Marketplace and be connected to it to successfully complete the purchase process.

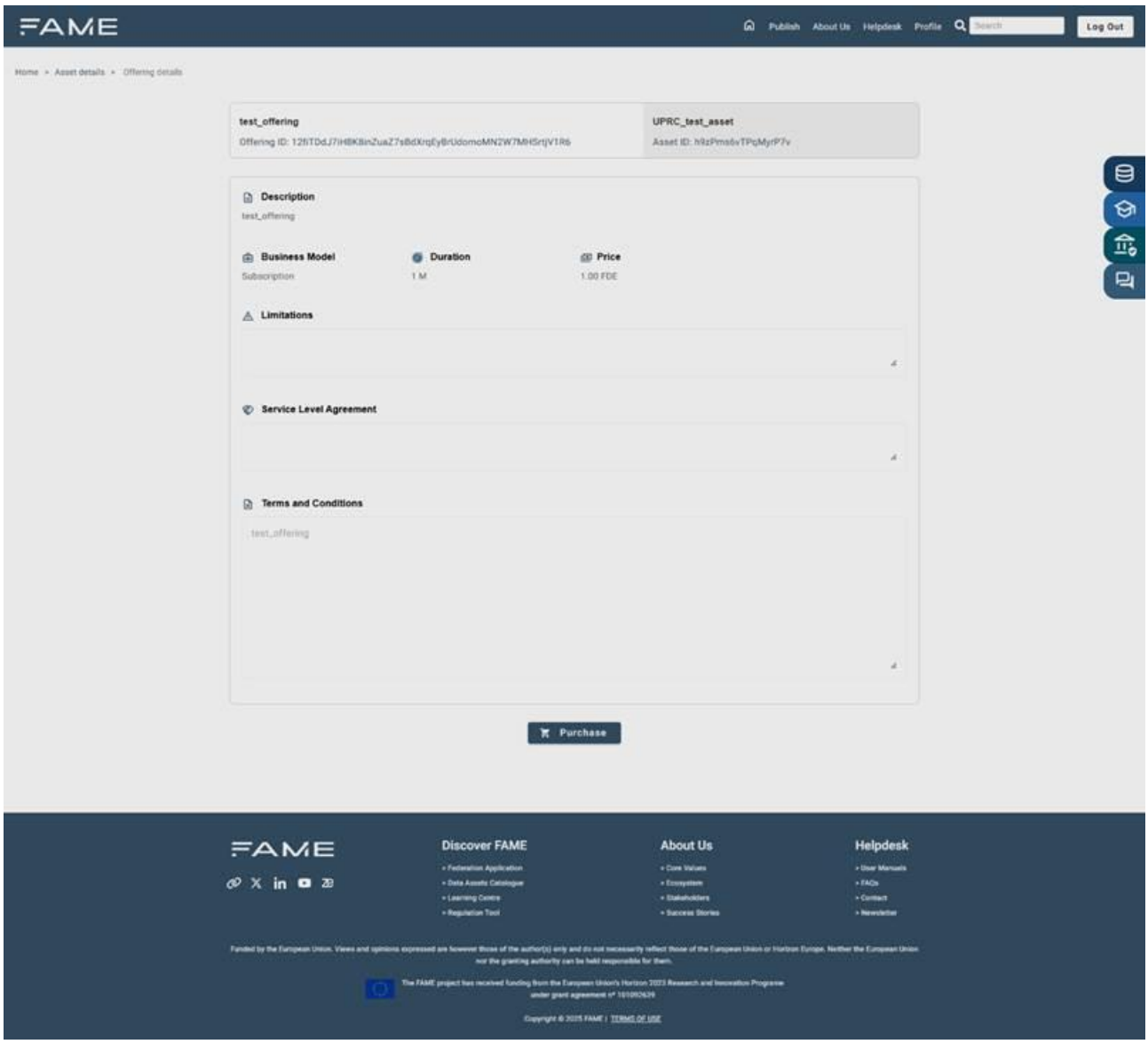


Figure 34 : Offering Details page (1)

Once the transaction is processed, a confirmation message appears, showing the transaction ID and providing a “Copy” button to make it easy for the end-users to track their purchase details directly within the marketplace (Figure 35).

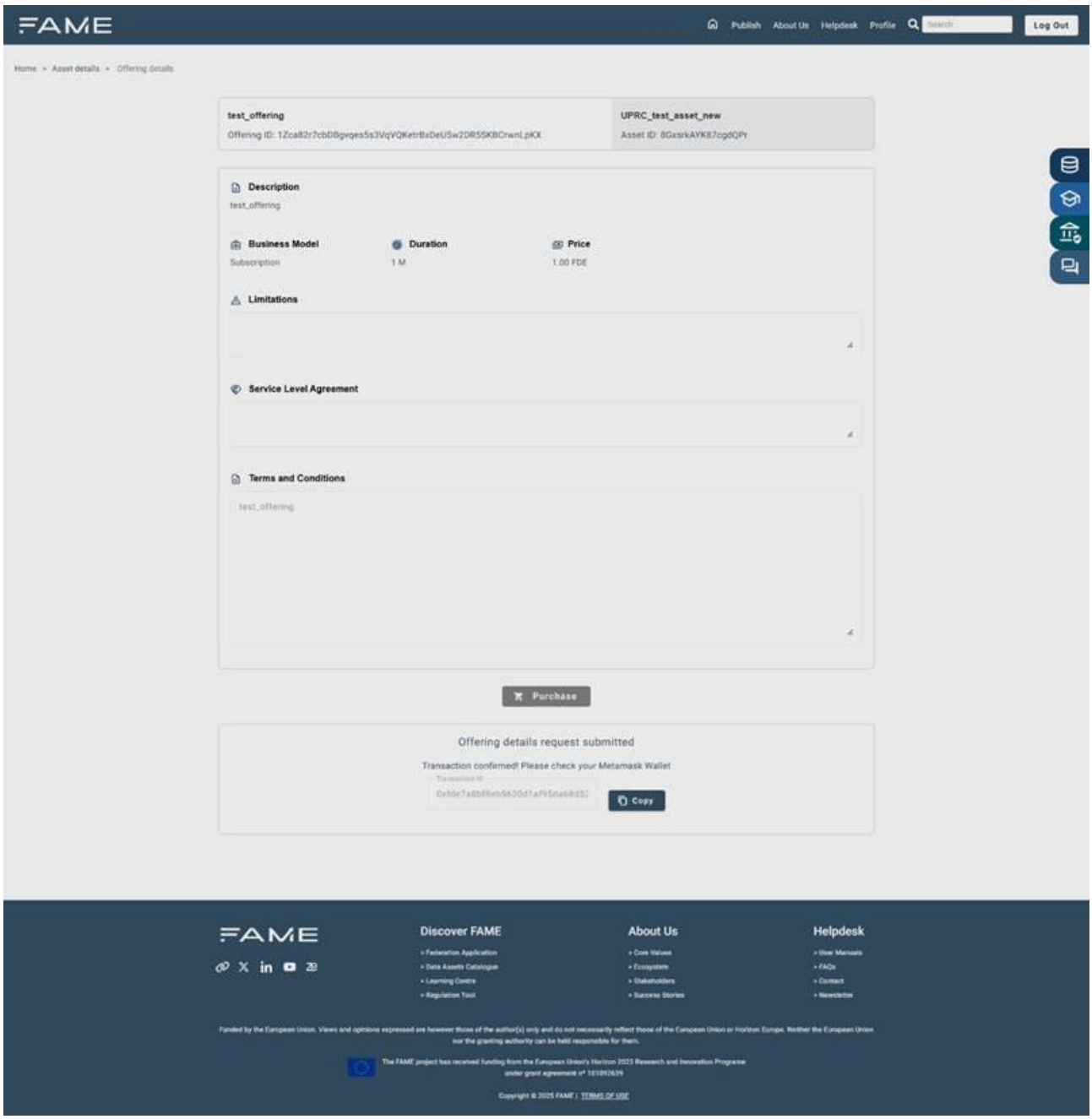


Figure 35 : Offering Details page (2)

Figure 36 illustrates the **Offering Definition page** that is accessible only by logged in end-users, which is used to create a new offering upon a specific data asset. A breadcrumb navigation bar at the top of the page indicates the current location of the end-users within the FDAC, allowing them to easily trace back to previous pages. The rest of the page contains an input form, where, for each new offering to be added to an asset, end-users are required to complete some related mandatory fields such as the offering title, summary, unit type, unit price, and terms and conditions. Additionally, the form provides optional fields for specifying limitations, SLA, and internal details like the trading account and instructions to the content delivery service. The form also incorporates information icons next to all the input fields to provide quick guidance, while inline warnings appear to notify end-users about missing or invalid data entries. At the bottom of the page, two (2) specific action buttons allow end-users to either “Discard” or “Confirm” the entry and the provided data for the offering to be created. Additionally, a dedicated “Pricing Advisor” button is available alongside the unit price field to help end-users determine suitable pricing strategies based on the provided information.

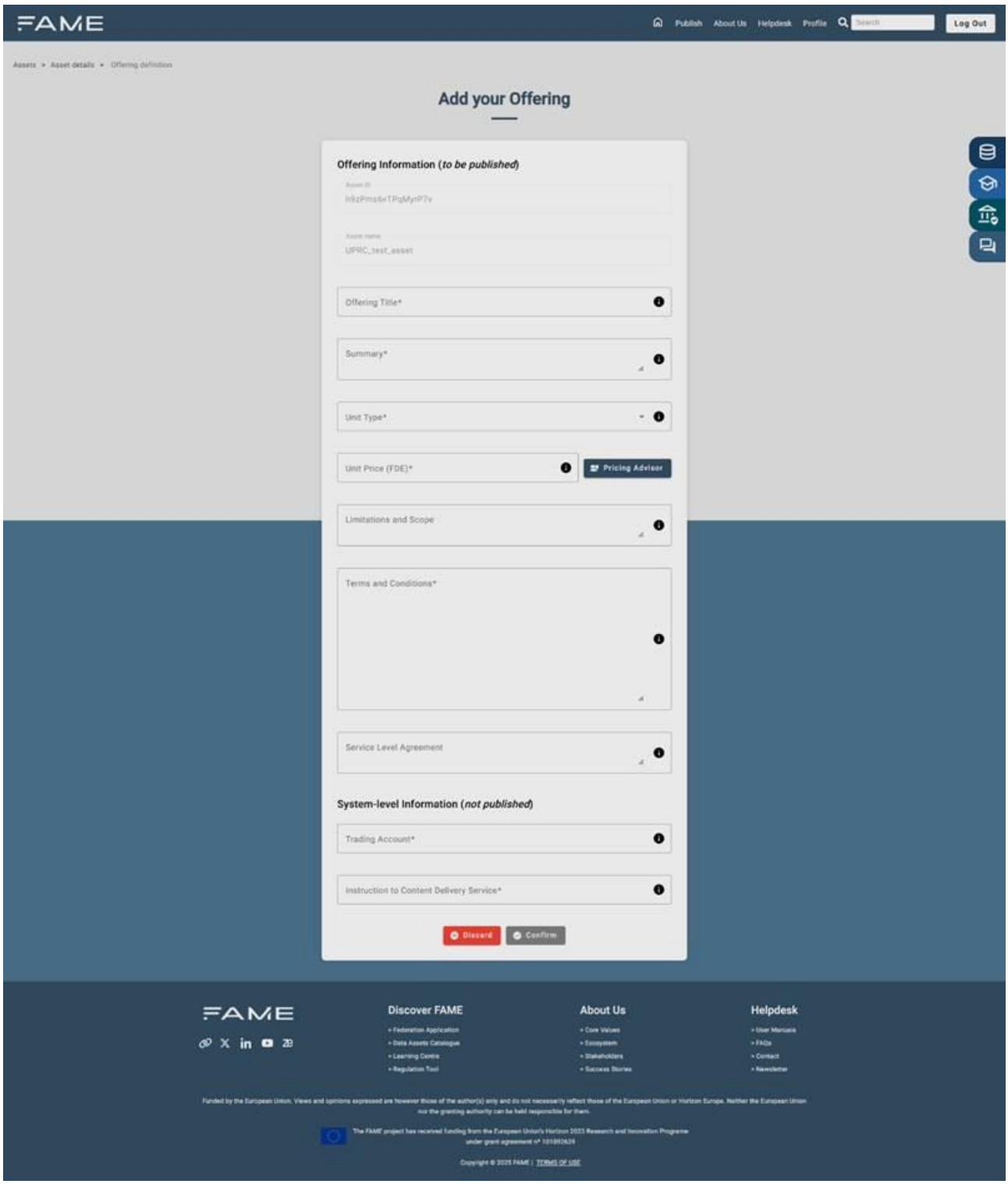


Figure 36 : Offering Definition page (1)

Once the form is completed and submitted, this interface confirms the offering creation by displaying an offering ID together with a direct “Go to entry” button to access and view the entry in the respective data asset in the FDAC, ensuring that end-users have immediate feedback and easy access to manage their provided offerings (Figure 37).

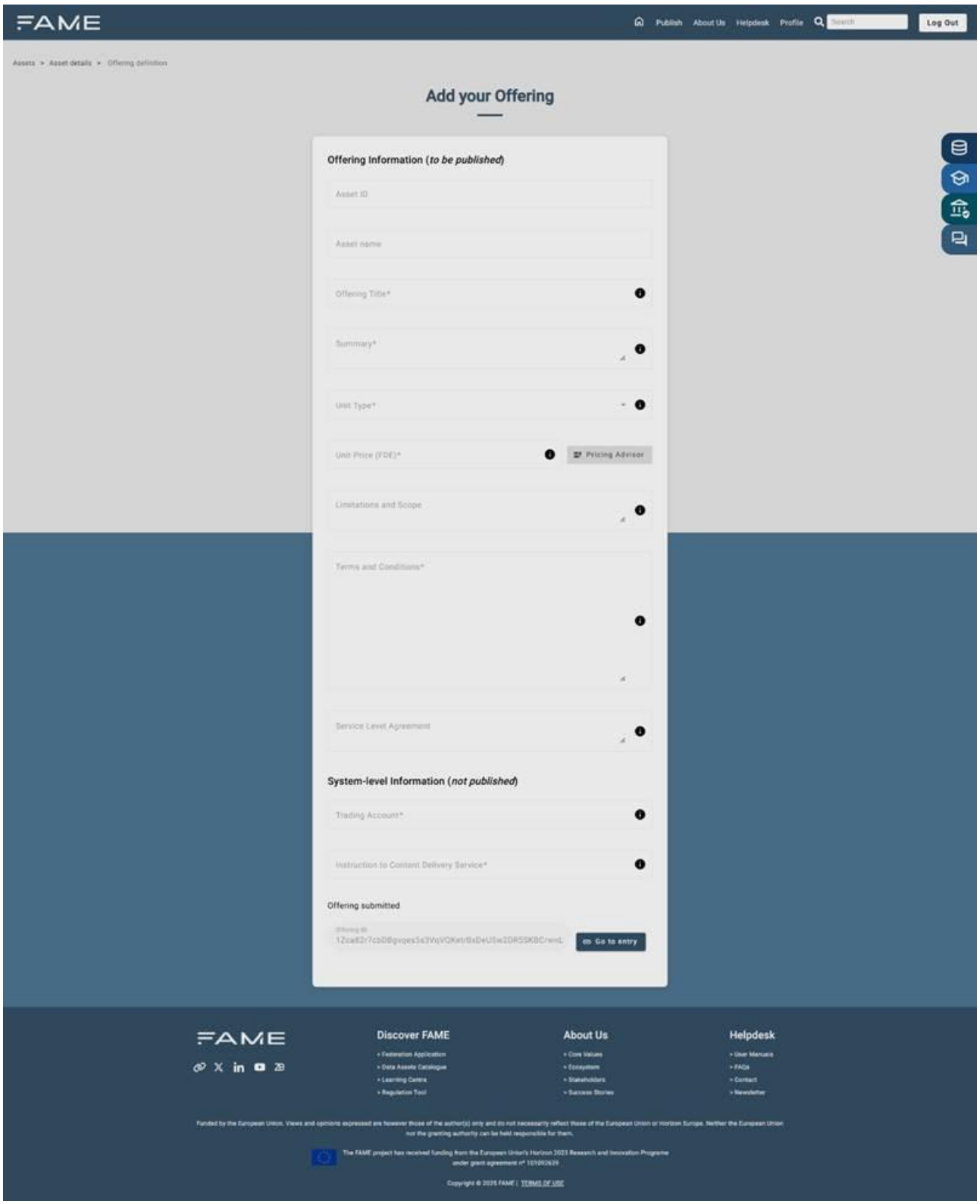


Figure 37 : Offering Definition page (2)

Figure 38 includes the **Asset Policy Manager page** that is accessible only by logged in end-users, where the latter can set up the access type that will be applied upon each published data asset in the marketplace. End-users are able to define access levels for a given asset, offering a variety of options, referring to public, restricted, confidential and onboarded access, being also able to configure policy rules using a set of fields and conditional logic, including parameters like field name, operator, and case. Once the configuration is complete, the defined rules can be saved through the “Save Rule” button, and immediately applied to manage the corresponding asset’s visibility and access control.

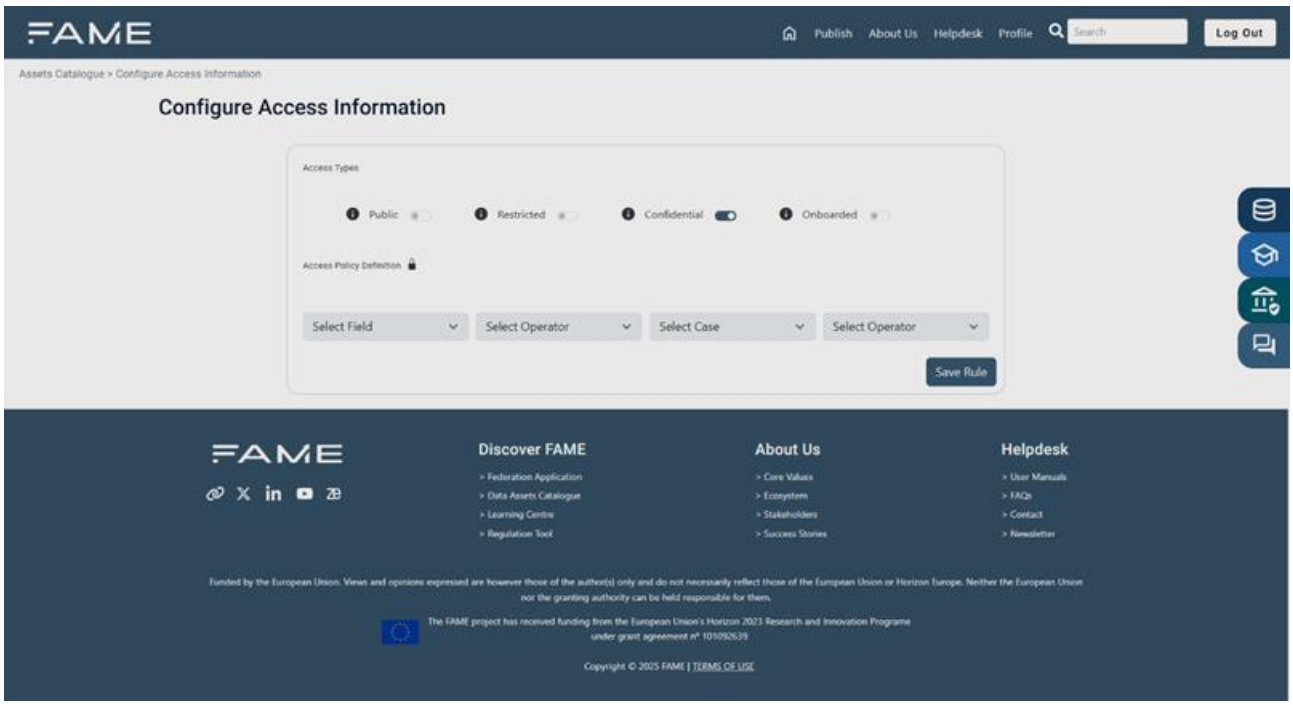


Figure 38 : Asset Policy Manager page

Figure 39 presents the **Learning Centre page** that is accessible by everyone, where end-users can explore all the available courses. Each course card displays the title, the number of chapters and the presence of quizzes, videos and PDFs per course. A filter panel on the left allows end-users to sort the content by name or date, improving navigation and discovery of the relevant material. The filter panel also features dedicated "Clear" and "Apply" buttons, enabling end-users to reset or execute their selected filters effectively.

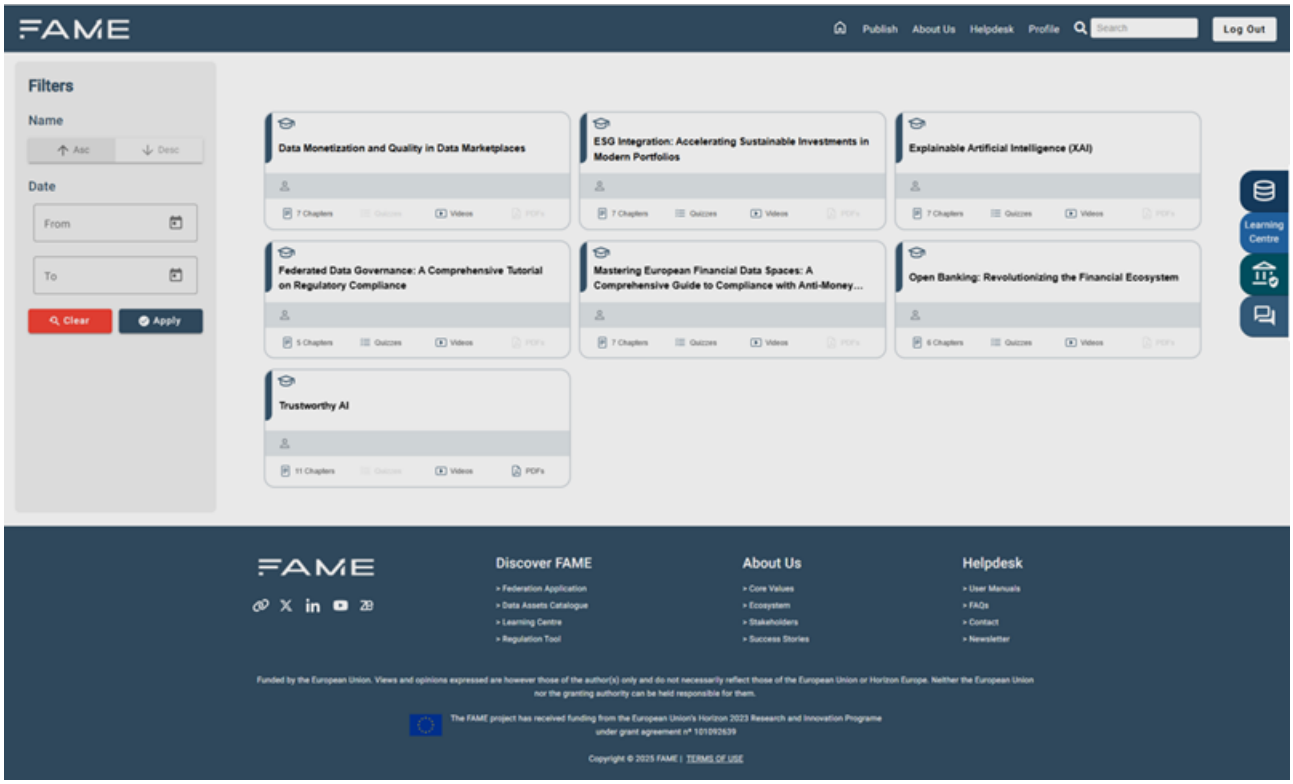


Figure 39 : Learning Centre page

Figure 40 depicts the **Course Details page** that is accessible by everyone, where the details of each course can be located. A breadcrumb navigation bar at the top of the page indicates the current location of the end-users within the Learning Center, allowing them to easily trace back to the previous page. In the rest of the page's content, end-users can access a specific course via the Learning Centre page, and see a course's detailed view that includes the lesson's structure, including all chapter titles and a "Start" button to initiate the demonstration of the course. Metadata such as cost, language, and certificate availability is included at the top, helping end-users to decide before engaging with the course.

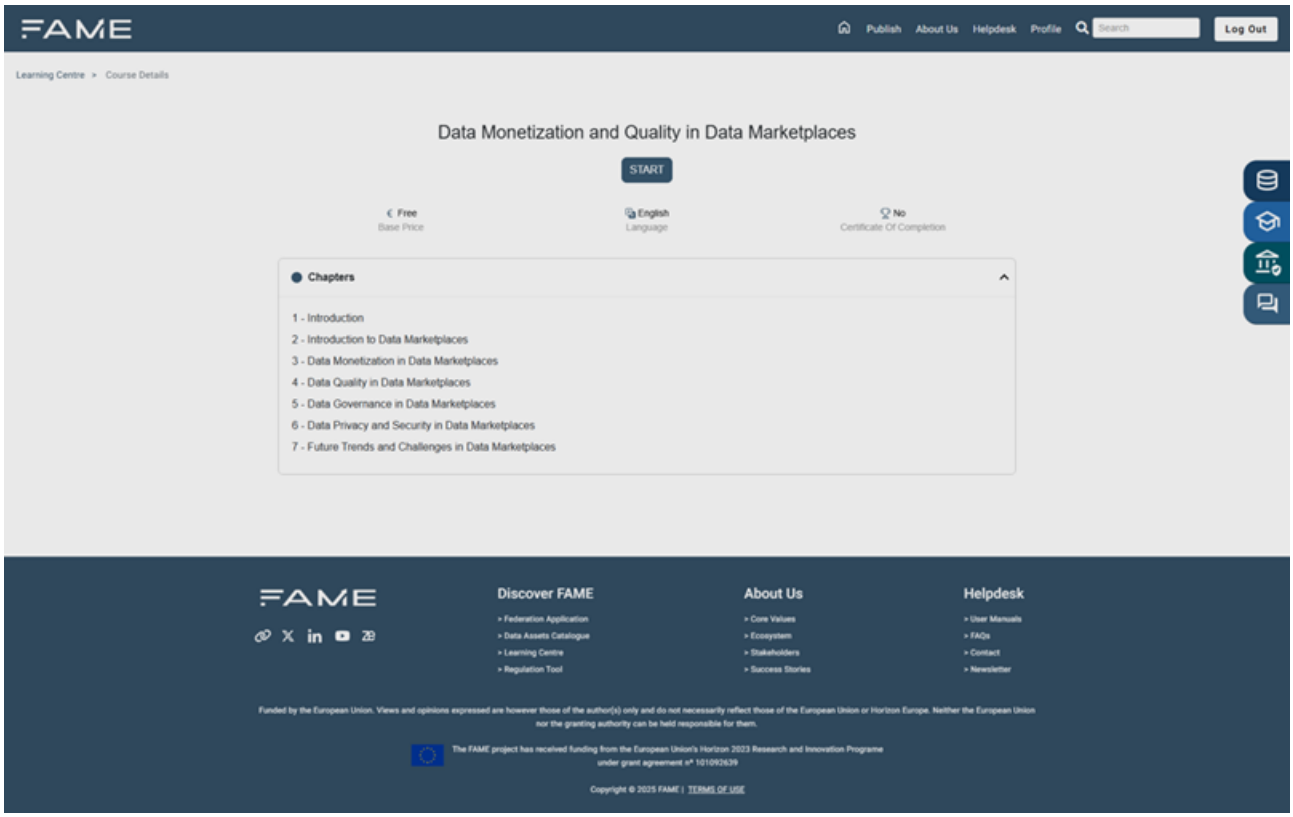


Figure 40 : Course Details page

Finally, in Figure 41, the **Terms and Conditions page** is presented, featuring a clear hierarchical structure that guides participants through the marketplace's rules and operational framework. At the top, the title distinctly identifies the purpose of the page, while subsequent sections comprehensively cover the scope, key definitions, access requirements, subscription details, acceptable use policies, and user rights. Each topic is organized under numbered headings to ensure ease of reading and reference, whilst embedded hyperlinks, such as the one directing to the "FAME Onboarding Procedure" provide convenient navigation for end-users seeking further details.

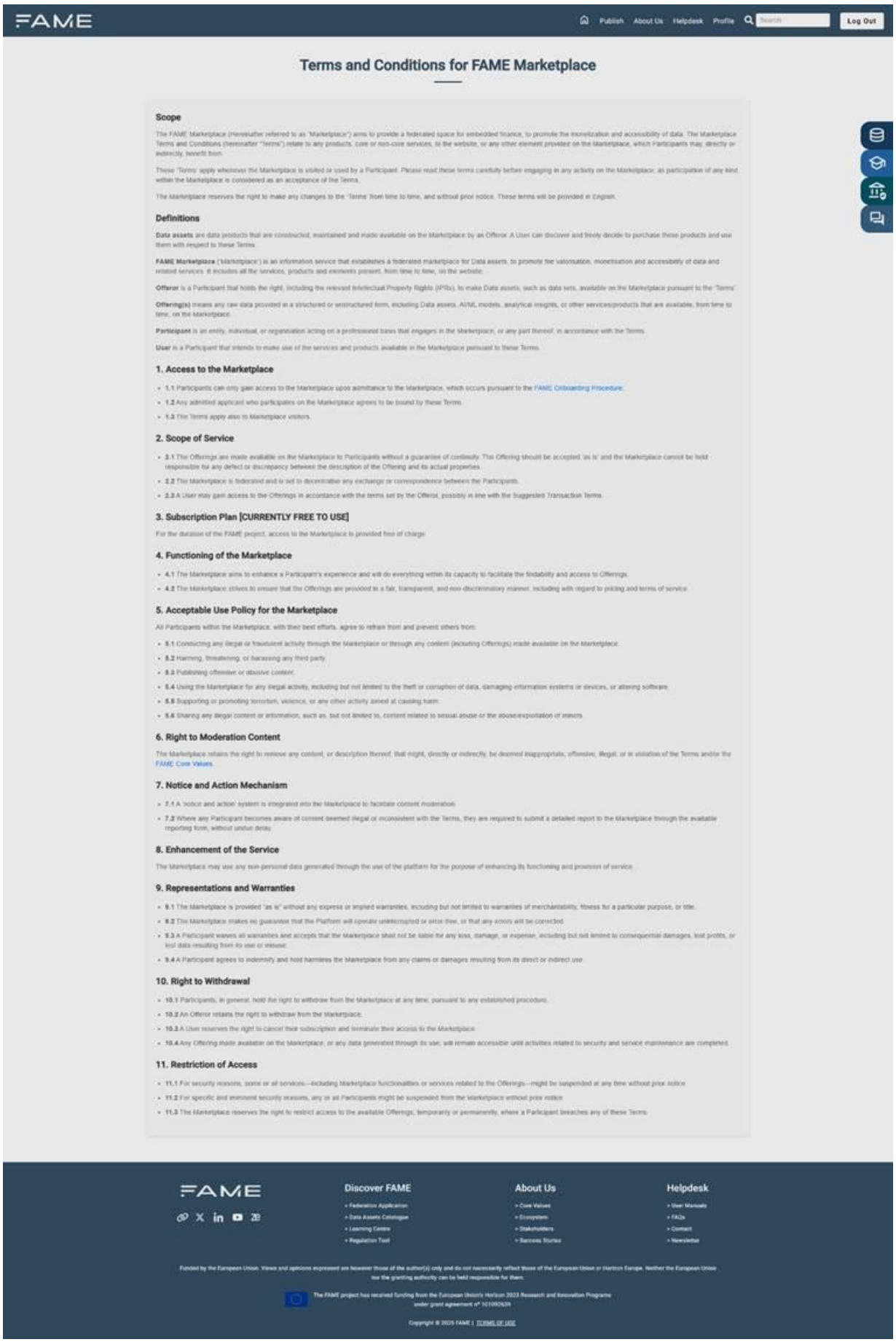


Figure 41 : Terms and Conditions page

3.3 Integration of Frontend & Backend Services

3.3.0 Integration Methodology

The FAME Dashboard has put in place a RESTful API communication among the FAME platform's frontend and backend parts, leveraging the HTTP protocol to facilitate communication between those two (2) layers. Exploiting RESTful APIs that are characterized by their stateless operation and reliance on standard HTTP methods (GET, POST, PUT, DELETE), a flexible way is supported to fetch, create, update, and delete data from the platform. This allows the frontend part of the FAME platform to dynamically display and update information based on the user's interactions and the backend data changes.

In addition to RESTful communication, the FAME Dashboard also supports iframe integration, which enables the embedding of external tools, visualizations, and services directly within the constructed UIs. This is particularly useful for integrating modular content such as dashboards and demonstration applications that operate as part of or alongside the FAME platform. The use of iframes allows external systems to be visually and functionally included without disrupting the main Dashboard's workflow or requiring redirection to a separate application. Furthermore, the Dashboard can communicate with external pages hosted within the extended FAME platform ecosystem. These include auxiliary web applications, micro-frontends, or documentation portals that are integrated under common authentication and routing policies. This capability ensures smooth user experiences across distributed services and facilitates the reuse of external content in a secure and consistent manner, leveraging mechanisms such as Angular routing, postMessage for cross-frame communication, or CORS-compliant HTTP exchanges.

A concise guide aligned with the Angular framework and the integration of the two (2) layers (frontend, backend) through RESTful APIs is provided below:

- **API**: Defines rules and protocols for the software applications/services/components to interact.
- **HTTP**: Transfers data over the web, enabling communication between clients (e.g., browsers) and servers.
- **Endpoints**: Represent specific resources on the server, used in RESTful APIs to perform actions like data retrieval, creation, update, and deletion.
- **HTTP Methods**: Various methods used in RESTful APIs for different operations:
    - GET: Retrieve data from the server.
    - POST: Send data to create new records.
    - PUT: Update existing records.
    - DELETE: Remove records from the server.

Integration through RESTful APIs

The complete process for successfully accomplishing the integration with RESTful APIs in the context of FAME, is outlined below:

- *Step 1*: Setting up backend API
    - Choose a backend technology stack such as Node.js with Express or Django for Python, ensuring compatibility with Angular.
    - Define routes and endpoints for the API, following RESTful principles.
    - Implement business logic and database operations within the API endpoints.

- *Step 2*: Sending requests from Angular frontend
  - Utilize Angular's built-in HttpClient module or libraries like Axios to send HTTP requests from the frontend.
  - Construct the appropriate request method (GET, POST, PUT, DELETE) and headers.
  - Include necessary data in the request body or as query parameters.
- *Step 3*: Handling responses in Angular frontend
  - Subscribe to the observable returned by HttpClient methods to receive responses from the backend.
  - Parse and process the response data received from the backend.
  - Update the Angular components and UI based on the response data.
  - Implement error handling to manage any potential errors or exceptions.

Integration through iframe embedding

The process for successfully embedding within the FAME Dashboard interface external tools, dashboards, or modular services, is provided below:

Step 1: Embedding external content via iframes (when applicable)
Use Angular components to securely embed external tools and views using <iframe> tags.
Configure appropriate sandbox and CSP headers to mitigate security risks.
If needed, enable two-way communication using window.postMessage to exchange messages between the embedded content and the Angular app.

Integration with external platform interfaces

The process for effectively linking to or incorporating other web applications that are part of the broader FAME ecosystem, is described below:

Step 1: Communicating with external platform pages
Utilize Angular Router or dynamic redirection to link to external web applications hosted within the FAME ecosystem.
Ensure proper authentication (e.g., token sharing or SSO) and CORS headers are configured for secure interaction.
Consider embedding these pages via iframe or opening them in new tabs, depending on the use case and security constraints.

As a final note for the integration approach, the FAME Dashboard embraces an API-First approach [28] to foster collaboration and streamline development with the various underlying backend parts of the platform. This means that developers prioritized the design and development of the APIs at the beginning of the project. This strategy ensures that the APIs are robust, well-documented, and ready to support various frontend applications (web, mobile, etc.) from the beginning. By focusing on the API-First approach, FAME has encouraged a more modular design, allowing teams to work in parallel and making the platform more adaptable to future requirements. Thus, a more collaborative approach has been put into practice, where the frontend and backend teams work together to design the APIs. Thus, all the implementations have taken place simultaneously so far, as the frontend teams could use any mock backend for their development work.

For efficiently achieving the integration between most of the frontend and the backend parts of the FAME platform, the abovementioned integration methodology has been followed, exploiting RESTful APIs. More specifically, the steps that have been followed are depicted below:

Integration through RESTful APIs

- *Step 1*: Start with API design
  - Before diving into the frontend development, developers should define their API requirements based on the functionalities they aim to support. This includes identifying the resources, actions, and data formats needed.
  - In this phase, the developers must provide the postman exported APIs' collection [29], and provide a related readme file with a concrete (local) installation guide for each API, following the below example:
    - Description of service endpoint: Count Items in a Specific Time Range
    - API Documentation: https://example.com/api-docs/#/Items/getItemsCount
    - HTTP Method: GET
    - API Endpoint: https://api.example.com/items/count
    - Query Parameters:
      - (i) time_from: Required. Start of the time interval (in seconds)
      - (ii) time_to: Required. End of the time interval (in seconds)
    - Example Request:
    - https://api.example.com/items/count?time_from=10&time_to=16
    - Example Response: {
      "status_code": 200,
      "status": "success",
      "count": 150
      }
- *Step 2*: Ensure proper documentation and versioning
  - It must be ensured that all APIs are thoroughly documented and versioned. This aids in maintenance and scalability, as new features can be added with minimal impact on existing functionalities.
  - In this step, the developers must document and version their APIs exploiting one of the following (or similar) tools:
    - SwaggerHub [30]: An API editor compliant with Swagger, also supporting AsyncAPI.
    - Online Swagger editor [31]: A browser-based editor for creating and testing Swagger/OpenAPI specifications.
- *Step 3*: Develop in parallel
  - With a clear documentation API (based on *Step 1* and *Step 2*), frontend and backend teams can work simultaneously, speeding up the development process. The frontend teams can use mock APIs or API stubs to start the development even before the backend is fully implemented.
- *Step 4*: Perform iterative testing
  - Regular tests must occur to validate the integration between the frontend and backend using the RESTful APIs to catch and fix issues early in the development cycle.
  - Establish a feedback loop between the frontend and backend teams to continuously refine the APIs based on real-world usage and frontend requirements.
- *Step 5*: Communicate with the integration team
  - It is recommended that developers subscribe to the Slack dedicated channel for directly communicating with the rest of the integration development teams, so as to solve any raised issues/technical points.

Integration through iframe embedding

For specific modules such as the FDAC and the Learning Centre that operate independently but are visually embedded within the FAME Dashboard, the integration has followed a secure iframe strategy at the component level:

Step 1: Angular's component architecture incorporates <iframe> elements dynamically, binding their source attributes to trusted module URLs retrieved from secure configuration services or environment settings.
Step 2: Content Security Policy (CSP) and X-Frame-Options headers are enforced both at the server and application level to restrict framing to approved origins and mitigate clickjacking or injection risks.
Step 3: When interactive scenarios require bidirectional communication (e.g., cross-frame messaging for data reloads or UI state synchronization), structured window.postMessage channels are implemented, with explicit message type validation to safeguard against unsolicited or malformed payloads.

Integration with external platform interfaces

In cases like the APM tool or additional FAME ecosystem applications (i.e., Federation Application, Regulation Tool), integration is achieved via external routing and interface linkage, designed for flexibility and controlled cross-service authentication:

Step 1: The Angular router is configured with dynamic route guards and resolver patterns, ensuring that authentication tokens (JWT or SSO session identifiers) are available prior to redirection.
Step 2: CORS policies are carefully tailored on backend services to accept authorized cross-origin requests initiated by the dashboard, aligning with the platform's federated security model.

3.3.1 Integration of FAME Federation Application & Backend Services

The integration between the FAME Federation Governance Application (FFGA) and the broader FAME backend services represents a critical architectural component that enables seamless federation management while maintaining the independence and transparency of the governance process. This integration ensures that federation decisions made through the FFGA are properly reflected across the entire FAME ecosystem.

Integration Architecture Overview

The FFGA integration follows the same RESTful API principles established for other FAME components, implementing secure communication channels with key backend services while maintaining the federated governance model's autonomy. The integration architecture supports both real-time communication for immediate actions and asynchronous processing for complex federation workflows.

Primary Integration Points:

1. **Operational Governance (GOV) Module**: Direct integration for federation member management and user onboarding
2. **Authentication and Authorization Infrastructure (AAI)**: Credential exchange and identity verification
3. **FAME Marketplace Platform (FMAP)**: User registration and credential provisioning

Integration with GOV Module

The most significant integration occurs between the FFGA and the Operational Governance (GOV) module, as documented in D4.3. This integration implements the federation management workflows described in Section 2.2.3.

Federation Member Registration:

```
POST /gov/v1.0/members
Content-Type: application/json
Authorization: Bearer {federation-token}

{
  "org": legal_entity,
  "type": organization.org_type,
  "rep": rep_data,
  "oa": did / ao_id
}
```

Integration Workflow:

1. **Application Approval**: When the FFGA voting process concludes with approval, the FFGA automatically triggers federation member registration in the GOV module, attaching either the did (self-OA) or oa_id (delegated-OA) as appropriate.
2. **Credential Request**: The FFGA obtains Personal Identifier (PID) from the GOV module for approved organizations
3. **Status Synchronization**: Real-time synchronization of federation status between FFGA and GOV module databases

Error Handling: The integration implements comprehensive error handling to manage cases where GOV module operations fail, ensuring data consistency and enabling manual intervention when necessary.

Integration with GOV Module for User Management

Once federation is approved, the federated organization gains the capability to onboard users to the FAME platform **only if it selected FAME as its Onboarding Authority**. In all other cases, user onboarding is performed through the OA's own platform or tooling, and the FFGA simply displays the OA reference.

Post-Federation User Onboarding Capabilities

OA Scenario	User invitation via FFGA	Responsible System
FAME selected as OA	**Yes** – the federation portal exposes the "Invite User" functions and issues Verifiable Onboarding Credentials (VOC) directly	FFGA → GOV User Management API

Another OA selected	No – the FFGA shows a banner: "User onboarding is handled by **{Selected OA}**."	Selected OA's platform
Applicant is self-OA (provided DID)	No – the organization must use its own onboarding tools; FFGA stores the DID for future reference	Applicant's own OA tooling

User Invitation API Integration:

```
POST /api/v1.0/users
Content-Type: application/json
Authorization: Bearer {admin-token}

{
  "rol": "USR",
  "cty": "IT",
  "aff": "ORG_ID",
  "fname": "John",
  "lname": "Doe",
  "email": "john.doe@organization.com",
  "phone": "+39123456789"
}
```

The FFGA maintains integration endpoints to query active authorities and support the user management workflow, ensuring that only properly federated organizations can invite users to the platform.

Asynchronous Processing and Workflow Management

The FFGA implements asynchronous processing capabilities to handle complex federation workflows that involve multiple backend services.

Workflow Orchestration:

- **Background Tasks**: Use of APScheduler for managing time-based operations (voting periods, credential expiration)
- **Status Tracking**: Comprehensive logging and status tracking across all integration points

Data Consistency and Synchronization

To maintain data consistency across the federated architecture, the FFGA implements several synchronization mechanisms:

Event-Driven Updates: The FFGA publishes federation events (application submitted, voting completed, member approved) that are consumed by relevant backend services for real-time updates.

Reconciliation Processes: Periodic reconciliation processes ensure that federation membership data remains consistent across all FAME components, identifying and resolving any discrepancies.

Audit Trails: Comprehensive audit logging of all integration activities enables troubleshooting and ensures compliance with governance requirements.

Security and Compliance

The integration architecture implements robust security measures to protect sensitive federation data:

API Security: All inter-service communications use JWT tokens with appropriate scopes and expiration policies **Data Encryption**: Sensitive data is encrypted both in transit and at rest **Access Control**: Role-based access control ensures that only authorized services can access federation management endpoints **Compliance**: Integration processes comply with GDPR requirements and FAME governance policies

Monitoring and Observability

The FFGA integration includes comprehensive monitoring capabilities:

Health Checks: Health checks of all integration endpoints to ensure service availability

Dashboard Visibility: Integration status and metrics are visible through the FAME operational dashboard

This integration architecture ensures that the FFGA operates as a seamless component of the broader FAME ecosystem while maintaining the independence and transparency required for effective federation governance.

3.3.2 Integration of Dashboard & Backend Services

In alignment with the methodology described in Section 3.4.1, the FAME Dashboard integrates seamlessly with various backend services through RESTful API calls, iframe embedding, or external interfaces, depending on the architecture and the nature of each component. More specifically, the integration of the frontend (Dashboard) with each backend service has been implemented as follows:

Federation Application Service is accessed through an external interface. It enables end-users to start and join the FAME Data Marketplace, managing membership and compliance onboarding for secure participation in data exchanges.

FDAC and Learning Centre (LC) Services are integrated via secure iframe embedding. These services function autonomously and are loaded dynamically within the Dashboard UI, allowing end-users to interact with them without navigating away from the main Dashboard.

Trading and Monetization (TM) Service is integrated with the Dashboard via RESTful API endpoints. It comprises two (2) main backend applications:

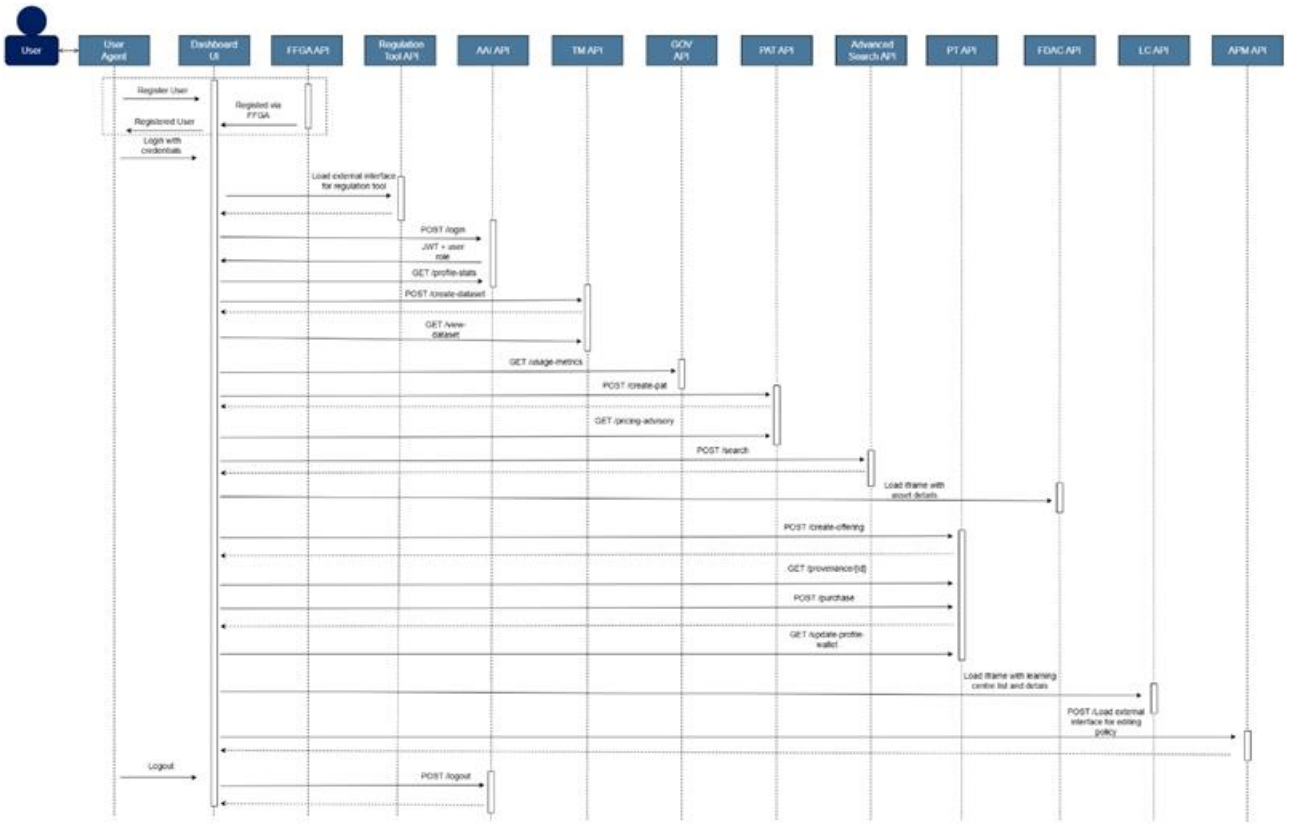
The Private Application, which handles backend-to-backend interactions with smart contracts. It manages offerings, governance transactions, and enforces internal data access policies. This component is not publicly exposed and implements strict access control.

The Public Application, which offers APIs for external users and platforms. It supports governance actions, payment processing, trading history tracking, and data access transactions. All operations are securely exposed with enforced authentication and rate-limiting policies. In addition, the FAME Data Access Processor acts as a reference microservice that validates data access based on on-chain tokens and is typically deployed by data providers to verify entitlement rights before serving content.

Provenance and Tracing (PT) Service integrates with the Dashboard via RESTful APIs, providing detailed metadata on the data assets, projects, and data usage provenance. It contributes to structured visualization within the Dashboard by supporting GET operations, while also including a blockchain connection layer that enables the integration of cryptographically verifiable data lineage.

Operational Governance (GOV) Service exposes RESTful APIs that deliver key operational metrics, such as usage trends, offering analytics and compliance data, allowing the Dashboard to present clear governance insights for transparency and informed decisions.
Authentication and Authorization Infrastructure (AAI) Service enables secure authentication and role-based access control via RESTful API integration. The Dashboard interacts with this module using login endpoints, token-based authentication and role resolution APIs.
Pricing Advisory Tool (PAT) Service offers API endpoints that assist end-users in evaluating pricing strategies upon their shared data assets. It supplies the Dashboard with analytical metrics and recommendations based on market data and offering parameters, contributing to better-aligned and optimized monetization strategies.
Advanced Search Service is integrated via RESTful APIs. It enables keyword-based and filter-based data assets searches through the frontend interface, and returns dynamic results to the end-users.
Asset Policy Manager (APM) Service is accessed through an external interface, with the Dashboard redirecting users to this module to manage and view access policies.
Regulation Tool Service is provided through an external interface. It assists end-users in tracking regulatory and data governance requirements by offering a searchable environment for existing standards, regulations, and deployment examples.

Th image bow presents a sequence diagram depicting the interaction flow between the FAME Dashboard and all the abovementioned backend components, while the rest of this Section provides technical information on the integration performed between the Dashboard and each backend component.



Sequence diagram of FAME Dashboard interactions with backend APIs

Federation Application Service

The Federation Application Service is accessed primarily through an external interface, enabling the successful integration with the Dashboard for secure interconnection with the onboarding process of the FAME Data Marketplace, whereas a dedicated API has been constructed for getting some specific statistics regarding the federated members of the marketplace. The table below summarizes the specific interactions and purpose of this integration.

Table 3 : API endpoints of Federation Application and Dashboard integration.

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
Federation Application	Provides the web application for onboarding external members into the FAME Data Marketplace	https://federation.fame-horizon.eu/	Open	https://gitlab.gftinnovation.eu/fame/ffga-fame-federation-goverance-application/-/tree/dev?ref_type=heads
Federation Statistics API	Provides public statistics about federated organizations and marketplace users	https://federation.fame-horizon.eu/api/public/federation-stats	Open	https://gitlab.gftinnovation.eu/fame/ffga-fame-federation-goverance-application/-/tree/dev?ref_type=heads

Regulation Tool Service

The Regulation Tool Service is provided through an external interface, enabling the successful integration with the Dashboard to facilitate tracking of regulatory and governance requirements. The table below outlines the main integration details and functional interactions.

Table 4 : API endpoints of Regulation Tool and Dashboard integration

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
Regulation Tool	Provides a searchable environment for regulations, standards, and deployment examples to support compliance tracking	https://regulation-tool.fame-horizon.eu/	Open	Not available

FDAC Service

The FDAC Service is integrated into the Dashboard via secure iframe embedding, enabling end-users to explore data assets directly within the main UI. The table below summarizes the integration details and its role in the platform.

Table 5 : API endpoints of FDAC and Dashboard integration.

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
Generate Asset List	Generates the URL with the embedded page with the asset list	GET https://fdac-open.api.dev.unparallel.pt/generateAssetListIframeURL	Open	https://fdac-open.api.dev.unparallel.pt/api-docs/#/FDAC/get_generateAssetListIframeURL
Generate Asset Details	Generates the URL with the embedded page with the asset details info	GET https://fdac-open.api.dev.unparallel.pt/generateAssetDetailsIframeURL/:assetId	Open	https://fdac-open.api.dev.unparallel.pt/api-docs/#/FDAC/get_generateAssetDetailsIframeURL__assetId_
Count	API used to count assets	GET https://fdac-open.api.dev.unparallel.pt/count	Open	https://fdac-open.api.dev.unparallel.pt/api-docs/#/FDAC/get_count
Asset data	Returns JSON data of an asset	GET https://fdac-open.api.dev.unparallel.pt/assetDetails/:assetId	Open	https://fdac-open.api.dev.unparallel.pt/api-docs/#/FDAC/get_assetDetails__assetId
Search Assets	Endpoint used to search assets	POST https://fdac-open.api.dev.unparallel.pt/searchAssets	Open	https://fdac-open.api.dev.unparallel.pt/api-docs/#/FDAC/searchItems
Add New Asset	Add new asset using DCAT model	POST api/interoperability/dcat/components	Closed	https://www.iot-catalogue.com/swagger/653a8be4b93bd08e33dd5e8e
Obtain Asset	Obtain the asset using the canonical model	GET api/interoperability/canonical/components/:id	Closed	https://www.iot-catalogue.com/swagger/653a8be4b93bd08e33dd5e8e
Publish Asset	Publishes an asset using	PUT api/data/components/publish/:id	Closed	https://www.iot-catalogue.com/swagger/653a8be4b93bd08e33dd5e8e

Learning Centre (LC) Service

The Learning Centre Service is also embedded through a secure iframe within the Dashboard, allowing end-users to access educational content and training resources without leaving the main environment of the FAME Data Marketplace. The table below highlights the integration specifics and its contribution to the platform.

Table 6 : API endpoints of Learning Centre and Dashboard integration.

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
Generate Training Resources List	Generates the URL with the embedded page with the training resources list	GET https://fdac-open.api.dev.unparallel.pt/generateTrainingResourceListIframeURL	Open	https://fdac-open.api.dev.unparallel.pt/api-docs/#/Learning%20Center/get_generateTrainingResourceListIframeURL
Generate Training Resource Details	Generates the URL with the embedded page with the training resource info	GET https://fdac-open.api.dev.unparallel.pt/generateTrainingResourceDetailsIframeURL/:trainingResourceId	Open	https://fdac-open.api.dev.unparallel.pt/api-docs/#/Learning%20Center/get_generateTrainingResourceDetailsIframeURL___trainingResourceId_

Trading and Monetization (TM) Service

The Trading and Monetization Service is integrated with the Dashboard through RESTful API endpoints, supporting both public and internal backend operations related to data trading and monetization. The table below details the integration specifics and their purpose.

Table 7 : API endpoints of TM and Dashboard integration.

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
FAME Trading & Monetisation Public API	Trading & Monetisation Public API	(GET, POST) https://tm.fame-horizon.eu/api/v1.0/	Open	https://tm.fame-horizon.eu/swagger
FAME Trading & Monetisation Private API	Trading & Monetisation Private API	(GET, POST, DELETE) /tm/v1.0/	Closed	Not deployed publicly

Provenance and Tracing (PT) Service

The Provenance and Tracing Service connects to the Dashboard via RESTful APIs, providing structured metadata on datasets, projects, and data lineage. For the PT Service, two (2) separate tables follow: one summarizing the internal APIs (Table 8) and another detailing the public APIs (Table 9), each illustrating its dedicated role and connection to the Dashboard.

Table 8 : API endpoints of PT (Internal) and Dashboard integration.

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
ListSources	Lists all trusted source records	GET /pt/v1.0/sources	Closed	*https://gitlab.gftinnovation.eu/fame/framework/pt/-/blob/main/swagger-specification.private.json?ref_type=heads
RegisterSource	Registers a new trusted source	POST /pt/v1.0/sources	Closed	*
RetrieveSource	Retrieves a specific trusted source by PID	GET /pt/v1.0/sources/{pid}	Closed	*
DeregisterSource	Unregisters a trusted source	DELETE /pt/v1.0/sources/{pid}	Closed	*
ListOnboardedSources	Lists all active trusted sources	GET /pt/v1.0/active-sources	Closed	*
RetrieveOnboardedSource	Retrieves an active trusted source by PID	GET /pt/v1.0/active-sources/{pid}	Closed	*
RetrieveOffering	Retrieves a catalogue entry of an active offering	GET /pt/v1.0/offerings/{oid}	Closed	*
ConfirmOffering	Confirms (publishes) an offering	PUT /pt/v1.0/offerings/{oid}/confirm	Closed	*
RejectOffering	Rejects (deletes draft) offering	PUT /pt/v1.0/offerings/{oid}/reject	Closed	*
GrantPermission	Grants blockchain permission to trading account	POST /pt/v1.0/permissions	Closed	*
CheckPermission	Checks if a trading account has blockchain permission	GET /pt/v1.0/permissions/{tid}	Closed	*
RevokePermission	Revokes blockchain permission from trading account	DELETE /pt/v1.0/permissions/{tid}	Closed	*

Table 9 : API endpoints of PT (Public) and Dashboard integration.

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
List Sources	Lists all trusted source records	GET https://pt.fame-horizon.eu/api/v1.0/sources	Open	*https://gitlab.gftinnovation.eu/fame/framework/pt/-/blob/main/swagger-specification.public.json?ref_type=heads
Retrieve Source	Gets details of a specific trusted source	GET https://pt.fame-horizon.eu /api/v1.0/sources/{pid}	Open	*
List Onboarded Sources	Lists all active trusted source entries	GET https://pt.fame-horizon.eu /api/v1.0/active-sources	Open	*
Retrieve Onboarded Source	Retrieves details of a specific active trusted source	GET https://pt.fame-horizon.eu /api/v1.0/active-sources/{pid}	Open	*
List Publisher Assets	Lists trusted assets published by a trading account	GET https://pt.fame-horizon.eu /api/v1.0/assets	Open	*
Publish Asset	Initiates publishing of a trusted asset	POST https://pt.fame-horizon.eu /api/v1.0/assets	Open	*
Retrieve Asset	Gets catalogue entry & provenance data for an asset	GET https://pt.fame-horizon.eu /api/v1.0/assets/{aid}	Open	*
Revise Asset	Updates existing marketplace info for an asset	PUT https://pt.fame-horizon.eu /api/v1.0/assets/{aid}	Open	*
Unpublish Asset	Archives asset and its offerings	DELETE https://pt.fame-horizon.eu /api/v1.0/assets/{aid}	Open	*

List Offerings for Asset	Lists all active offerings for a trusted asset	GET https://pt.fame-horizon.eu /api/v1.0/offerings	Open	*
Publish Offering	Publishes new offering linked to an asset	POST https://pt.fame-horizon.eu /api/v1.0/offerings	Open	*
Retrieve Offering	Gets catalogue entry of a specific offering	GET https://pt.fame-horizon.eu /api/v1.0/offerings/{oid}	Open	*
Unpublish Offering	Removes an offering from the marketplace	DELETE https://pt.fame-horizon.eu /api/v1.0/offerings/{oid}	Open	*
Retrieve Catalogue Stats	Gets general stats of the marketplace catalogue	GET https://pt.fame-horizon.eu /api/v1.0/stats	Open	*

Operational Governance (GOV) Service

The Operational Governance Service is integrated through RESTful APIs, supplying the Dashboard with governance metrics, analytics, and compliance insights. For the GOV Service, two (2) separate tables follow: one summarizing the internal APIs (Table 10) and another detailing the public APIs (Table 11), highlighting their distinct integration points and purposes within the platform.

Table 10 : API endpoints of GOV (Internal) and Dashboard integration.

API Name	**Short Description**	**Endpoint Prefix**	**Access Level**	**Documentation Detail**
Start Member Onboarding	Starts onboarding of a new member	POST /gov/v1.0/members	Closed	*https://gitlab.gftinnovation.eu/fame/framework/gov/-/blob/main/swagger-specification.private.json?ref_type=heads
Confirm Registration	Confirms registration of member	PUT /gov/v1.0/members/{pid} /confirm-registration	Closed	*
Confirm Deregistration	Confirms deregistration of member	PUT /gov/v1.0/members/{pid} /confirm-deregistration	Closed	*

Check Active Membership	Checks if member is active	GET /gov/v1.0/members/{pid}/check	Closed	*
Check Authority	Checks authority affiliation	GET /gov/v1.0/authorities/{did}/check	Closed	*
List Active Members	Lists active members	GET /gov/v1.0/active-members	Closed	*
Resolve Member PID	Resolves PID to member details	GET /gov/v1.0/active-members/{pid}	Closed	*
List Active Authorities	Lists active authorities	GET /gov/v1.0/active-authorities	Closed	*
List Trading Accounts	Lists trading accounts	GET /gov/v1.0/accounts	Closed	*
Start Account Request	Starts processing of account request	POST /gov/v1.0/accounts	Closed	*
Get Trading Account	Retrieves trading account details	GET /gov/v1.0/accounts/{tid}	Closed	*
Confirm Permissioning	Confirms account permissioning	PUT /gov/v1.0/accounts/{tid}/allowed	Closed	*
Confirm De-permissioning	Confirms account de-permissioning	PUT /gov/v1.0/accounts/{tid}/disallowed	Closed	*
Enqueue Request	Adds new processing request to queue	POST /gov/v1.0/rqueue	Closed	*
Retrieve Request	Retrieves request from queue	GET /gov/v1.0/rqueue/{rid}	Closed	*
Finalize Request	Finalizes a queued request	PUT /gov/v1.0/rqueue/{rid}	Closed	*
List Users	Lists users onboarded in the Root Authority	GET /gov/v1.0/users	Closed	*
Start User Onboarding	Starts onboarding process of a candidate user	POST /gov/v1.0/users	Closed	*

Retrieve User	Retrieves specific user information	GET /gov/v1.0/users/{uid}	Closed	*

Table 11 : API endpoints of GOV (Public) and Dashboard integration.

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
List Members	Retrieves list of members	GET https://gov.fame-horizon.eu/api/v1.0/members	Open	*https://gitlab.gftinnovation.eu/fame/framework/gov/-/blob/main/swagger-specification.public.json?ref_type=heads
Onboard Member	Starts onboarding a new member	POST https://gov.fame-horizon.eu/api/v1.0/members	Open	*
Retrieve Member Info	Retrieves full information set of a member	GET https://gov.fame-horizon.eu/api/v1.0/members/{pid}	Open	*
Update Member Info	Updates non-trusted info linked to a member	PUT https://gov.fame-horizon.eu/api/v1.0/members/{pid}	Open	*
Migrate Member Entity	Changes legal entity info of an existing member	POST https://gov.fame-horizon.eu/api/v1.0/members/{pid}/migrate	Open	*
Offboard Member	Offboards an existing member	POST https://gov.fame-horizon.eu/api/v1.0/members/{pid}/offboard	Open	*
Check Active Membership	Checks active membership status	GET https://gov.fame-horizon.eu/api/v1.0/members/{pid}/check	Open	*
List Active Members	Lists active members	GET https://gov.fame-horizon.eu/api/v1.0/active-members	Open	*

Resolve Active Member PID	Resolves PID into name and details	GET https://gov.fame-horizon.eu/api/v1.0/active-members/{pid}	Open	*
List Trading Accounts	Lists trading accounts	GET https://gov.fame-horizon.eu/api/v1.0/accounts	Open	*
Enroll Trading Account	Starts enrollment of a new trading account	POST https://gov.fame-horizon.eu/api/v1.0/accounts	Open	*
Retrieve Trading Account	Retrieves details of a trading account	GET https://gov.fame-horizon.eu/api/v1.0/accounts/{tid}	Open	*
Disenroll Trading Account	Permanently disenrolls a trading account	DELETE https://gov.fame-horizon.eu/api/v1.0/accounts/{tid}	Open	*
Credit Trading Account	Mints FDE tokens and transfers to trading account	POST https://gov.fame-horizon.eu/api/v1.0/accounts/{tid}/credit	Open	*
Debit Trading Account	Burns FDE tokens from a trading account	POST https://gov.fame-horizon.eu/api/v1.0/accounts/{tid}/debit	Open	*
Check Request Status	Checks the status of a request	GET https://gov.fame-horizon.eu/api/v1.0/requests/{rid}	Open	*
Retrieve Ecosystem Stats	Retrieves basic stats of FAME ecosystem	GET https://gov.fame-horizon.eu/api/v1.0/stats	Open	*
List Users	Lists users in ROA database	GET https://gov.fame-horizon.eu/api/v1.0/users	Open	*
Onboard User	Starts onboarding process for a user	POST	Open	*

		https://gov.fame-horizon.eu/api/v1.0/users		
Re-invite User	Restarts onboarding process for a user	POST https://gov.fame-horizon.eu/api/v1.0/users/{uid}/reinvite	Open	*
Retrieve User	Retrieves user details	GET https://gov.fame-horizon.eu/api/v1.0/users/{uid}	Open	*
Delete User	Deletes a user	DELETE https://gov.fame-horizon.eu/api/v1.0/users/{uid}	Open	*
Offboard User	Offboards a user	POST https://gov.fame-horizon.eu/api/v1.0/users/{uid}/offboard	Open	*
Accept Onboarding Invitation	Accepts invitation to complete user onboarding	POST https://gov.fame-horizon.eu/api/v1.0/invitations	Open	*

Authentication and Authorization Infrastructure (AAI) Service

The Authentication and Authorization Infrastructure Service integrates with the Dashboard via RESTful APIs to ensure secure end-user authentication and role-based access. The table below highlights the integration points and their functional impact.

Table 12 : API endpoints AAI and Dashboard integration.

API Name	**Short Description**	**Endpoint Prefix**	**Access Level**	**Documentation Detail**
Issuer Credential	Issues credentials to the user	POST /credential-offers	Closed	*https://gitlab.gftinnovation.eu/fame/framework/aai/-/blob/main/README.md?ref_type=heads
Credential Offer Status	Checks credentials' status	GET https://auth.fame-horizon.eu/credential-offer-status	Open	*

Auth Requests	Verifies user credentials when logging into the marketplace	POST https://auth.fame-horizon.eu/definitions/FAMEv1/auth-requests	Open	*
Auth Status	Verifies the authentication status, indicating whether the user has accepted or declined the request from the mobile wallet	GET https://auth.fame-horizon.eu/auth-status	Open	*
OpenId Credential Issuers	OpenId API link for issuers	GET https://identity.fame-horizon.eu/openid-credential-issuer	Open	*
OpenId Credential Configurations	OpenId API link for configurations	GET https://identity.fame-horizon.eu/well-known/openid-configuration	Open	*
Token	Gets the token	POST https://identity.fame-horizon.eu/token	Open	*
Credentials	Performs credentials' communications with wallet	POST https://identity.fame-horizon.eu/credentials	Open	*
User Validations	Checks whether the user is onboarded to the FAME Data Marketplace, and returns JWT token	POST https://auth.fame-horizon.eu/api/user-validations	Open	*
Create Machine Identities	Used for internal systems interactions	POST api/create-machine-identities'	Closed	*

Pricing Advisory Tool (PAT) Service

The Pricing Advisory Tool Service is integrated into the Dashboard through RESTful API endpoints, delivering pricing strategy insights and recommendation analytics. The table below outlines the integration features and their purpose.

Table 13 : API endpoints PAT and Dashboard integration.

API Name	**Short Description**	**Endpoint Prefix**	**Access Level**	**Documentation Detail**

Pricing Parameters	Receives input values from the offering creation process (such as asset description, asset type, asset ID, offering ID, etc.), and based on these input parameters, returns a specific list of questions with predefined answer sets and types	POST https://pat.fame-horizon.eu/pat/v1.0/pricing-params	Open	https://pat.fame-horizon.eu/swagger#/Pricing%20Advisory%20Tool%20-%20PAT/PatController_getParams
Pricing Advice	Gets a completed questionnaire and the offering identifier as input and returns a recommended price for the asset	POST https://pat.fame-horizon.eu/pat/v1.0/pricing-advice	Open	https://pat.fame-horizon.eu/swagger#/Pricing%20Advisory%20Tool%20-%20PAT/PatController_getPricing

Advanced Search Service

The Advanced Search Service is connected to the Dashboard via RESTful APIs, enabling dynamic keyword and filter-based content searches. The table below details how this integration supports search functionalities within the platform.

Table 14 : API endpoints Advanced Search and Dashboard integration.

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
FDAC Search	Search for data elements on the FDAC	POST https://fame-fdac.iot-catalogue.com/api/data/search	Open	https://fame-fdac.iot-catalogue.com/swagger/653a8be4b93bd08e33dd5e8e
GOV Active Members	Allows the retrieval of active members in the FAME Data Marketplace	GET /gov/v1.0/active-members	Closed	https://gov.fame-horizon.eu/swagger
PT Offerings	Allows the retrieval of a specific offering	GET /pt/v1.0/offerings/{oid}	Closed	https://pt.fame-horizon.eu/swagger
PT Offerings	Allows the retrieval of all the offerings for a specific asset	GET /api/v1.0/offerings	Closed	https://pt.fame-horizon.eu/swagger

Asset Policy Manager (APM) Service

The Asset Policy Manager Service is accessed through an external interface, with the Dashboard redirecting users to manage and view access policies. The table below illustrates the main integration points and their relevance.

Table 15 : API endpoints APM and Dashboard integration.

API Name	Short Description	Endpoint Prefix	Access Level	Documentation Detail
Assets Policy Editor	Exposes the necessary functionality to create new policy for an asset, and also can delete it	(POST, DELETE) /private/api/v1.0/asset-policy-editor	Closed	Not documented publicly
Assets Policy Editor	Fetches an existing policy and updates an existing policy of a specific asset	(GET, PUT) https://apm.fame-horizon.eu/public/api/v1.0/asset-policy-editor	Open	*https://apm.fame-horizon.eu/public/swagger-ui/index.html#/
Assets Content Access Controller	Checks if the underlying end-user has content access to a specific asset	POST	Open	*

		https://apm.fame-horizon.eu/public/api/v1/asset-access/check-one		
Assets Content Access Controller	Checks if the underlying end-user has content access to a list of assets	POST https://apm.fame-horizon.eu/public/api/v1/asset-access/check-many	Open	*
Assets Content Access Controller	Retrieves a list of all the assets that the underlying user has access to their contents	GET https://apm.fame-horizon.eu/public//api/v1/asset-access/check-all	Open	*
Assets Marketplace Visibility Controller	Checks if the underlying end-user can view a specific asset on the FAME Data Marketplace	POST https://apm.fame-horizon.eu/public//api/v1/asset-visibility/check-one	Open	*
Assets Marketplace Visibility Controller	Checks if the underlying end-user can view a list of assets on the FAME Data Marketplace	POST /api/v1/asset-visibility/check-many	Open	*
Assets Marketplace Visibility Controller	Retrieves a list of all the asset IDs that the underlying user can view on the FAME Data Marketplace	GET /api/v1/asset-visibility/check-all	Open	*

3.3.3 Internal Integration of Backend Services

In the first place, we should clarify what the integration of backend services is about, and in what it differs from the Dashboard / Backend integration that was described in section 3.3 of this document. In a nutshell, the former is also called *internal integration*: its goal is to enable cross-module communication for those workflows that require the *orchestrated cooperation* of multiple platform modules.

The most elaborate example of such workflow is "Asset Publishment" (see deliverable "D4.4 Blockchain-based Data Provenance Infrastructure II" [8]): although the starting point is a call to an Open API service endpoint provided by the Provenance and Tracing (P&T) module, the workflow then proceeds in the background with the involvement of the Federated Data Asset Catalogue (FDAC), Asset Policy Management (APM) and Operational Governance (GOV) modules.

In this scenario, P&T acts as the *initiator and orchestrator*, while FDAC and GOV are *contributors*. All cooperative workflows supported by the FAME platform follow this pattern, with one initiator/orchestrator and one or more contributors. Internal integration is thus the development, testing and deployment of *non-public* service endpoints (i.e., deployed on the common hosting environment of the FAME platform's backend but not accessible from the public Internet) that serve the needs of cooperative workflows.

A much simpler but more generally applicable example is user authentication / authorization: every Open API service endpoint, regardless of the module, needs to ensure that their callers are actually authorized to consume the provided service. In order to do that, the caller of a service endpoint must provide a valid Verifiable Onboarding Credential (VOC) token issued by a trusted Onboarding Authority.

While the details of the onboarding and authentication process are not in the scope of the present document (see WP3 deliverables for more information on that), suffice to say that all Open API service endpoints need to integrate with the Authentication and Authorization Infrastructure (AAI) platform module.

Collectively, non-public endpoints represent the *Integration API* of the FAME Platform. The Integration API follows the same approach of the Open API: REST[1] architectural style on top of the HTTP communication protocol. A significant difference between the two APIs is that Integration endpoints can only be called by other modules running in the same hosting environment – which is, at the time of writing, a Kubernetes cluster provided by Amazon Web Services (AWS) and managed by GFT. Access control over the Integration API is obtained by means of networking rules rather than through the integration with the AAI module. Moreover, as all communications travel internally to the hosting environment, the HTTP channel used by non-public service endpoints is *not* encrypted to make the information exchange more efficient.

Another difference is the pattern used to address API endpoints. For Open API operations, the pattern is the following:

```
https://<module-acronym>.fame-horizon.eu/api/<api-version>/<operation-path>
```

The public network address starts with a third-level domain name that identifies the platform module – for instance, `gov` for Operational Governance, `tm` for Trading & Monetization. The service path always has `api` as its first element and the software release number as the second. Finally, the third element points to the specific operation. As an example, this is the current endpoint of the Operational Governance operation that is used to register a new member of the FAME Federation:

```
[POST] https://gov.fame-horizon.eu/api/1.0/members
```

For Integration API operations, the pattern is this:

```
http://<lan-address>/<module-acronym>/<api-version>/<operation-path>
```

In this case, the network address is only reachable within the hosting environment's local area network (LAN). In a Kubernetes environment as the one currently in use, this address is the service name and the TCP port it listens to – both parameters being configured at deployment time. Then, the service path starts with the module acronym (`gov`, `tm`, etc.). The rest of the path follows the same convention as for the Open API. For instance, this is the endpoint of the Trading & Monetization service used to create a new offering token:

```
[POST] http://tm-internal:3000/tm/1.0/offerings
```

The following interaction diagrams show two major workflows where cross-module communication is required.

[1] REpresentational State Transfer - see https://www.codecademy.com/article/what-is-rest

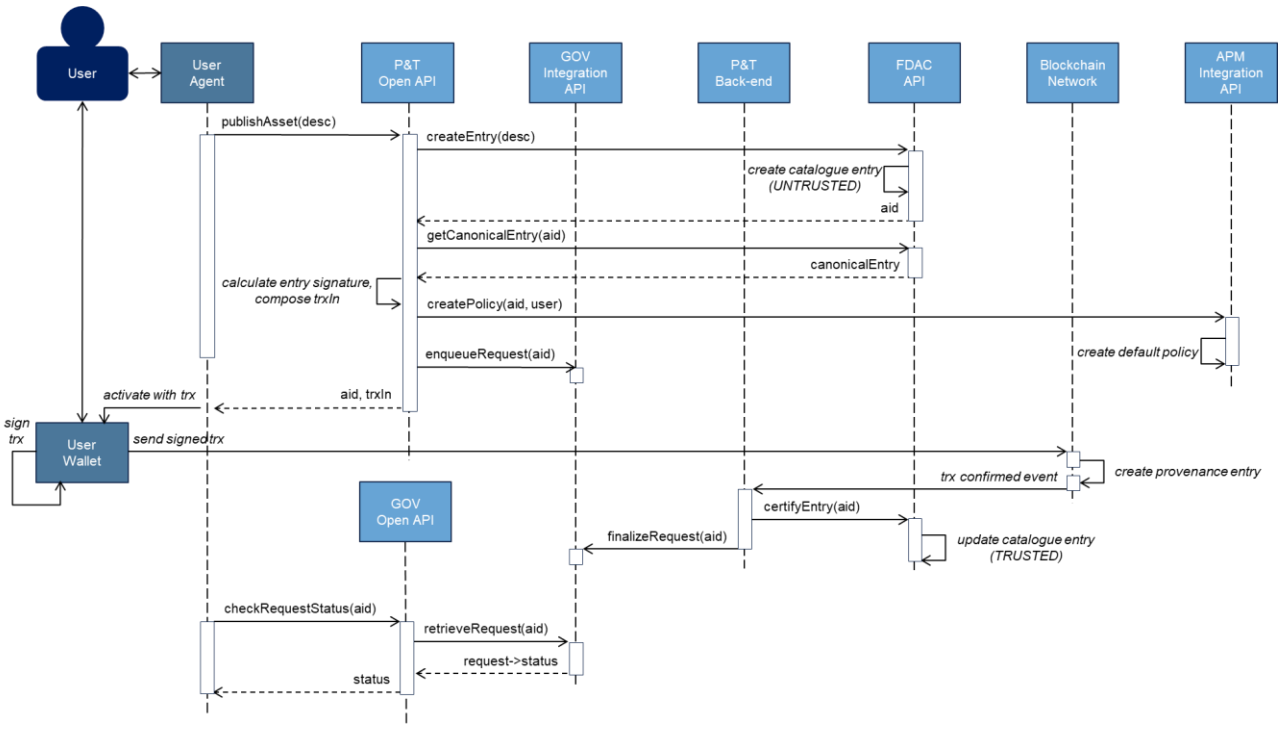


Figure 42: Asset Publishment

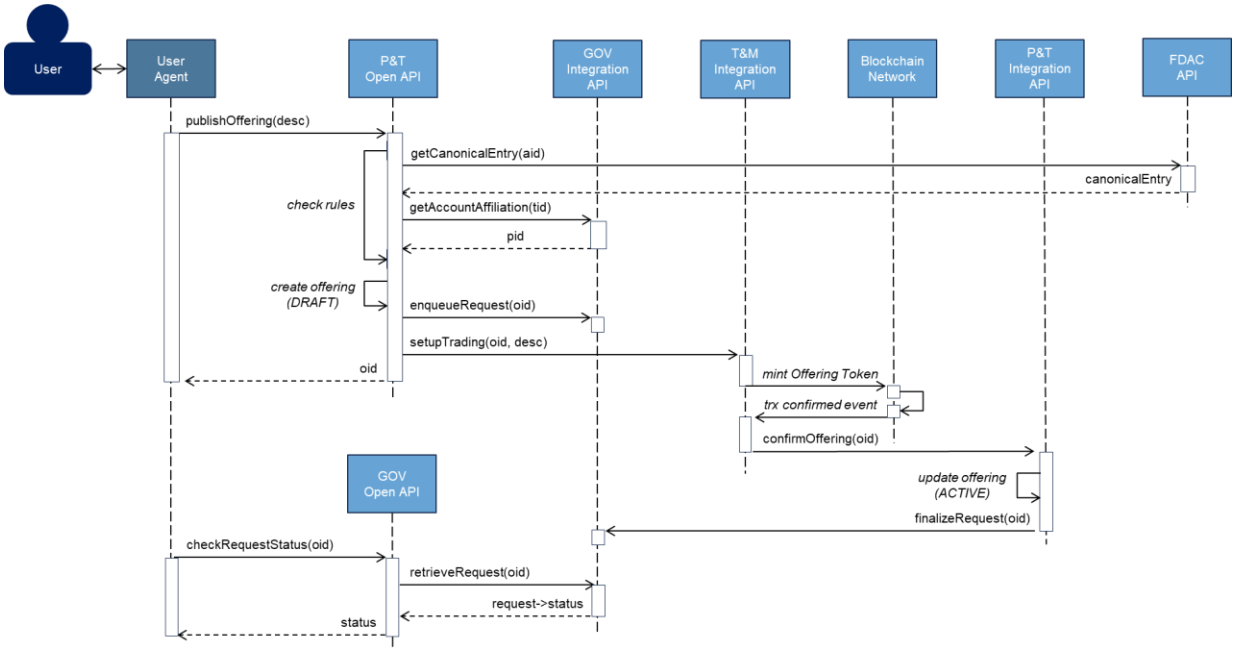


Figure 43: Offering Publishment

With regards to the integration of the FDAC module, it should be noted that FDAC is a pre-existing independent system that runs in a separate runtime environment, so that the standard approach for the deployment of non-public services (e.g., the `createEntry` operation) could not be adopted. Instead, FDAC's integration points are deployed as public services that are only accessibly with a system-owned API token.

Internal integration points are documented in the same way as FAME's Open API: online by means of the Swagger UI[2] and offline in the relevant project deliverables. The only exception to this rule is for closed-source modules - namely, FDAC and APM - that only provide offline documentation. Here

[2] See https://swagger.io/

below we provide a screenshot of the Swagger UI, documenting the internal integration points provided by the P&T and GOV modules.

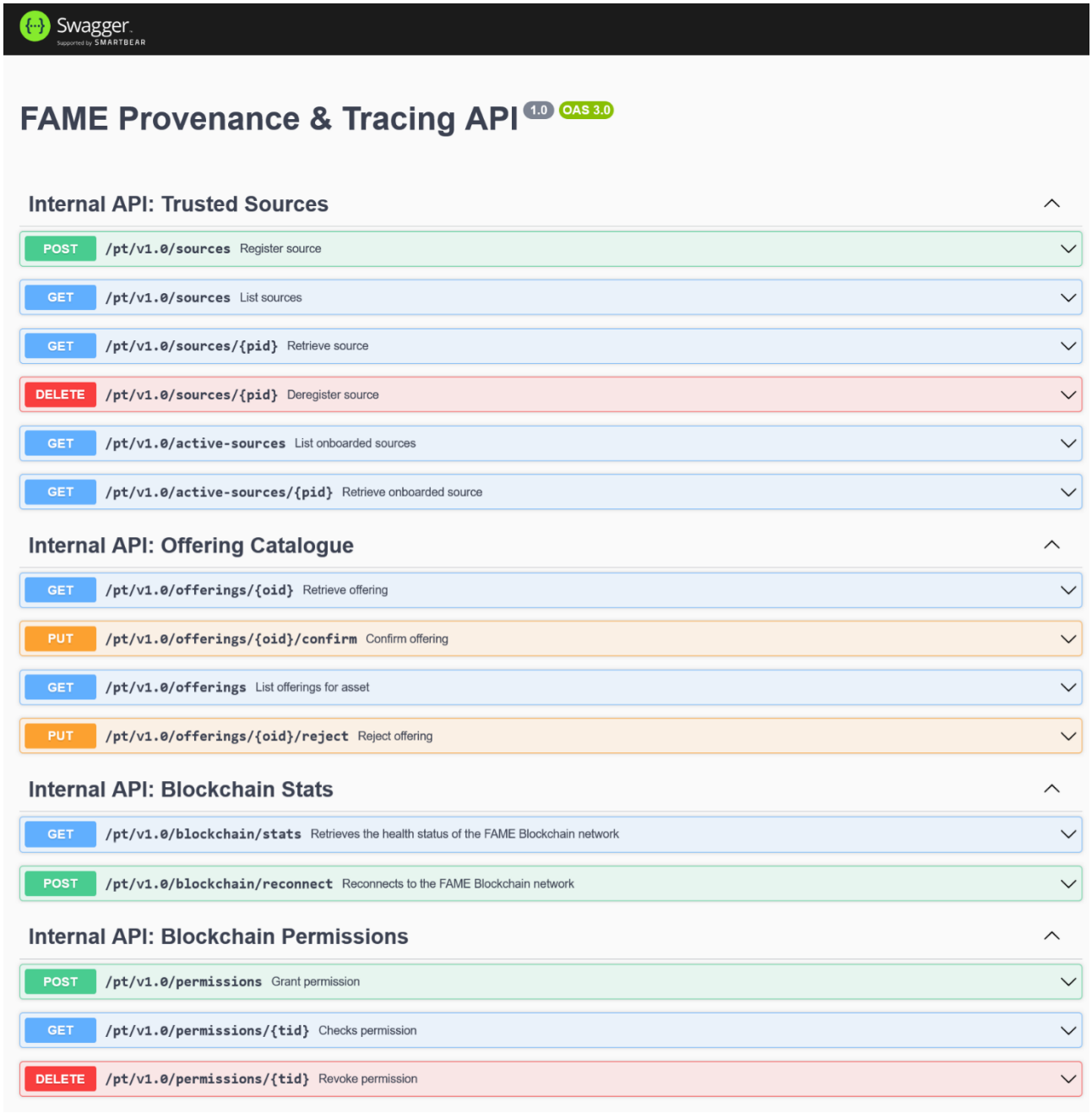


Figure 44: P&T service endpoints for internal integration

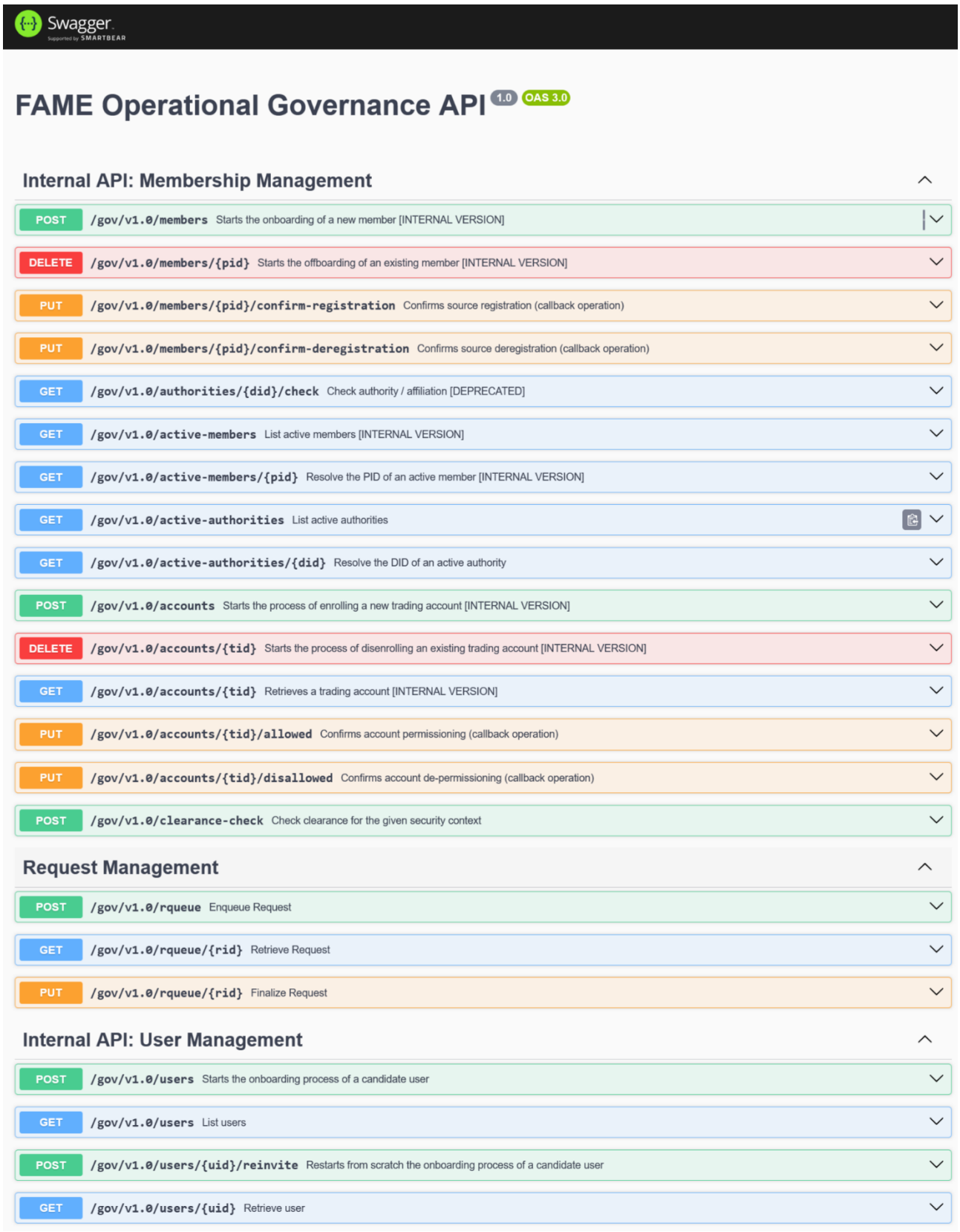


Figure 45: GOV service endpoints for internal integration

4 FAME Data Assets Integration

4.0 Introduction

This chapter presents the integration mechanism provided by Federated Data Asset Catalogue (FDAC) to support the integration with the other FAME components. As the FDAC is the component responsible for the indexing and management of the Asset metadata, all operations relates with the asset addition, update and discovery require to interact. FDAC. The following of this section provides the high level description of the FDAC, present it mechanisms for integration and describes how they were used in FAME.

4.1 Federated Data Asset Catalogue (FDAC)

The FDAC functions as a registry within FAME, tasked with systematically cataloging and archiving various assets. These assets include datasets, AI models, services, and essential documentation, ensuring a structured approach to data management. The registry can represent assets originating from FAME activities and index assets from external sources such as relevant Data Spaces or Data Marketplaces. All asset information is accessible to any FAME component requiring it through well-defined interfaces.

This section outlines FDAC's main functionalities, which include storing and indexing asset metadata and providing interfaces to support CRUD (Create, Read, Update, Delete) actions on these assets. These functionalities are supported by the "Asset Management" and "Database" components shown in the figure below. Additionally, the "Search" component enhances user discovery functionalities. More information about each component of the FDAC architecture can be found in D3.5 – Federated Data Assets Catalogue II.

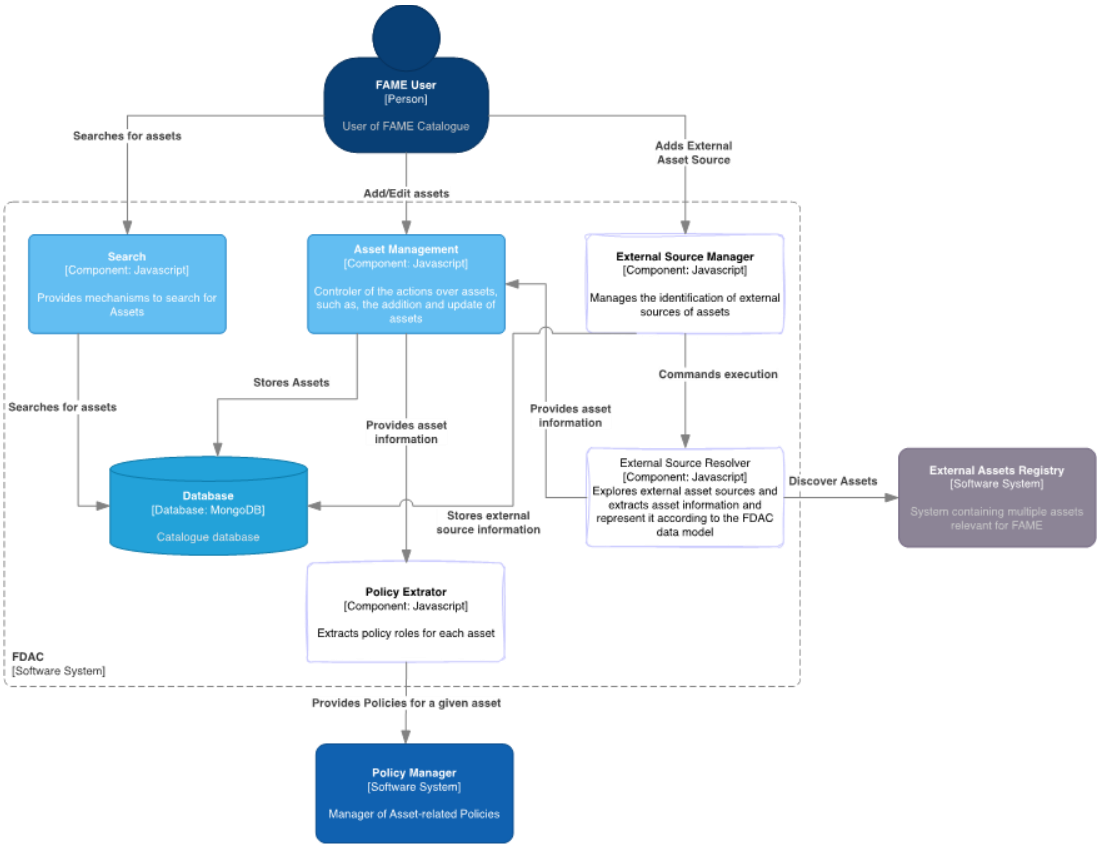


Figure 46 : FDAC in C4 Architecture

4.1.0 REST API

FDAC provides a REST API to enable the communication with other systems and FAME components, allowing not only the add and retrieve information from the catalogue but also to access to some functionalities. Such REST API provides an authentication mechanism based on authentication tokens to ensure that only authentication systems have access to the information.

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 add, edit, and retrieve information about assets are listed in the figure below.

Figure 47 : FDAC REST API Swagger

A dedicated endpoint is also provided to allow the execution of more advanced search queries. This search endpoint, which allows users to submit queries to the FDAC for components that match specific criteria, is shown in figure bellow. 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 following figure.

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

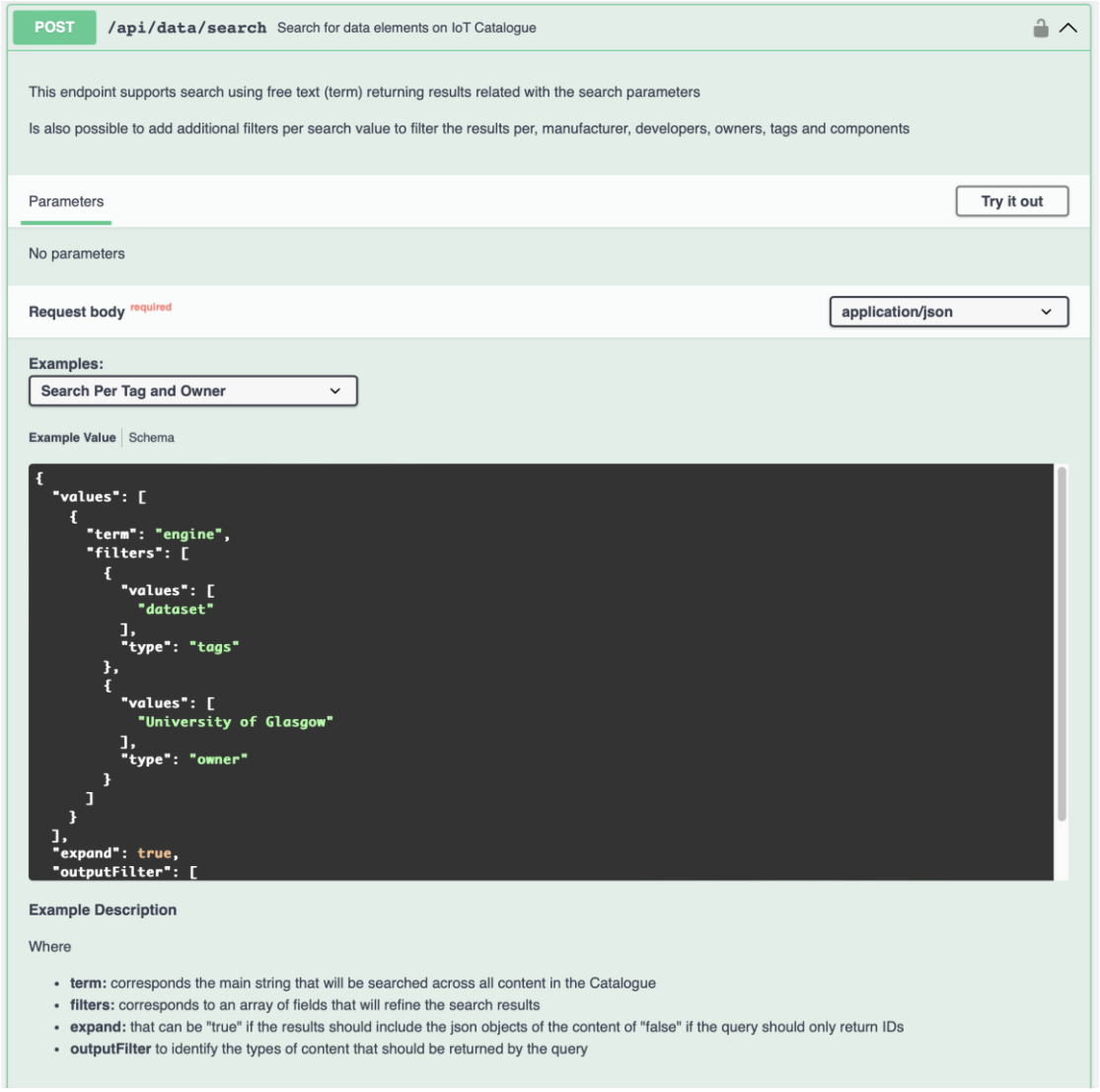


Figure 48 : FDAC REST API Search response example

4.1.1 Iframe

FDAC has its own user interfaces to present asset listings and asset detail page, among others. In order to allow and simplify the usage/integration of such user interfaces in the user interface of other systems, FDAC recommends the usage of Iframes[4]. FDAC provides iframes with the capability to show only specific visual elements, instead of a full page, and the capability to customize the colours of the rendered elements, providing the iframe with some customization tools to allow a better visual integration on the existing interfaces of the systems where the iframe will be integrated.

To better integrate Iframes with the designed behaviour to the external systems (host system), some mechanisms are provided to allow the host system to send information to the iframe. Using such mechanisms, host systems to have some control over the iframe, like the ability select to the content shown or create automations relating user actions on elements of the host system with changes in the visualizations inside the iframes.

[4] https://www.w3schools.com/html/html_iframe.asp

4.2 Integration

The REST API exposed by FDAC and the Iframes are the two main approaches to integrate FDAC with external systems. On situations where the integration requires information exchange between FDAC backend and other systems, the REST API is the approach more indicated.

However, on situations where an integration is required at the level of the User interface, then Iframes are the recommended approach. This approach is used for the integration with the FAME Dashboard and its usage is described below.

4.2.0 Dashboard Integration

Two different iFrames are integrated in the FAME Dashboard. One to show the lists of assets the in the assets listing page (accessible on the page: {dashboard_address}/fdac) and another to show the details of an asset (accessible on the page: {dashboard_address}/asset/{asset_id}).

The result of integration of the iframe presenting the list of assets is show in Figure 49, where the region of the page corresponding to the Iframe is highlighted in the dashed rectangle. This iframe presents the assets as cards in a paginated list. The list of assets shown iframe is controlled by a list of IDs that the dashboard provides to the Iframe when the page is loaded. Moreover, information about filters (such as tags) can also be passed to the Iframe using the PostMessage() HTML method.

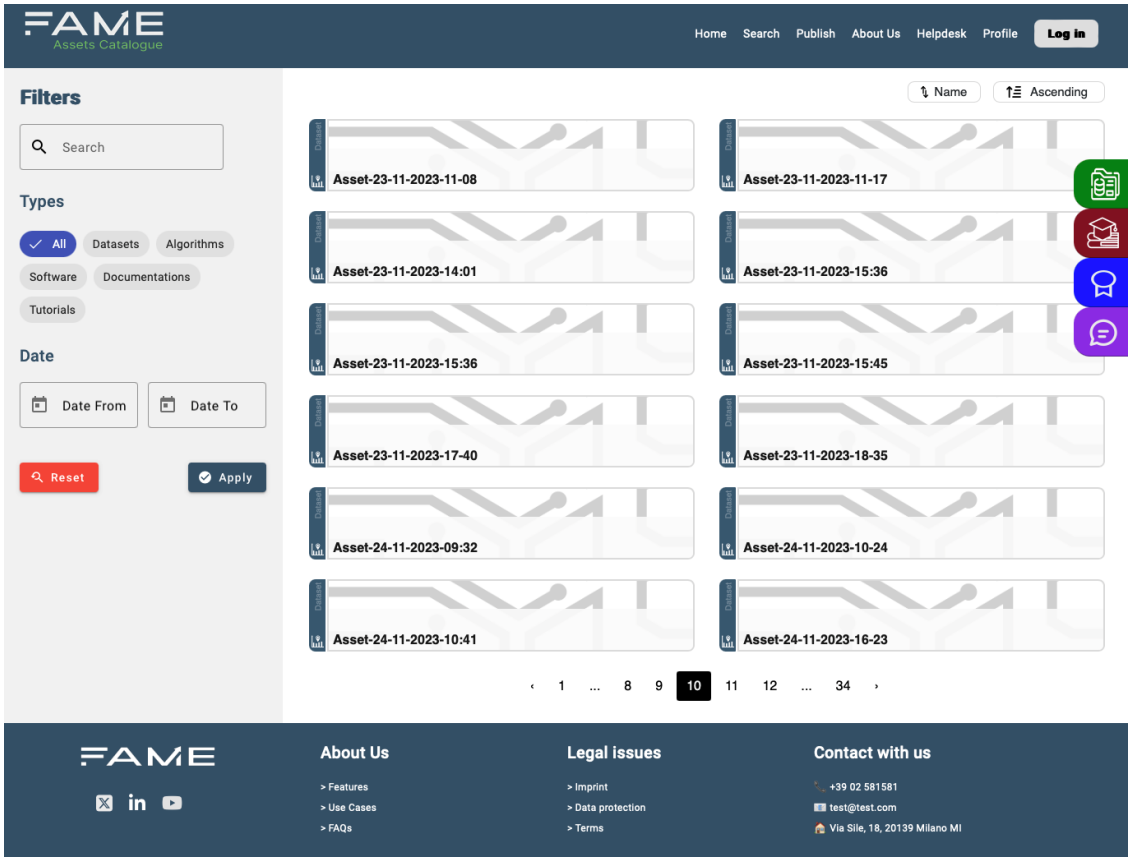


Figure 49 : Highlight of FDAC's iframe to show list of assets

The result of the integration of iframe for present the details of an asset is shown in Figure 50, as well as represented the corresponding area by the dashed rectangle. This iframe receives the assets ID to be presented. This information is retrieved from the address bar an passed to the iframe when the page is loaded.

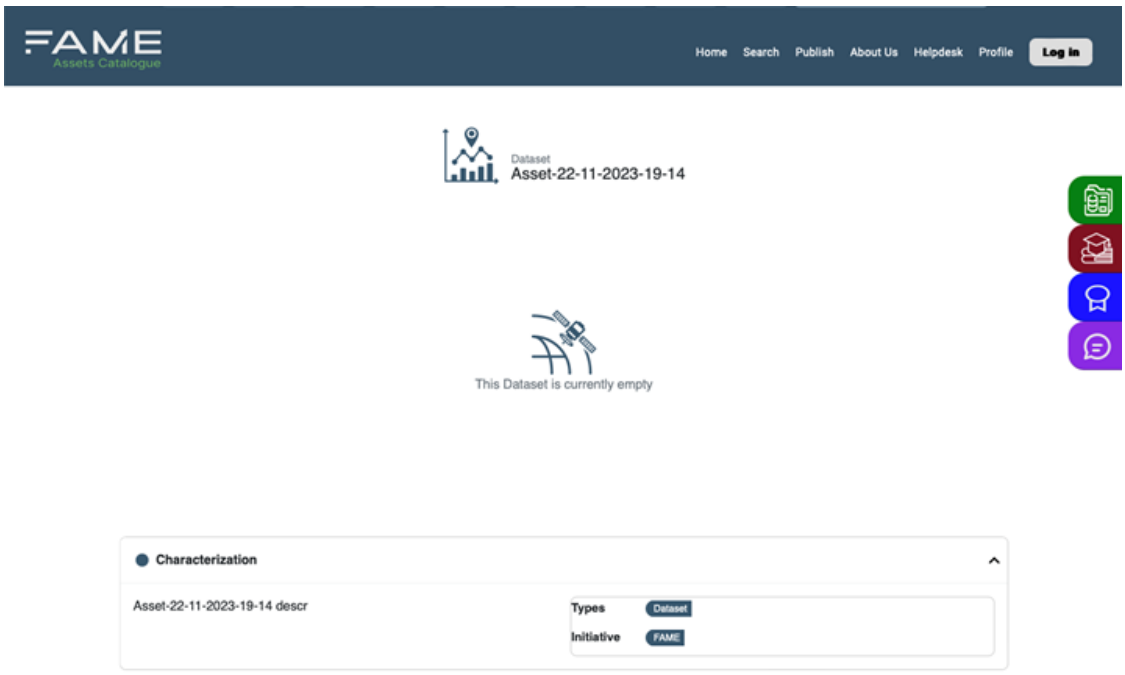


Figure 50 : Highlight of FDAC's iframe to show asset details

4.2.1 Asset Policy Manager Integration

The Asset Policy Manager (APM) is used to identify which assets a used as access to based-on the policies stored in it. The diagram is Figure 51, represented the information flow needed for the integration of the APM. When a user accesses the /fdac page, the Dashboard used the User Authentication information (JWT Token) to ask the APM which assets this User can see, based on the user characteristics and the policies in place for the assets. The APM returns a list with the IDs of the assets that the user can visualizes and the Dashboard sends that list to the FDAC Iframe that will then list all the assets.

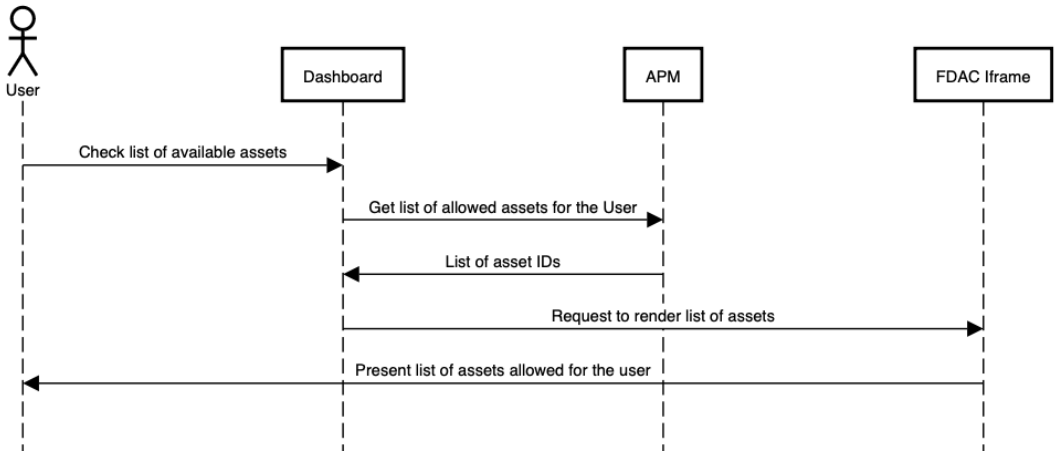


Figure 51 : Information flow of the integration between Dashboard, APM and FDAC Iframe

5 FAME Connector

5.0 Introduction

The FAME Connector is a component in the Federated Data Space architecture of the FAME platform, enabling secure, scalable, and transparent data consumption between data providers and consumers. Given the decentralization principle inherent in Federated Data Spaces, the FAME Connector plays a pivotal role in allowing data to reside and be managed by providers on their own premises while being accessible to authorized consumers.

This section outlines the detailed architectural and implementation considerations for deploying a FAME Connector. It presents a reference design, describes the key components, and discusses the implications and implementation requirements for stakeholders involved in the data sharing process.

5.1 Overview of Data Consumption Components

The high-level architecture of the FAME Connector comprises several components, which are depicted in the following diagram:

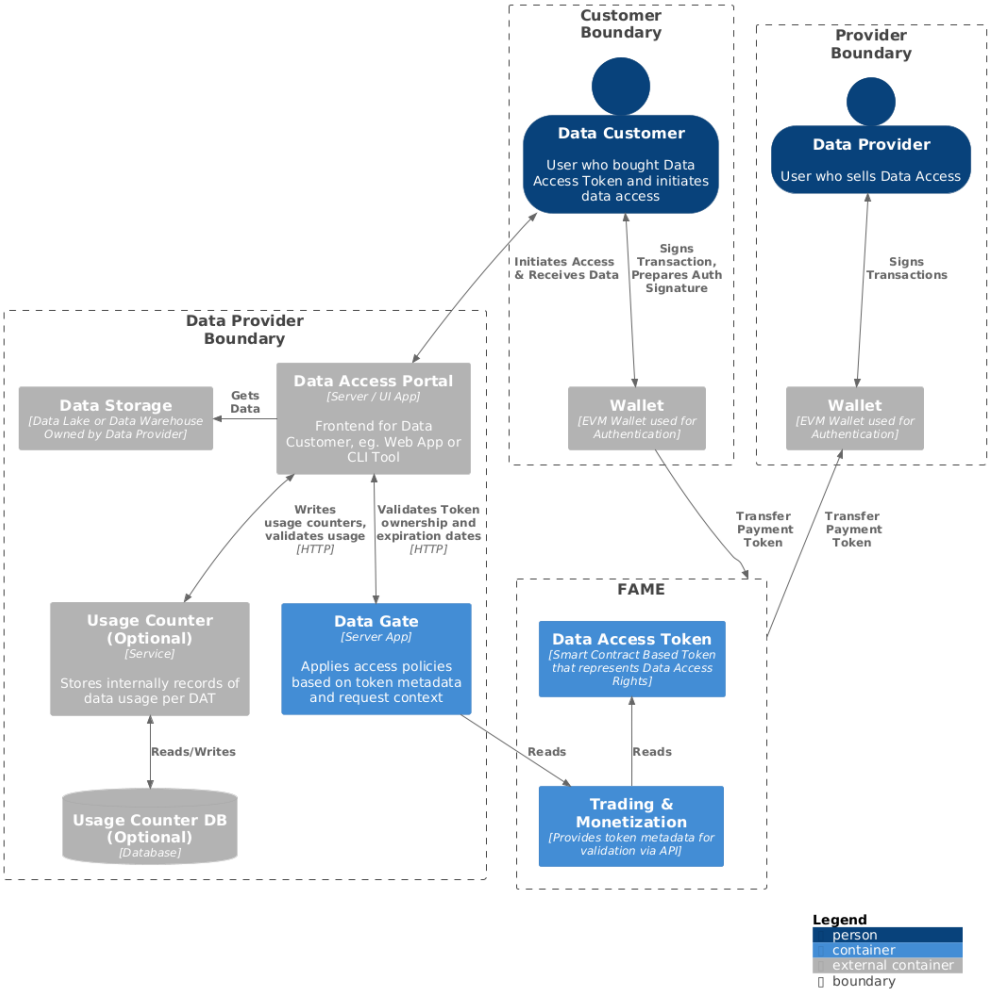


Figure 52 : high-level architecture of the FAME Connector

In this diagram, the blue boxes represent components provided by the FAME platform as operational systems or deployable artifacts, while the grey boxes represent components the Data Provider must implement or configure.

5.1.0 Component Descriptions

- **Data Customer (Persona)**: The end-user who acquires Data Access Tokens (DATs) from the FAME Marketplace.
- **Customer Wallet (External System)**: An EVM-compatible wallet (e.g., MetaMask), holding DATs and used for authentication.
- **Data Access Portal (Provider's On-Premise)**: A consumer-facing interface for data access deployed by the Data Provider.
- **Data Gate (Provider's On-Premise)**: A secure server application enforcing data access controls and verifying token ownership.
- **Usage Counter (Provider's On-Premise, Optional)**: An optional component tracking data consumption under Pay-As-You-Go (PAYG) or Pay-As-You-Use (PAYU) models.
- **Data Storage (Provider's On-Premise)**: Provider-managed data repositories (e.g., Data Lakes, S3 Buckets).
- **Trading & Monetization (FAME Platform)**: FAME Public API providing necessary metadata.

5.2 Justification of Decentralized Data Storage

The architecture of the FAME Connector explicitly follows a Federated Data Space model, emphasizing decentralized data management. In alignment with the European Data Governance Act and principles of self-sovereignty, privacy, and security, there is no centralized data storage within the FAME platform. This design decision ensures:

- **Data Sovereignty**: Providers retain full control over their data, managing security, compliance, and access policies directly.
- **Enhanced Security and Privacy**: Data is stored locally, minimizing the risk associated with centralized breaches or unauthorized access.
- **Compliance and Regulation**: Providers can better manage their regulatory responsibilities (e.g., GDPR compliance) by having direct control over their data storage infrastructure.

5.3 Current State and Implementation Approach

It is important to highlight that the FAME Connector is not delivered as a ready-to-deploy standalone system. Instead, the FAME project provides:

- **Reference Design**: Architectural patterns and recommendations that stakeholders can adapt to their specific needs.
- **Reference Implementation**: Sample implementations demonstrating essential functionalities and integration patterns.
- **Deployable Core Elements**: Key elements, such as the **Data Gate**, are provided as deployable components or artifacts, significantly reducing the implementation effort required from Data Providers.

5.3.0 Data Gate Overview

The Data Gate is a critical deployable component provided by the FAME project, responsible for enforcing secure access control to data resources. Its main function is validating consumer requests through cryptographic verification, checking token ownership and validity, and managing payment token withdrawals under PAYG monetization models.

The Data Gate exposes a set of API endpoints that enable these functionalities. Below is the screenshot from Open API Documentation with a brief description of each endpoint.

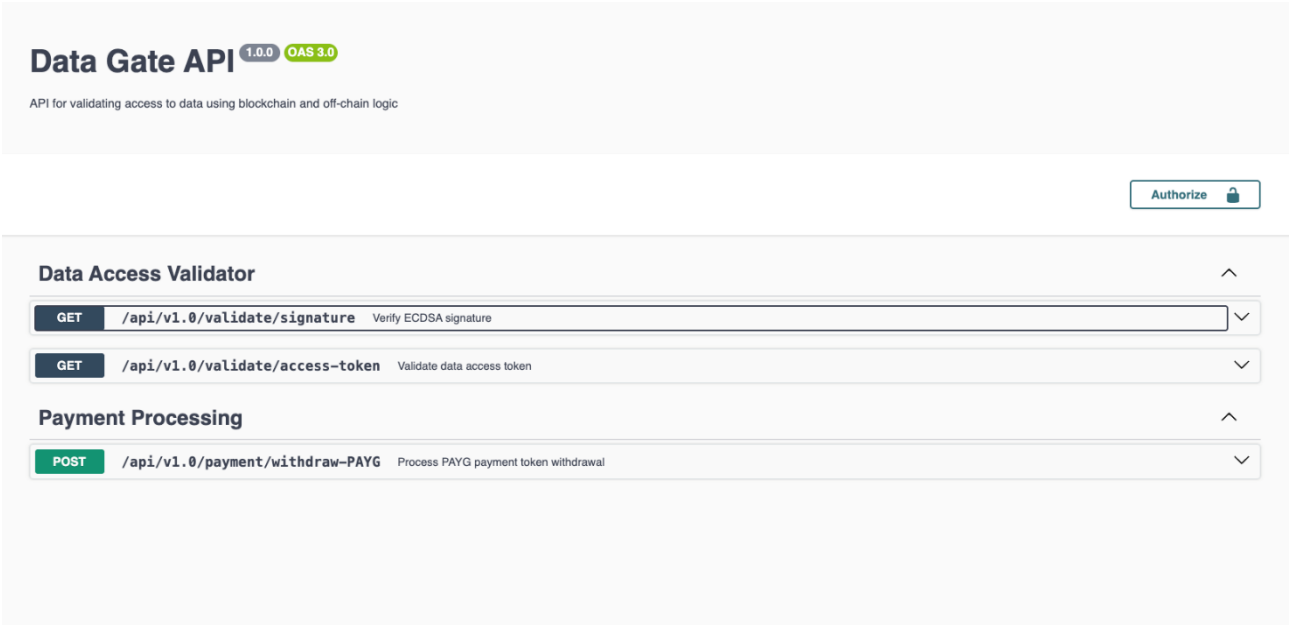


Figure 53 : Data Gate API

- **Cryptographic Signature Verification**
  The Data Gate verifies cryptographic signatures through the endpoint (GET /api/v1.0/validate/signature). It ensures that the provided signature corresponds precisely to the Ethereum address claimed by the data consumer, providing strong authentication by confirming the consumer's identity.
- **Data Access Token Validation**
  Through the (GET /api/v1.0/validate/access-token) endpoint, the Data Gate validates various aspects of data access tokens (DATs), including ownership, expiration, and specific access rights depending on the token model (Subscription - SUB, Pay-As-You-Go - PAYG, or Pay-As-You-Use - PAYU). This comprehensive validation guarantees that only legitimate and active token holders can access protected data resources.

- **PAYG Payment Token Withdrawal**
  For monetization under the Pay-As-You-Go model, the Data Gate exposes the endpoint (POST /api/v1.0/payment/withdraw-PAYG) to securely handle financial transactions. It withdraws payment tokens directly from the consumer's Ethereum wallet, transferring them to the provider's account. This transaction is transparently executed and recorded on the blockchain, ensuring compliance and traceability of financial agreements.

5.3.1 Proof-of-Concept Data Access Portal

As part of the reference implementation, FAME has also developed a **Proof-of-Concept (POC) Data Access Portal**. This POC application demonstrates and tests the interaction with the validation mechanisms provided by the Data Gate, illustrating the end-to-end data consumption workflow as described below in the detailed sequence diagrams for Subscription (SUB), Pay-As-You-Go (PAYG), and Pay-As-You-Use (PAYU) token types.

The portal is implemented following a client-server architecture, consisting of a dedicated backend server and a frontend UI application. The backend server securely stores and manages the Data Provider's private key, which is essential for securely triggering the withdrawal of payment tokens from the Data Consumer's wallet during the PAYU data consumption workflow. The backend functionality is fully documented and available via an API, as illustrated in the provided Swagger screenshot below.

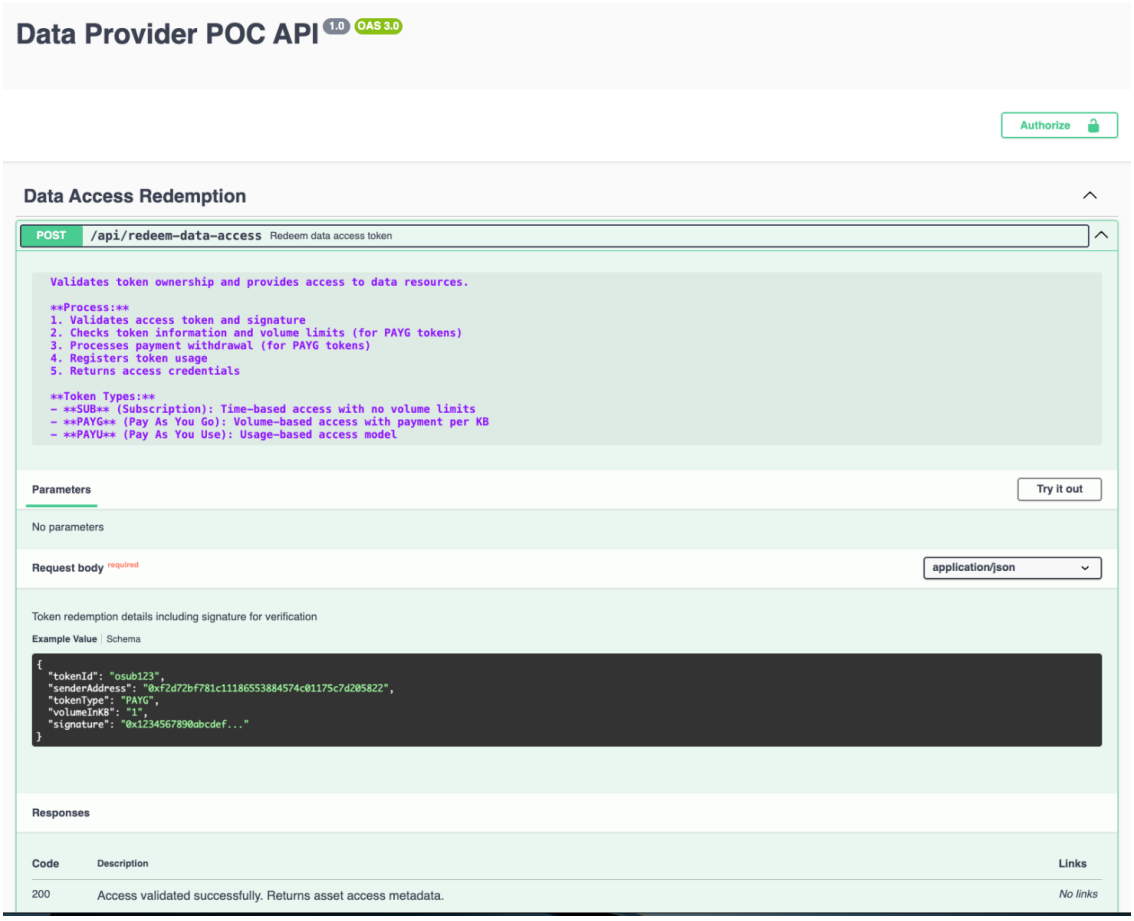


Figure 54 : API swagger documentation

The frontend UI application provides user-friendly interactions, enabling Data Consumers to seamlessly navigate through data assets and trigger data consumption workflows.

The detailed features of the POC Data Access Portal include:

- **List of Available Data Assets:**
  The portal displays available datasets or data assets for consumption. Consumers can browse assets, each clearly associated with an offering and token type.

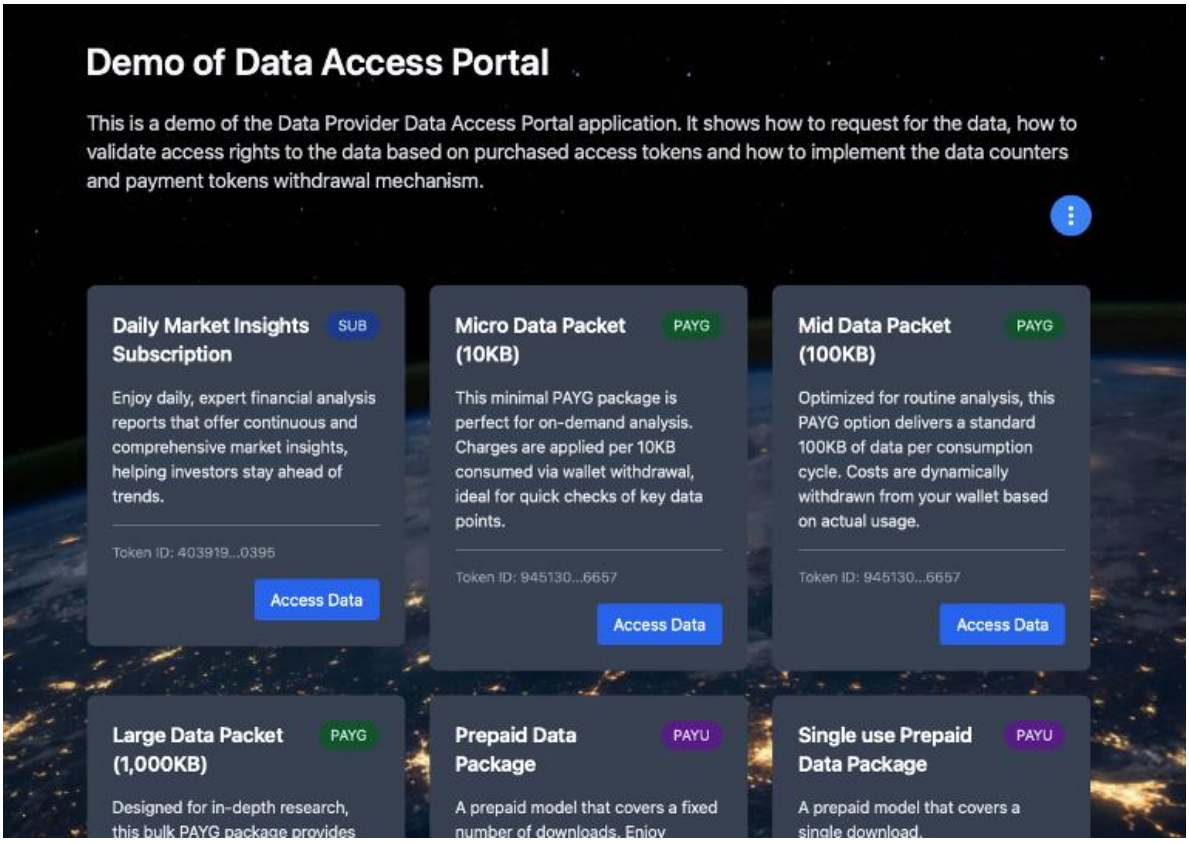


Figure 55 : Demo Portal. List of Available Data Assets

- **Validating the Data Customer Signature:**
  Before accessing data, the portal verifies the consumer's cryptographic signature to confirm ownership of the wallet holding the appropriate Data Access Token (DAT). This step ensures that the consumer requesting access is indeed the legitimate owner of the token and associated wallet.

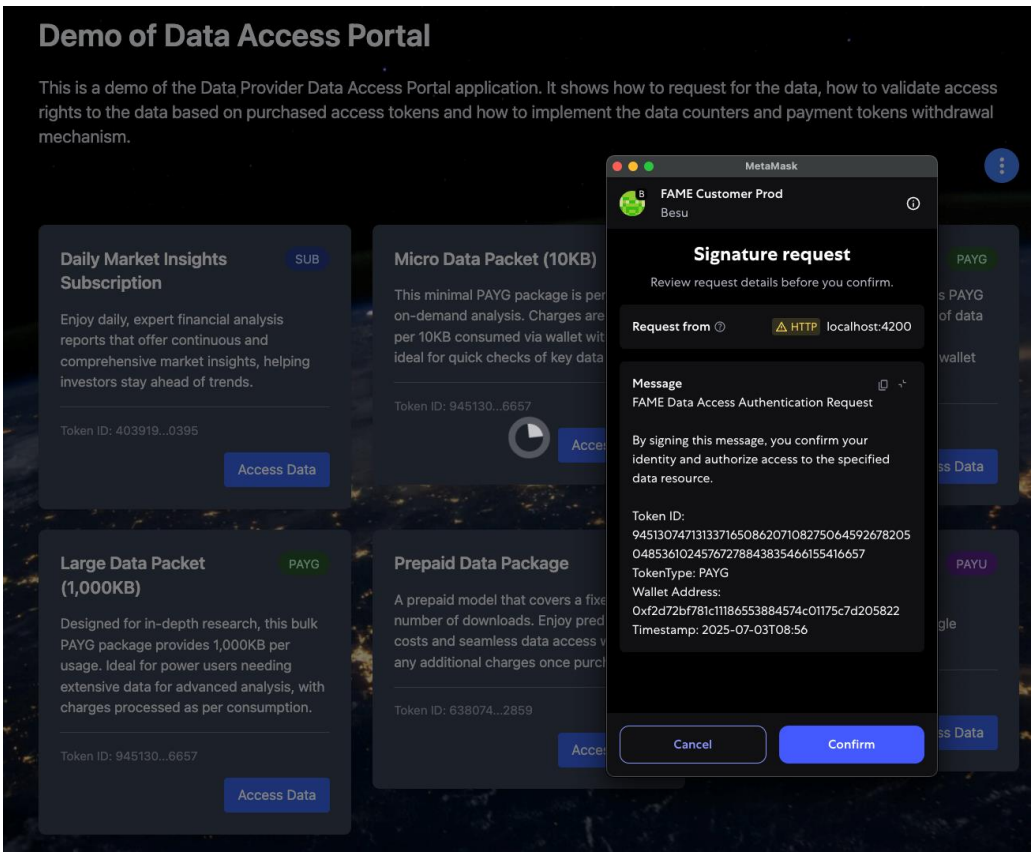


Figure 56 : Demo Portal, Validating the Data Customer Signature

- **Asset Selection for Consumption:**
  Authorized users select an appropriate data asset based on their acquired token rights, initiating the validation workflow via the Data Gate component.
- **Access Restriction Message:**
  If a user's public address does not hold the appropriate Data Access Token, the portal displays a clear message indicating that access is disabled, effectively demonstrating real-time token validation and enforcement of access policies.

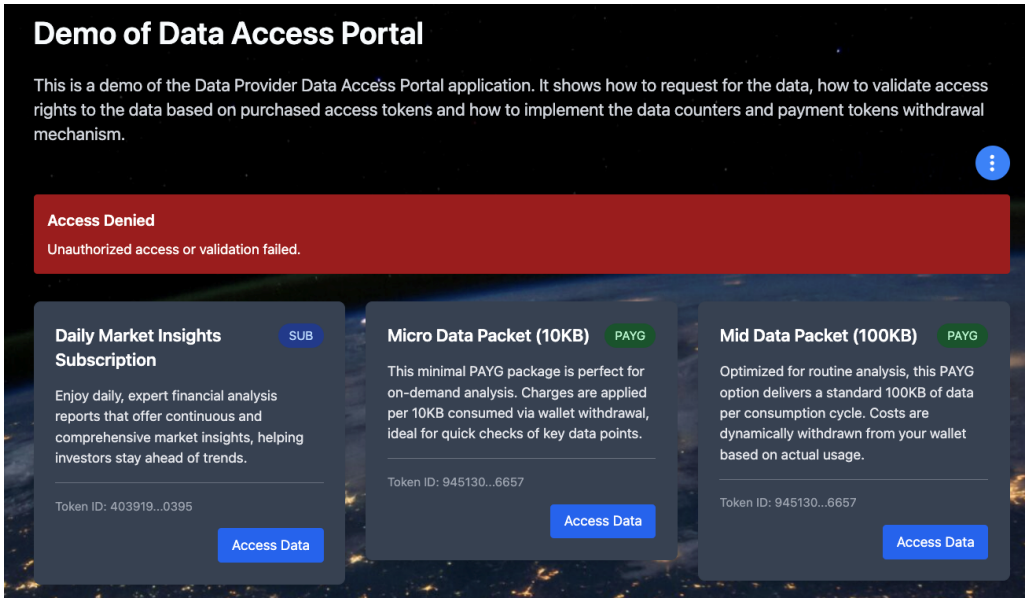


Figure 57 : Demo Portal, Access Restriction Message

- **Mocked Data Access Page:**
  Upon successful validation, users are presented with a mocked version of the data access page, representing the secure delivery of data to authorized consumers.

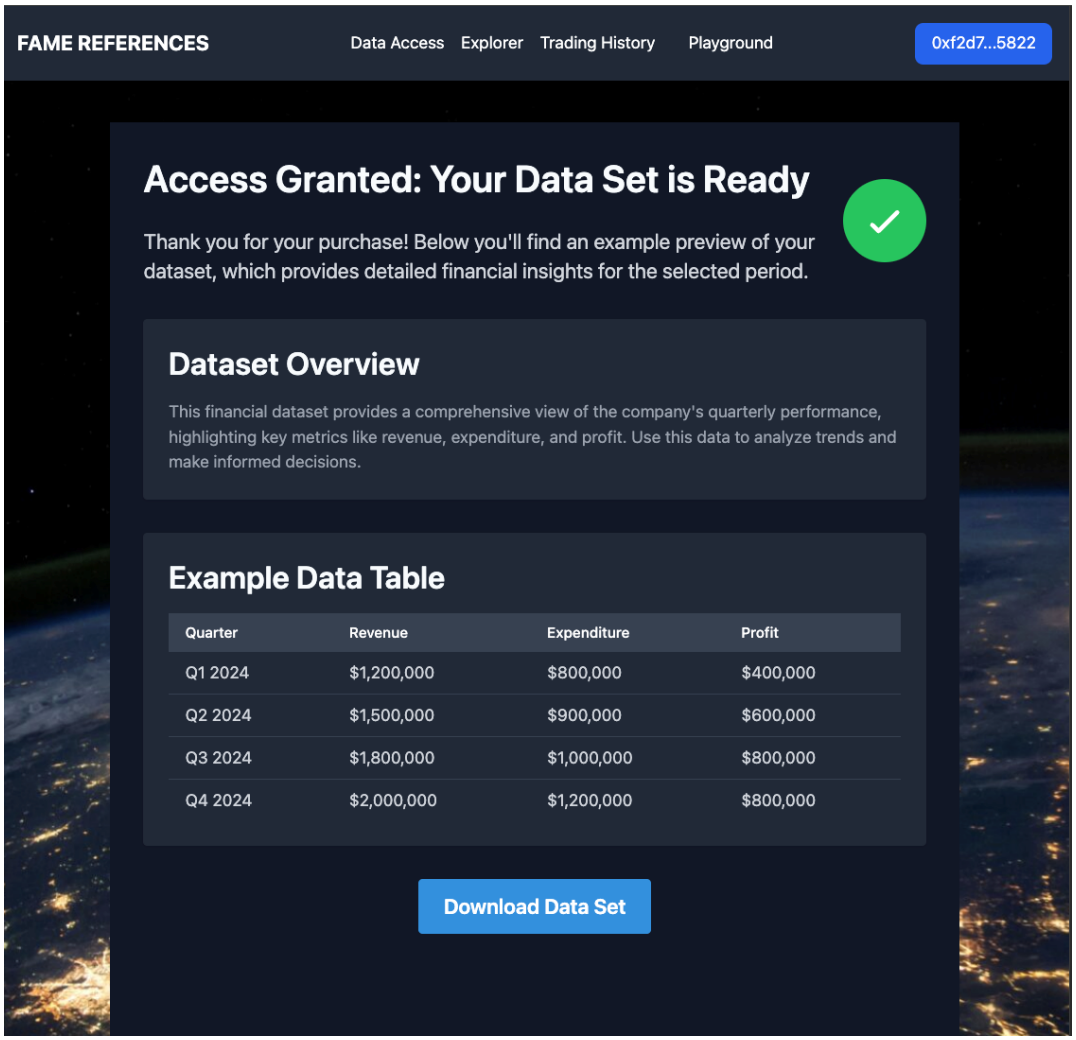


Figure 58 : Demo Portal, Data Access Page

- **POC Usage Counters for PAYG and PAYU Models:**
  The portal includes a POC implementation of usage counters for PAYG and PAYU monetization models, demonstrating the tracking of consumption metrics such as the volume of data downloaded or the number of permitted downloads. These usage counters interact with validation logic to enforce data consumption limits in real time.
- **Automated Transfer of Payment Tokens (PAYU Workflow):**
  For the PAYU monetization model, the POC portal demonstrates the automated transfer of payment tokens from the consumer's wallet to the provider's wallet upon successful data consumption.

This enhanced POC provides tangible, practical demonstrations of the FAME Data Connector functionality, clearly illustrating how integrated Data Gate security mechanisms, usage tracking, and automated payment handling work in practice.

5.4 Provider-Side Implementation Requirements

To securely integrate with the FAME Connector, Data Providers must address the following steps and considerations:

5.4.0 Prerequisites

- Successful onboarding on the FAME platform.
- Definition and registration of assets and offers within the FAME platform.

5.4.1 Implementation Steps

1. **Deploy Data Storage Infrastructure**:
    a. Providers set up secure repositories (Data Lakes, S3 Buckets) ensuring controlled data access.
2. **Deploy Data Access Portal**:
    a. Handles consumer authentication via wallet-signed messages.
    b. Displays available datasets and associated offers with defined access conditions.
    c. Forwards data requests securely to the Data Gate.
3. **Deploy and Configure Data Gate**:
    a. Validates consumer requests and DATs.
    b. For PAYG models, facilitates token withdrawals from consumer wallets by securely managing the provider's private keys.
4. **Implement Usage Counter (Optional)**:
    a. Recommended for metered usage models (PAYG/PAYU) to ensure accurate billing and monitoring.

5.5 Consumer-Side Data Access Workflow

Consumers interact with the Data Providers Data Access Portal through a clearly defined process involving the following stages:

5.5.0 Prerequisites

- Possession of an EVM-compatible wallet.
- Acquisition of relevant DAT (NFT) from the FAME Marketplace.

5.5.1 Access Steps

1. **Authentication**: Connect a wallet to the Data Access Portal, signing messages for account ownership verification.
2. **Data Request**: Use the portal interface to specify data requests based on entitlement.
3. **Automated Validation**: Automatic validation of the DAT and consumer requests by the Data Gate.
4. **Data Delivery**: Secure delivery of the requested data from the provider's storage upon successful validation.

5.5.2 Detailed Sequence Diagrams

The following diagrams illustrate specific consumer workflows for different monetization models.

5.5.2.0 Subscription-Based Access

In Subscription model, access is granted as long as the consumer's subscription token (NFT) is valid and not expired.

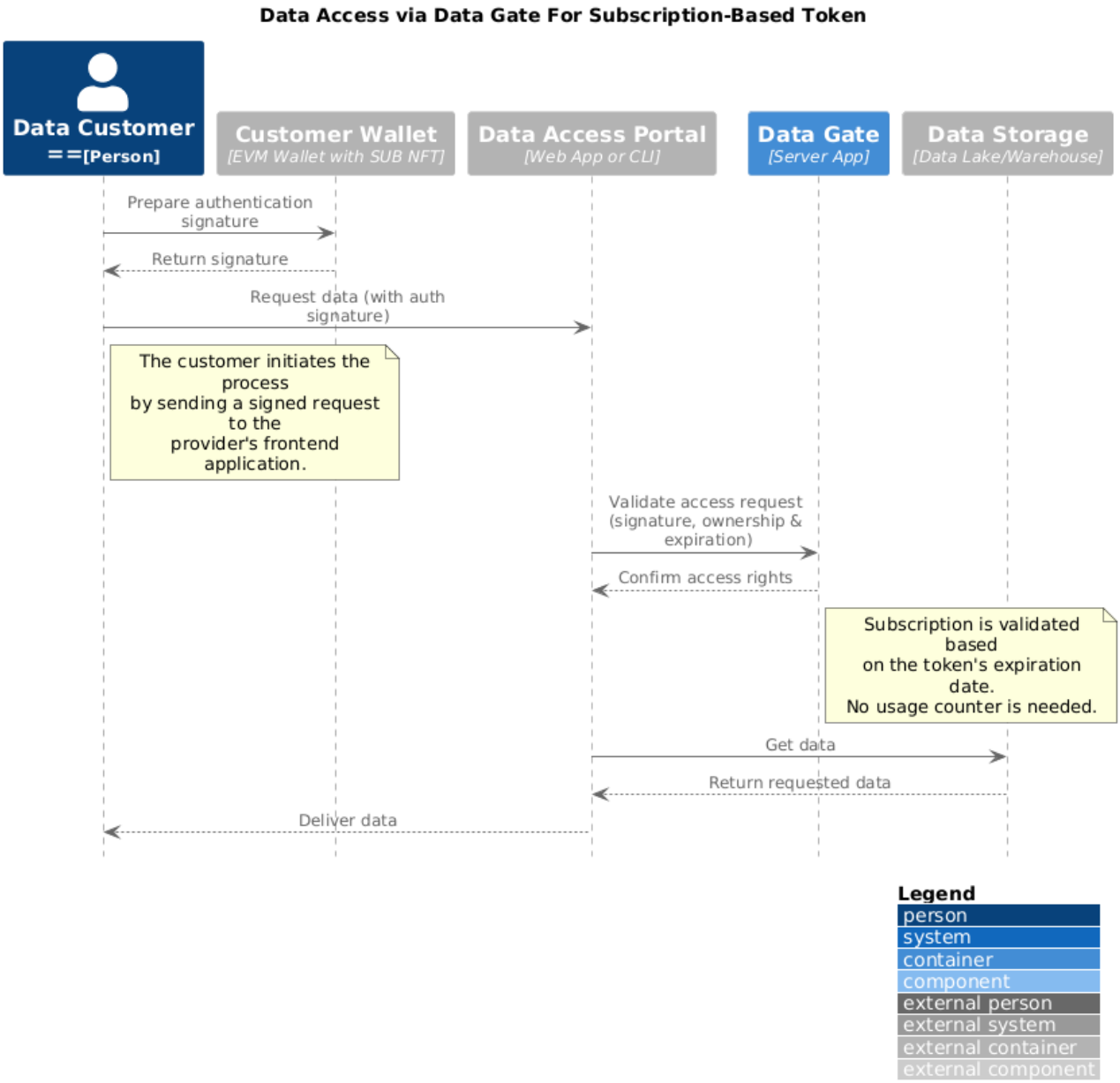


Figure 59 : Data Access via Data Gate for subscription-based token

Workflow Steps:

1. The customer authenticates with the Data Access Portal by signing a message.
2. The Data Access Portal sends the request to the Data Gate with offer ID, which validates the signature, token ownership, and expiration date.
3. If valid, the Data Gate confirms access rights.

4. The Data Access Portal retrieves the data from storage and delivers it to the customer.

5.5.2.1 Pay-As-You-Go (PAYG) Access

PAYG model involves tracking data usage (e.g., volume) and settling payments as data is consumed.

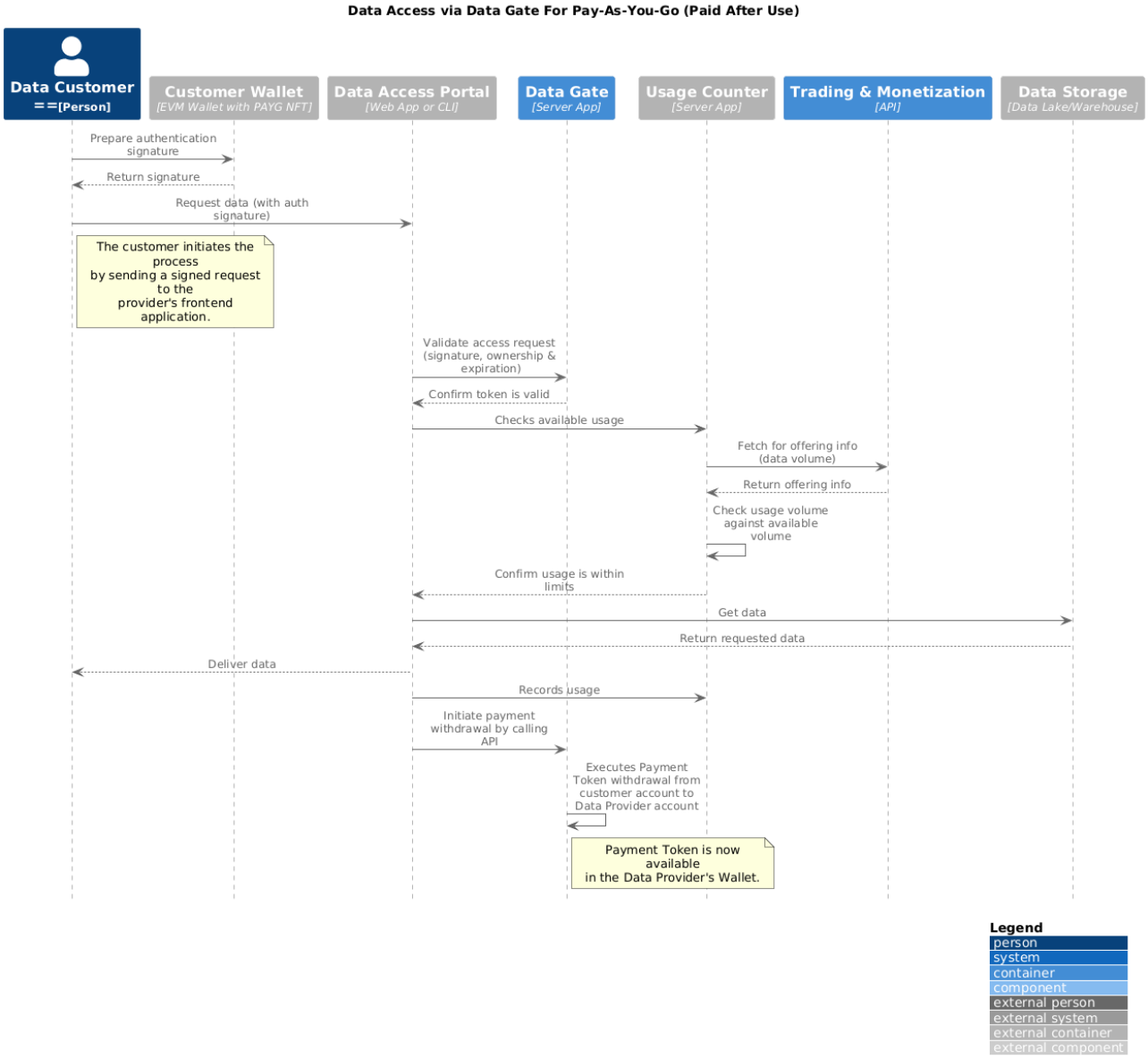


Figure 60 : Data Access via Gate for Pay-As-You-Go

Workflow Steps:

1. The customer authenticates.
2. The Data Gate validates the request.
3. The optional Usage Counter checks if the requested usage is within the available volume defined in the offering.
4. The Data Access Portal retrieves and delivers the data.
5. The Data Access Portal records the usage in the Usage Counter.

6. Finally, the Data Access Portal can trigger the Data Gate to initiate the withdrawal of payment tokens from the customer's wallet.

5.5.2.2 Pay-As-You-Use (PAYU)/Prepaid Access

In PAYU prepaid model, the consumer has a limited number of downloads or credits associated with their token.

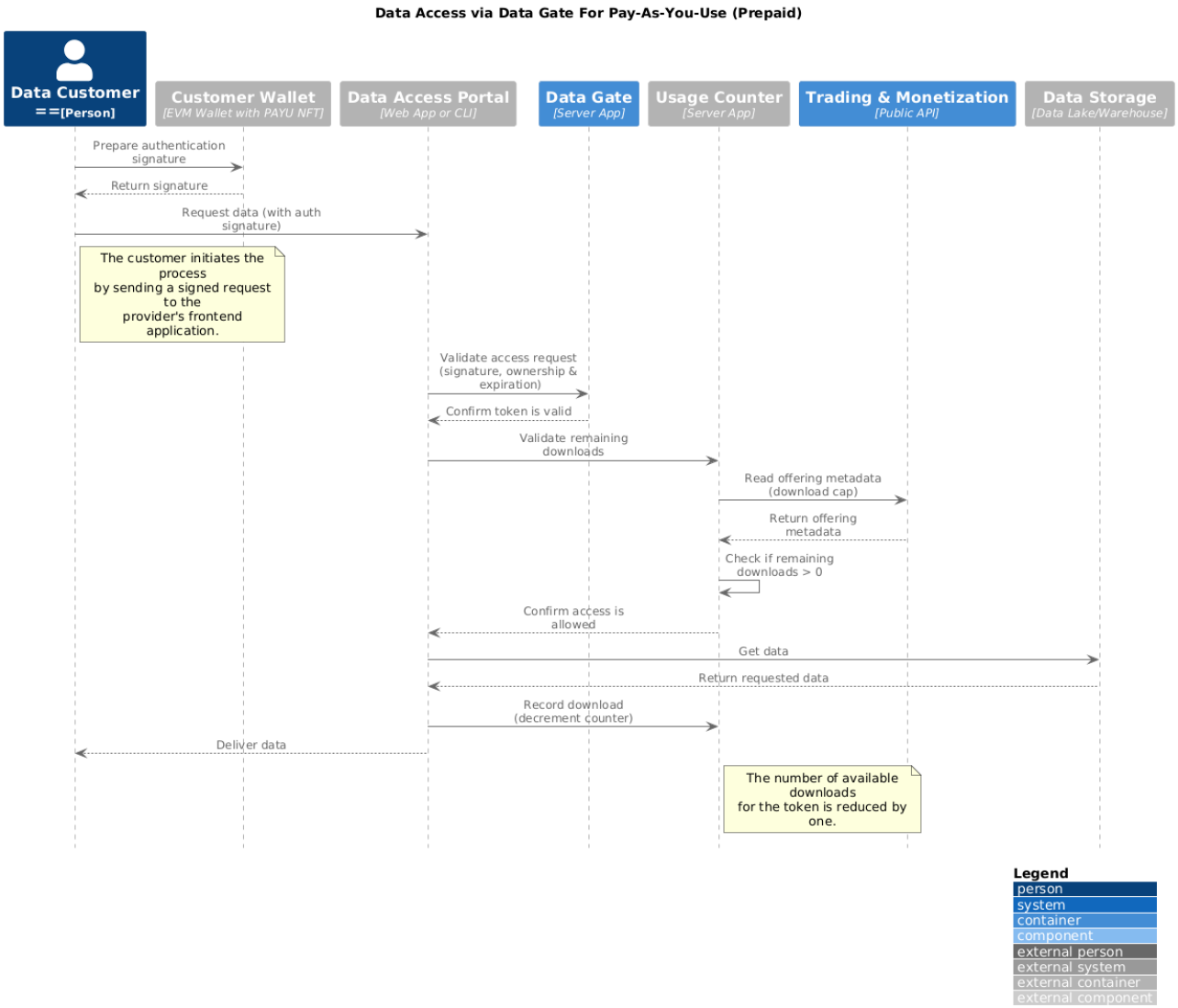


Figure 61 :Data Access via Gate for Pay-As-You-Use

Workflow Steps:

1. The customer authenticates.
2. The Token Gate validates the request.
3. The Usage Counter validates that the customer has remaining downloads/credits by checking the offering's metadata on the FAME platform.
4. If access is allowed, the Data Access Portal retrieves and delivers the data.
5. The Data Access Portal instructs the Usage Counter to decrement the remaining credits.

6 FAME CI/CD Integration Infrastructure

6.0 Introduction

This section explores the complexities of combining various software components, or packages, created from different project tasks within the FAME Solution Architecture. Several key challenges emerge:

- **Technology diversity:** Integrating modules built using disparate technologies necessitates a flexible solution that can accommodate different systems.
- **Data interoperability:** Seamless data transfer between these diverse modules is crucial, requiring a non-centralized architecture that promotes information sharing.
- **Modular design:** A modular platform structure is essential for adaptability, scalability, and efficient development.
- **Governance and DevOps:** A robust governance framework, coupled with DevOps practices like Continuous Integration/Continuous Deployment (CI/CD), is vital for effective implementation, automation, integration and quality assurance.

The FAME Solution Architecture highlights the significant challenges posed by integrating modules developed using varying technologies. To overcome these obstacles, a multifaceted approach is required. This includes a solution capable of handling diverse components, ensuring smooth data flow through a non-monolithic structure, and adopting a modular platform design. By implementing a comprehensive governance framework and leveraging DevOps methodologies, FAME development and integration was streamlined, with enhance quality, and has delivered a flexible, scalable solution.

6.1 Microservices Approach

The microservices approach is a software development methodology where applications are built as a collection of small, independent, and loosely coupled services. Each service is responsible for a specific business function and can be developed, deployed, and scaled independently.

The microservices approach promotes agility, scalability, and resilience in software development, making it well-suited for building complex and rapidly evolving applications.

This methodological change has both advantages and disadvantages. The advantages can be summarized as:

- **Simple to develop**: Each microservice is independent and small.
- **Simple to upgrade**: Since each microservice is independent it’s possible to upgrade each component independently.
- **Simple interaction**: Each microservice communicates with each other through well-defined and standard interfaces (API).
- **Simple to scale**: Each microservice can be scaled independently.
- **Scale on demand**: It is possible to run multiple microservices behind a load balancer to scale on demand request.
- **Technology Diversity**: Microservices can be built using different programming languages, frameworks, and databases, depending on the specific requirements of each service.

The disadvantages of the methodological change can be summarized as:

- **Complexity**: Splitting an application into multiple independent microservices increases the complexity of the deployment process.
- **Monitoring**: Monitoring each microservice requires having many metrics and logs to manage it.
- **Performance**: All communications occur on the network so these are slower than memory communications.
- **Debugging**: A Monolithic application is much easier to debug and test due to the fact it is composed by single indivisible units.

6.1.0 Microservices with Containers

The spread of containers led, as already mentioned, to the necessity to change the approach of the developers in the creation of an application, moving from design of monolithic applications, where the various components (generally UI, Business logic and Data-layer) were strongly coupled among them, to microservices applications where the various components are decoupled from each other.

Figure 62 : Monolithic vs Microservice

Our approach for a rapid iterative development of a microservice based software infrastructure builds upon a widely used methodology like DevOps, as will be described in Chapter 6.

In the last years there has been a strong transformation in conceptualizing benefits of containers, similar to what happened in the early 2000s with the advent of virtualization, due to containers' spread which led to rethinking both how to manage the infrastructure and how to design and build the applications.

The introduction and the wide spread of containers concept and technologies have made it possible to improve the computational management of infrastructures, thanks to the possibility of removing the overhead generated by the use of the hypervisor (integrated software that allows to virtualize the HW resources of a server and make them available among several applications) and through the usage of the functionalities already available within the Linux OS kernel, as in Figure 63.

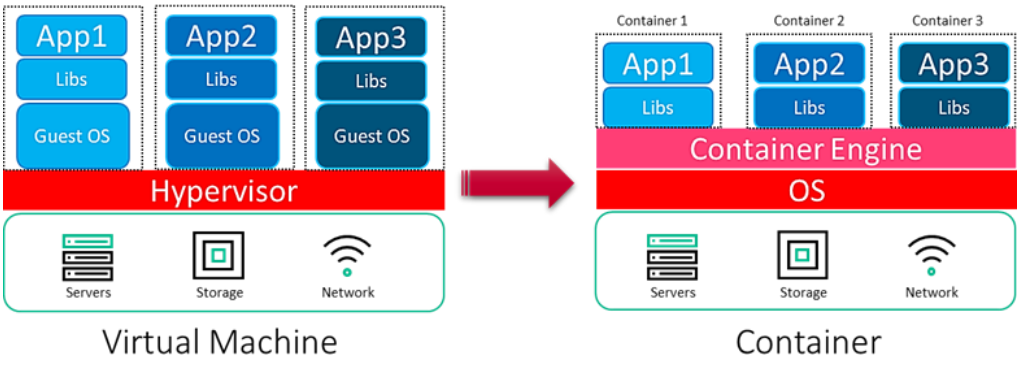


Figure 63 : VM vs Container

Docker, a widely used container engine, leverages Linux Control Groups and Linux Namespaces to create isolated container environments similar to virtual machines (VMs).

- **Linux Control Groups** allocate specific amounts of system resources—such as CPU, memory, disk I/O, and network—to each container, preventing resource exhaustion and ensuring fair sharing.
- **Linux Namespaces** isolate containers from each other and the host system by providing distinct views of system resources like process IDs, network stacks, and file systems.

This combination of technologies enables Docker to efficiently run multiple containers on a single host without compromising security or performance. An example is given in the figure below.

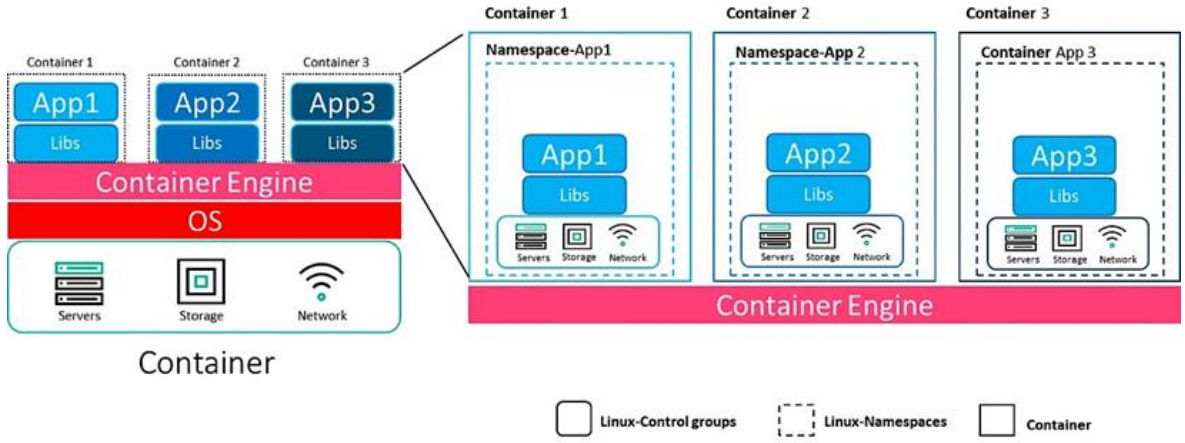


Figure 64: Container kernel properties

The main advantages to using containers to respect VMs can be summarized in the following macro points:

- Size: A container is small.
- Overhead: No full OS is required.
- Speed: Boot time is faster.
- Scaling: Real time provisioning.

Overall, container architecture offers many benefits, including improved portability, scalability, efficiency, and consistency for deploying and managing modern applications.

At the same time, to take advantage of containers it is necessary to rethink and redesign the currently monolithic applications as microservices based applications.

6.1.1 Kubernetes Containers Orchestration

The arrival on the market of containers and related microservices based applications, on the one hand enabled applications that quickly scale according to the requirements and that could be easily updated, on the other hand meant that software previously managed as a single indivisible piece was split into several dozen microservices (containers), making it more difficult to manage them.

In this context, the necessity to develop a tool that was able to manage the lifecycle of the microservices (deployment, scaling, and management) arose: such a tool was developed by Google with the name of "Project Seven of Nine "and released as open-source software in 2014. Today that tool is widely known as Kubernetes. For an outline of the entire Kubernetes solution, see the documentation on kubernetes.io.

The following bullet points are provided with the sole scope of highlighting the features Kubernetes provides to developers and integrators of the FAME platform using the FAME CI/CD infrastructure.

- **Service discovery and load balancing**: Kubernetes can expose a container using the DNS name or using its own IP address. If traffic to a container is high, Kubernetes is able to load-balance and distribute the network traffic so that the deployment is stable.
- **Storage orchestration**: Kubernetes allows to automatically mount a storage system of different types, such as local storages, public cloud providers, and more.
- **Automated rollouts and rollbacks**: a developer can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, a developer can automate Kubernetes to create new containers for deployment, remove existing containers and adopt all their resources to the new containers.
- **Automatic bin packing**: Kubernetes works with a cluster of nodes that are used to run containerized tasks. Kubernetes is configured with how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.
- **Self-healing**: Kubernetes restarts containers that fail, replaces containers, kills containers that do not respond to your user-defined health check, and does not advertise them to clients until they are ready to serve.
- **Secret and configuration management**: Kubernetes stores and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. Secrets and application configuration can be deployed and updated without rebuilding container images, and without exposing secrets in the stack configuration.

6.1.1.0 Kubernetes architecture

A Kubernetes (aka **K8s**) cluster is made up of two macro blocks, the first one called Control Plane (Master) and the second one called Data Plane (Worker).

The Control Plane constitutes the brain of the cluster and internally it is made up of the following components:

- **Kube-APIserver** is the component that exposes cluster API and truly is the main component, since Kubernetes has been designed and built to base all the operations on the use of the API.
- **Etcd** is a key value database that maintains all information relating to the status of the cluster.
- **Kube-scheduler** schedules which nodes of the Data Plane runs the containers (in Kubernetes named POD, the smallest deployable units of computing that can be created and managed in Kubernetes) based on the resources required, cluster status and affinity and anti-affinity rules.
- **Kube-controller manager** consists of a set of control processes which:
    - Check if cluster nodes are active.
    - Check if number and status of running POD is aligned with the required one, in case some POD became unavailable for example.
    - Control and create tokens to access on the K8s resource.
    - Populates the Endpoints object (that is, joins Services & PODs).

The Data Plane is the part where the workload is carried out, i.e. where the PODs are put into execution and it is characterized by the following components:

- **Kube-proxy** is a proxy that allows communication to the PODs from within and outside the cluster.
- **Kubelet** checks that PODs are running.

- **Container runtime (engine)** is software responsible for running containers; Kubernetes can use any CRI-O implementation like: Docker, CRI-O and containerd.

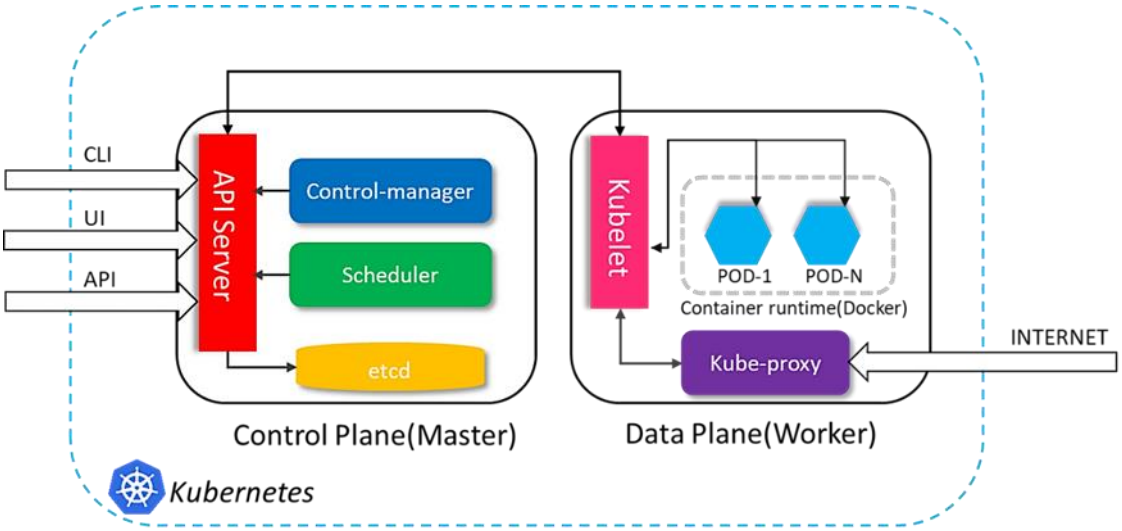


Figure 65 : Kubernetes

For the purpose of this chapter it is sufficient to outline two main concepts available in Kubernetes that we will use later to implement the Sandbox concept in the FAME integration process.

1. **Namespaces**: They are a logical grouping of a set of Kubernetes objects to whom it's possible to apply some policies, in particular:
   - **Quote** sets the limits on how many HW resources can be consumed by all objects.
   - **Network** defines if the namespace can be accessed or can access to other Namespaces, in other word if the Namespace is isolated or accessible.

   Different policies can be given to different namespaces.

2. **POD**: This is the simplest unit in the Kubernetes object. A POD encapsulates one container, but in some cases (when the application is complex) a POD can encapsulate more than one container. Each POD has its own storage resources, a unique network IP, access port and options related to how the container/s should run.

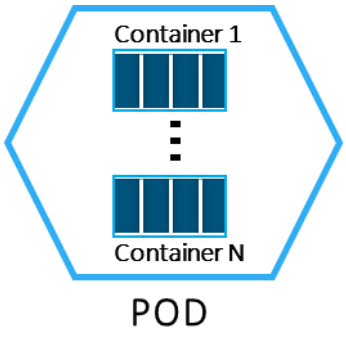


Figure 66: Graphical view of a POD in Kubernetes

6.2 Development View

The FAME CI/CD infrastructure with its components and relationships is depicted in the following figure.

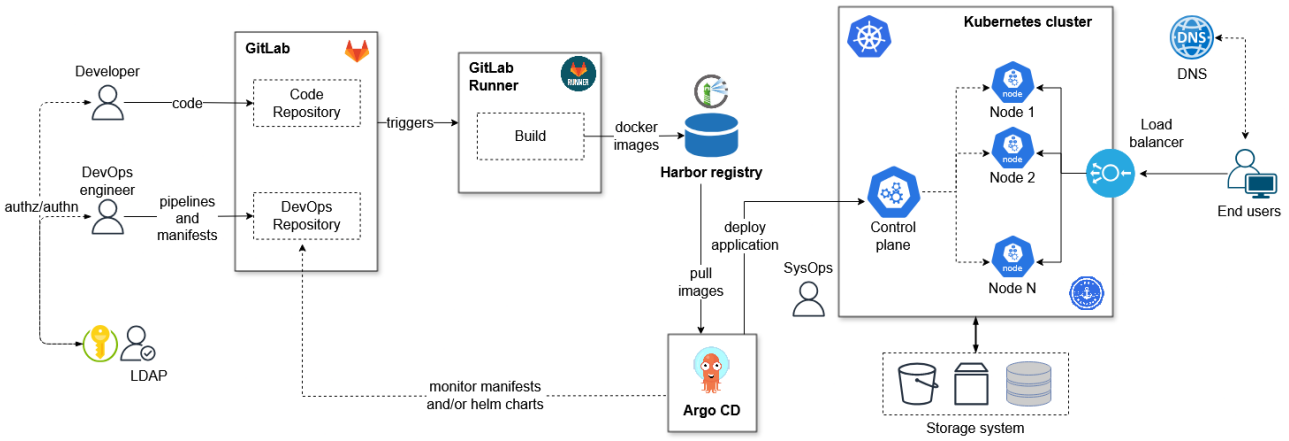


Figure 67: FAME CI/CD Infrastructure - Logical View

In the current implementation, the Kubernetes cluster is composed of 1 control plane node and 6 worker nodes. The cluster uses a storage system for the data persistence of the applications. This system can use different storage types like: object stores, block devices, network drives, file systems, etc.; the storage can be assigned to a specific node or shared among the cluster nodes.

The FAME CI/CD Solution uses the following application/tools:

- **GitLab**, the Source Code Version application, used to store and version the application source code. The CI/CD pipelines are stored in GitLab as well; pipelines are executed in GitLab using specific *runners*.
  A runner is basically a process which is launched to run the pipeline jobs, and is terminated when the jobs are completed.
- **Harbor**, the docker-compatible image registry, used to push and pull application images, maintaining version history.
- **Argo CD**, a declarative GitOps continuous delivery tool for Kubernetes. It automates the deployment process of applications, by synchronizing the current live state of the cluster with the desired one. The desired state is described in specific manifests, in a declarative way. For FAME, a specific and dedicated Argo CD Project is defined for applications deployments management.
- **LDAP server**, used to store user credentials and for SSO access to the other applications like GitLab and Harbor
- **Kubernetes Dashboard**, to view resources in the cluster and useful for troubleshooting and debugging, providing access to the logs of the application Pods
- **Cert-Manager**, a certificate management tool used to manage, generate and automatically renew the SSL certificates provided to secure the applications exposed to Internet

To develop and maintain the applications in the FAME CI/CD platform, three different roles are required:

- **Developer**, which is responsible for the application code development, testing, and versioning in the proper repositories. The developer also has the task of writing the Dockerfile of the application to be implemented and deployed
- **DevOps Engineer**, which is responsible for writing the CI/CD pipelines in GitLab, as well as the Kubernetes manifest files or Helm charts.

- **SysOps Engineer**, which is in charge of the Kubernetes cluster and DevOps tools maintenance, as well as other administrative and operational tasks like resources configuration, namespaces quota and limits settings, etc.

Developers and DevOps engineers are the main actors involved in the implementation of the CI/CD pipeline. They must be registered in the platform with email and password, and these credentials are stored in a LDAP server. The registration process of the users in the platform is called *onboarding*.

The user access to the other DevOps applications, like GitLab, is guaranteed by the same credentials stored in the LDAP server.

6.3 Deployment View

In the FAME CI/CD Architecture, each application is deployed in a Kubernetes cluster in its own namespace, in order to achieve a good degree of isolation from other applications. Deploying applications in different namespaces allows a better control of resources in terms of CPU, memory and storage assigned to each namespace. In fact, it's possible to set limits and assign resources quotas for each namespace.

This organization model for the applications also facilitates monitoring and troubleshooting. If necessary, complex applications could be deployed in more than one namespace, but the isolation principle between namespaces remains valid.

The following picture illustrates the isolation principle between namespaces.

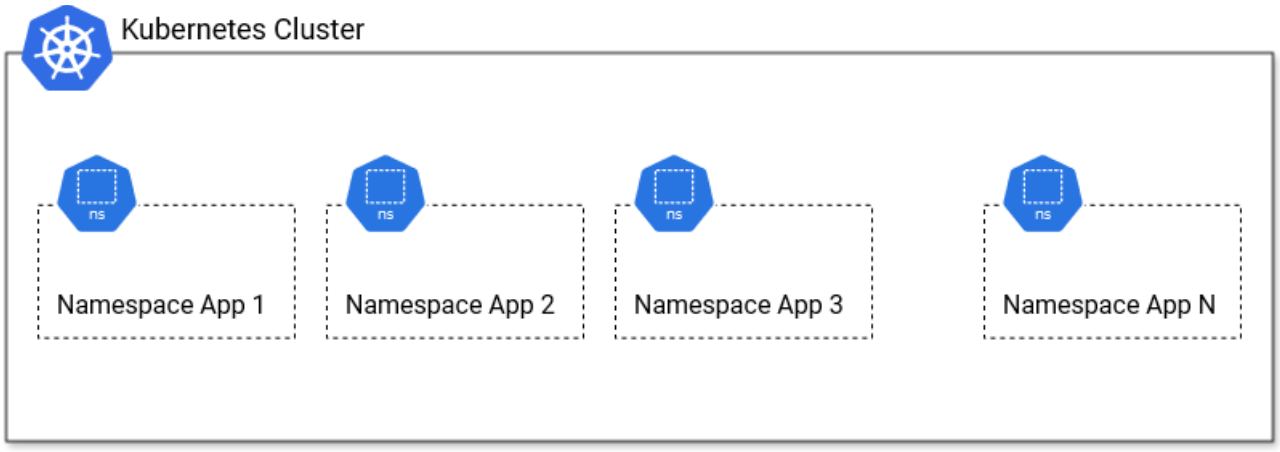


Figure 68: Kubernetes Cluster

Argo CD is the tool in charge of the continuous deployment phase (CD) of the CI/CD workflow. Once the application is properly built by the GitLab pipeline and the application images are pushed in the registry, Argo CD manages the deployment in the cluster, aligning the live state of the application with the desired state, described in the manifest files stored in the specific GitLab folder of the application repository.

This way, each change in the code or configuration, which is committed in the source code repository in GitLab, produces the building and the deployment of a new version of the application in the cluster. No manual tasks are needed in this phase; however, when necessary, GitLab also allows to manually rerun pipelines, either completely from the beginning or specific stages only.

Basically, the deployment can be performed in two different ways: either by applying Kubernetes manifest files, or by installing helm charts. Argo CD is able to automatically manage both methods without manual adjustment.

6.3.0 Exposing the Application

To expose applications and services to Internet, several components are involved in the platform.

A specific Kubernetes Ingress resource must be created and configured for each application or service. The Ingress resource uses a specific and well-known Ingress Class for all the applications (pilots), independently of the namespace in which they are allocated.

A dedicated Load Balancer is used to manage traffic among different applications and services, through the Ingress component installed in the cluster.

SSL certificates must be created and installed for each application or services to allow secure communications. The component involved in the creation and management of the certificates is the Cert-Manager application above mentioned.

Finally, a DNS record must be created and configured for the application; this record points to the cluster load balancer. The load balancer recognizes the host requested and route the traffic to the proper application or service.

6.3.1 Storage & Volumes Management

A given amount of storage can be set for each namespace, by configuring specific resources in the Kubernetes cluster, called PersistentVolumes. This kind of resource is bound to a physical storage asset, and at the same time provides access to this storage into the namespace. This approach allows the deployment of stateful applications like databases.

If an application requires a given amount of storage, the following steps must be performed:

- first, a PersistentVolume resource must be created in the cluster, referencing a physical storage asset having the same amount of space.
- a PersistentVolumeClaim resource must be defined in a manifest file to be deployed, referencing the volume created above. This resource will be created during the deployment of the application
- a Pod which needs to access the storage must specify the claim resource created above. In this way, the Pod can access the required storage.

Developers must provide any storage requirements for their applications. Volumes resources are created in the cluster by SysOps engineers, while DevOps engineers are in charge of declaring the claim and proper references for Pod resources in the manifest files. The following picture illustrates how the above resources are organized in the cluster.

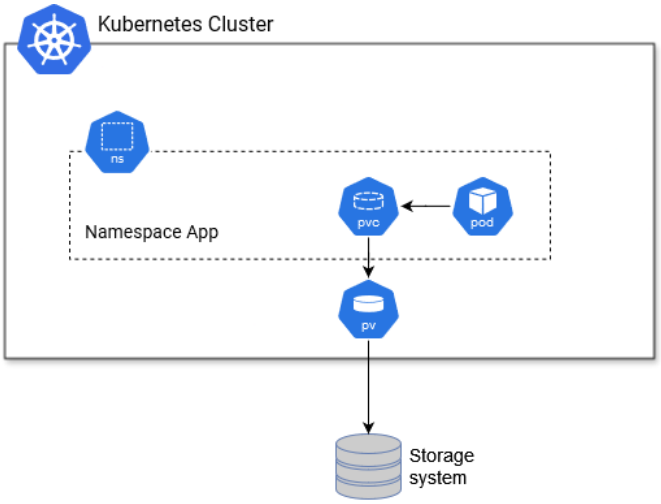


Figure 69: Kubernetes Persistent Volumes

7 Blueprint Guidelines for FAME Deployment

7.0 Guidelines Overview

This chapter describes the process and guidelines for the development and the deployment of an application using the FAME CI/CD infrastructure with the DevOps principles. It is provided with the main purpose to help the developers of FAME components, such as the dashboard, the backend, analytics, and technology components, the know-how to use the infrastructure to integrate the applications in a common namespace.

The following picture depicts the typical CI/CD workflow with the main actors (developer, DevOps engineer, end-user) involved.

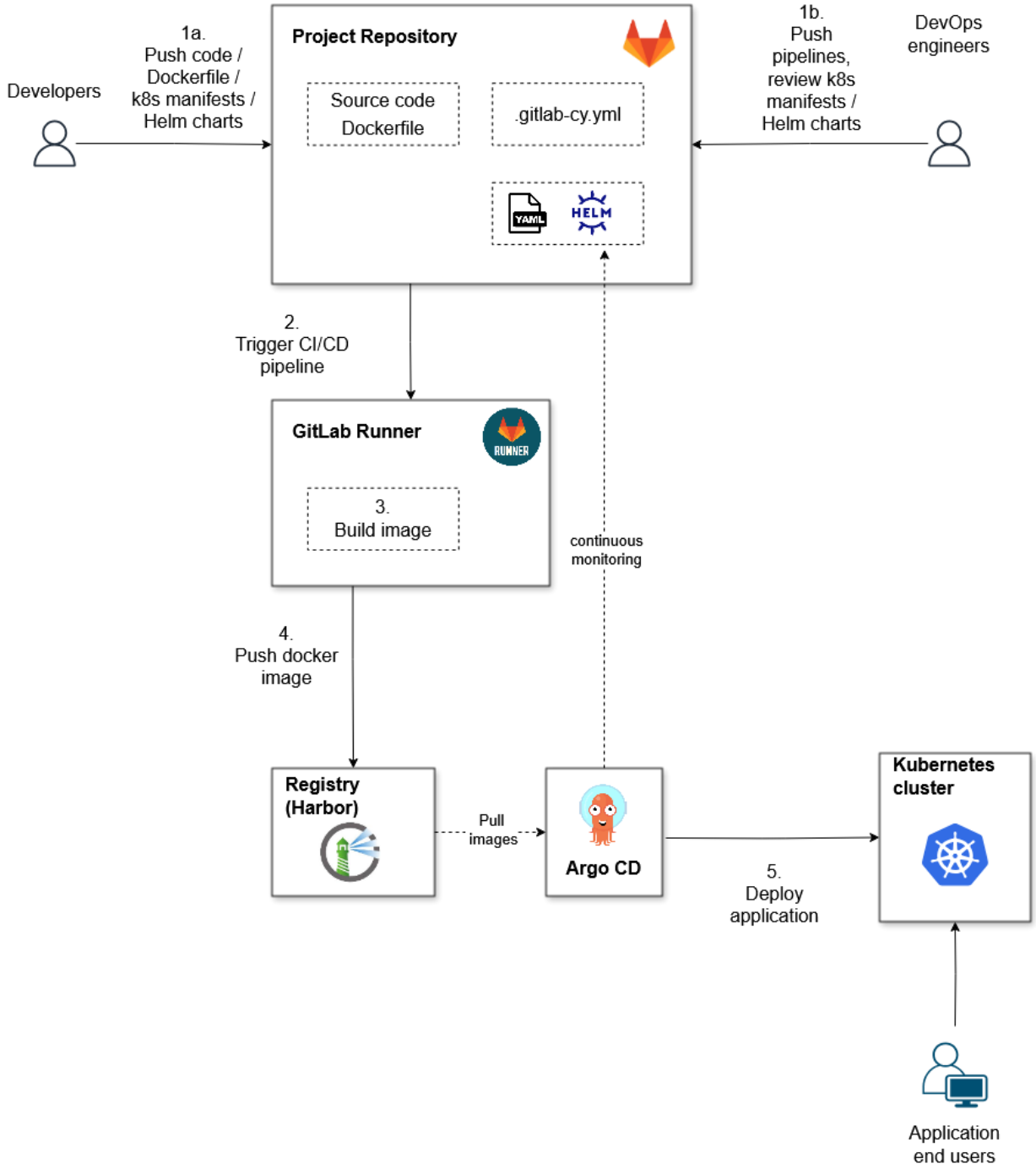


Figure 70: FAME CI/CD Workflow

7.0.0 Development Cycle

With respect to the diagram above, the development cycle consists of the following steps:

(1a.) A Project Repository is created on GitLab for each application or service to be developed. The developer uploads the application code on the repository and prepares the Dockerfile, which will be used to produce the docker image in a later step. The code, including Dockerfile and all material necessary to build the application, is committed to the GitLab repository. The developer is also in charge of writing Kubernetes manifests and/or Helm charts, which are used to deploy the applications in the cluster.

(1b.) A DevOps engineer writes the CI/CD pipeline code in a specific file located in the repository, (named *.gitlab-cy.yml* by default). The pipeline code can be organized in several stages, executed in the order specified, and contains at least all the instructions necessary to build the application docker image (the *build* stage). Other stages can also be included for specific purposes like tests, code quality checks, etc. The DevOps engineer should review Kubernetes manifests or Helm charts to optimize cluster resources usage or other parameters (based on any information provided by SysOps).

(2.) As soon as the code is committed to the GitLab repository, the pipeline is automatically triggered, and a process (called GitLab runner) is instantiated to start the execution of the stages indicated in the pipeline.

(3.) As mentioned before, the GitLab pipeline mandatorily contains the *build* stage of the CI/CD workflow (i.e. the CI part of the workflow). In this stage, the Dockerfile is used to produce the docker image of the application or service. Each docker image produced in this stage should be versioned, typically using a tag or a hash (sha256).

(4.) If the build stage is successful, the docker image produced is pushed into the docker image registry (the Harbor application), to be properly stored and versioned.

(5.) The CI/CD workflow proceeds with the *deploy* phase (the CD part), which involves the Kubernetes manifest files and/or Helm charts..

The deploy phase is orchestrated by Argo CD, which implements the GitOps *pull-based* approach. To manage deployments, Argo CD refers to the repository folder where the manifest files or Helm charts are stored. The files in this folder represent the desired state for the applications in the cluster. For the first deployment, Argo CD reads the manifest files and installs the applications and other various resources accordingly (e.g., Deployment, Statefulset, Service, ConfigMap, Ingress, etc.), pulling the necessary images from Harbor registry. Once the first deployment has been performed, Argo CD continuously monitors the manifest files folder and checks if there are any differences between the desired state of the applications and the actual state in the cluster. If there are any differences, Argo CD aligns the state of the cluster applications with the state described in the repository files. In this way, the repository acts as a *single source of truth* for the applications in the cluster.

The manifest files or the Helm charts are typically stored in a specific folder in the same GitLab project repository which contains the application code, and versioned along with the application code. Another approach could be to use a different repository than the application code; both these approaches are valid and have their pros and cons, depending on the use case.

The development cycle can therefore be summarized as follows:

- *Development*: developers write code locally using their preferred IDE or text editor. Developers are responsible for the local application code tests.
- *Version Control*: Application code is pushed to GitLab for version control and collaboration, both between developers on the same team and between different teams.
- *Continuous Integration* (CI): each time a change in the code is committed in the GitLab repository, a CI pipeline is automatically triggered.
- *Artifact generation* (aka *build*): successful CI pipeline execution produces deployable Docker images
- *Artifact repository storage*: Artifacts are stored in a repository in Harbor, and made available for deployment

7.1 Building a FAME Component

A FAME component, like an application or a service, is built using a CI/CD pipeline in GitLab. The pipeline scripts use so-called CI/CD variables, stored and configured in GitLab.

The use of variables has two main purposes:

- avoid hardcoding and allows reuse of configurations in different scripts pertaining to the same project, even if the code is located in multiple repositories
- store secret information that should not be shown in plain text

Currently, several CI/CD variables are defined and used in the build stage of the pipeline, as follows.

- **DOCKER_REGISTRY**, points to the Harbor URL and used to publish container images
- **DOCKER_USER**, the username for Docker login to Harbor.
- **DOCKER_PASSWORD**, the password for Docker login to Harbor

The DOCKER_USER and DOCKER_PASSWORD credentials are specific for CI/CD pipelines and different from user credentials, which are instead used for logging in to GitLab. The use of specific credentials for the pipeline allows better control of permissions, which can be established at project level and not at user level.

Moreover, because a user can belong to different groups and associated to multiple projects, establishing permissions at user level does not allow to specify different user permissions for different projects. Therefore, the permission model for the pipeline is a group/project based access control instead of a user based access control

7.2 Building an Image with Pre-compiled Libraries/Binaries

All necessary libraries should be imported in the application code, typically from external repositories, so that will be part of the built application image.

It’s also possible build an application image by referencing multiple images in the Dockerfile using specific features like Docker multi-stage build

In this case, the CI/CD pipelines can be still used to build the application as described before, using same variables as necessary.

8 FAME Scalability in the EU Strategy

8.0 Introduction

This section explores the potential for FAME to evolve in step with the latest regulatory and technological developments currently taking shape at the European level. It focuses on key initiatives such as eIDAS 2, the European Blockchain Services Infrastructure (EBSI), and the Digital Euro. These efforts are expected to play a central role in shaping the future of digital identity, decentralised data exchange, and digital finance. FAME is well-positioned to adapt to these changes, ensuring its continued relevance and alignment with the broader European digital strategy.

8.1 FAME & EIDAS Compliance

This section outlines the potential impacts of the EU Digital Identity (EUDI) requirements stemming from the eIDAS 2.0 regulation on the FAME marketplace, specifically possible required feature implementations and opportunities for the marketplace.

8.1.0 Background

The eIDAS regulation establishes a framework for electronic identification and trust services for electronic transactions in the European Union's internal market. It aims to enable secure and seamless electronic interactions between citizens, businesses, and public administrations across borders. Specifically, the regulation establishes a harmonized infrastructure for wallets and electronic attestations to be used in digital interactions between citizens and organisations.

The original idea of a harmonized digital identity scheme evolved into a much more powerful backbone for digital data exchange. It can be seen as an additional layer on top of the existing internet infrastructure to support appropriate identification, authentication, attestation exchange, and digital signature for the European digital ecosystem.

Member states must provide one or more EUDI wallets for their citizens and organisations (“for natural and legal persons”) from Jan 2027 on. It is important to notice that not only governmental issued identities but also any attestation from any organisation can be issued into the wallet and used for digital interactions. The features such a wallet must provide are Identification, Authentication, the issuance, storage, and presentation of (Qualified) Electronic Attestations, and the performance of Qualified Electronic Signatures.

Specifically, the timeline for eIDAS 2.0 is the following:

Q1 2023 - Large Scale Pilots were set up and funded by the European commission.

Q2 2024 - eIDAS 2.0 was published and entered into force.

Q4 2024 - wallet-specific implementation acts published by the European Commission that form the basis for the technical and legal implementation

Q1 2027 - Mandatory European Digital Identity Wallets will be rolled out by the member states

Q1 2028 - Banks, amongst others, mandated to accept EUDI credentials - otherwise fines up to €5M or 1% of revenue

Furthermore, it’s worth noticing that the legal adoption of the eIDAS takes place on two layers: i) the implementation of the eIDAS in the member states providing a wallet infrastructure and shaping their digital economies, and ii) integrate the wallet in central regulatory frameworks, e.g. in the upcoming Payment Service Regulation, the upcoming Digital Euro Regulation, the Financial Data Access

directive, and Digital Operational Resilience Act to ensure that the wallet can be used to facilitate controls to ensure legal compliance by providing a common set of technical solutions.

The eIDAS 2.0 is an ambitious initiative and requests a radical rethinking with respect to services in the European economy with possible huge impact with respect to the initial setup costs but also the existing business models. Therefore, the European Commission established a massive piloting setup, with hundreds of European companies preparing the new infrastructure. Four major projects - LSP Potential, European Wallet Consortium, NOBID, DC4EU - set up multiple project streams to investigate the implementation of the eIDAS and specifically the EUDI Wallet in various critical areas of the European Ecosystem and public administration:

EU Digital Identity Wallet Consortium (EWC): This consortium is working on developing a standard and interoperable EU Digital Identity Wallet.

Potential: This pilot project explores and demonstrates the potential applications of the EU Digital Identity Wallet across various sectors.

NOBID: The Nordic-Baltic eID Project (NOBID) focuses on cross-border eID solutions and their integration with the eIDAS 2 ecosystem, including testing wallet-based payment authorization flows.

DC4EU: DC4EU aims to develop decentralized identity solutions, particularly focusing on digital credentials in education and social security, aligning with the eIDAS 2 framework.

These projects ended in 2025 and are to be continued in two new projects WE BUILD and APTITUDE:

WE BUILD stands for “Wallet Ecosystem for Business and payments Use cases on Identification, Legal representation and Data sharing”.

APTITUDE is an acronym of Advanced Project for Trusted Identity Technologies and Unified Digital Ecosystem will target travel and payment use cases.

These enormous investments on both the European Commission and the European industry address a bold goal. 360 Mio. EU digital identities should use the EUDI Wallet by 2030 which is 80% of the European population.

The EUDI use cases are legally obliged to implement the EUDI Wallet infrastructure span 10 domains, such as payments, insurance, education, healthcare, travel, etc. Large enterprises, such as banks and payment service providers, have to accept identity credentials throughout their user journeys for various use cases related to identification, authentication, authorization, and e-signature. It is crucial to understand that supporting the wallet is key to make compliance implementations much more efficient and cost effective. EUDI can enable companies to satisfy various ongoing and upcoming EU payment regulations with a single technical infrastructure.

So, what is the European Identity Wallet from a business perspective? As a core functionality it is i) an always-available, instantly verifiable identity that works seamlessly across Europe with only a mobile phone or computer. Interoperability is ensured by the obligatory implementation of a common technical standard based on OpenID4VC. It is ii) A secure personal vault that securely stores credentials - which can be qualified (QEAA) with a high level of assurance of the credentials legal binding attestation, or non-qualified (EAA) – letting users easily share credentials in a privacy-preserving manner. And finally, it is iii) simplifying all digital interactions, making complex identification, authorization, authentication - as many activities as keys on a keychain.

8.1.1 Potential Impacts on the FAME Project

Whereas there is no obligation for citizens to use the wallet, some organisations are obliged to accept the wallet from natural and legal persons, specifically if they have to apply strong user authentication due to regulatory obligations.

In scope is any organisation from the banking domain, health sector, public administration, energy suppliers, water suppliers, telecommunication providers, and Very Large Online Platforms (VLOPs) under the Digital Services Act. Very Large Online Platforms are in scope if they provide services to more than 45 mio users per month, which may apply to marketplaces servicing business customers and aggregate through their customers that number of end users.

Specifically, the FAME Marketplace may move into the scope of eIDAS obligations if it integrates into the customer facing channels of its eIDAS-regulated clients, or if it becomes a Very Large Online Platform (VLOPs) under the Digital Services Act.

There may also be opportunities, specifically regarding the user/client interactions with and on the marketplace: the FAME Marketplace may leverage the features of the EUDI Wallet (personal AND business) for a better user experience and efficiency (see Sphereon integration).

More opportunities may be related to the technical core architecture of the marketplace: the FAME Marketplace may integrate the EUDI Business Wallet functionality to manage data exchange over EUDI protocols.

As a summary, the eIDAS regulation's requirements for electronic identification and trust services may impact the FAME project in the following areas:

User Authentication: The project may need to comply with specific standards for electronic identification and authentication of users, especially when handling sensitive data or providing access to secure services. The EUDI Wallet provides identification and authentication functionalities based on the government issued PID (Personal Identity Data) or Electronic Attestation Attributes (EAA credentials) potentially issued by the FAME Marketplace or a trustworthy partner.

Cross-border Recognition: If the FAME project involves interactions with users or entities from different EU member states, it may need to ensure that electronic identification means issued in one member state are recognized and accepted in others. The assurance of interoperability of the EUDI wallet can be leveraged to gain a pan-European authentication solution, which saves the FAME marketplace from providing such a service itself to its clients.

Data Security and Privacy: The project must adhere to eIDAS principles for data security and privacy, particularly when processing personal data related to electronic identification. The privacy-preserving features of the EUDI wallet help to protect the personal data of users but also to protect business clients not exposing too much data to central data repositories. It’s a helpful tool to minimize data in circulation.

Trust Services: The project may need to integrate or rely on qualified trust services, such as electronic signatures, electronic seals, and electronic time stamps, for certain operations or transactions. This is specifically of importance to legal binding procedures as digital signatures, providing proof and evidence, and reliable audit procedures.

To ensure alignment with eIDAS requirements, the FAME initiative should evolve along the following path:

- Review the eIDAS regulation and related guidelines to understand the applicable obligations.

- Conduct a thorough assessment of the project's authentication, data security, and trust service requirements.
- Create a service design to leverage the EUDI wallet functionalities in favour of the marketplace: Identify and implement necessary technical and organizational measures to comply with eIDAS standards.
- Seek legal counsel to ensure compliance and address any specific legal questions.

Further assessment and discussions will be required to fully address the potential impacts and necessary actions related to the eIDAS regulation. Also, because the regulatory but also the technical standards are in the making in 2025 and 2026.

8.2 European Blockchain

8.2.0 The European Blockchain Services Infrastructure (EBSI)

The European Blockchain Services Infrastructure (EBSI) is an initiative by the European Commission and the European Blockchain Partnership (EBP), comprising 29 European countries. Its core purpose is to leverage blockchain technology to deliver secure, trusted, and efficient cross-border services for public administrations, businesses, and citizens across the European Union.

EBSI aims to foster trust and transparency in digital interactions by providing a pan-European decentralized network that ensures the authenticity and integrity of data without reliance on central authorities. Strategically, EBSI is envisioned as the first EU-wide public sector-driven blockchain infrastructure, adhering to European values and regulations, particularly concerning data privacy and sovereignty. This foundation is critical for accelerating the digitalization of cross-border services and establishing a trusted digital environment throughout Europe.

Key services offered by EBSI include:

- **Verifiable Credentials (VCs):** Enabling the issuance, management, and verification of digital documents and qualifications (e.g., diplomas, professional certifications, social security records) that are tamper-resistant and verifiable across borders using Decentralized Identifiers (DIDs). This empowers individuals with self-sovereign control over their data.
- **Trusted Data Exchange:** Facilitating secure and auditable exchange of sensitive data between authorized parties, ensuring data integrity and provenance.
- **Track and Trace:** Providing capabilities for immutable recording of product lifecycles and supply chain events, exemplified by the concept of Digital Product Passports.
- **Legal Entity Verification:** Offering a robust mechanism for verifying the legal status and attributes of organizations.

EBSI's architecture is built on a distributed network of blockchain nodes across Europe. It supports a multi-DLT approach, notably utilizing **Hyperledger Besu** with IBFT 2.0 consensus, and **Hyperledger Fabric**. This design allows for pluggable protocols, ensuring flexibility and adaptability. Users interact with EBSI services, such as managing their DIDs and VCs, through secure digital wallets that prioritize privacy and user control, ensuring personal data is not stored on the ledger itself.

8.2.1 Similarities Between FAME and EBSI

Despite their distinct primary domains, FAME and EBSI share fundamental principles and technological foundations that highlight their inherent compatibility and potential for synergy:

- **Shared Blockchain Technology:** Both FAME and EBSI leverage enterprise-grade blockchain platforms, specifically utilizing **Hyperledger Besu**. This common technological backbone facilitates potential integration by reducing technical incompatibilities and allowing for a shared understanding of network mechanics, smart contract execution, and consensus mechanisms.
- **Goals for Secure and Transparent Digital Interactions:** At their core, both initiatives strive to enhance trust, transparency, and security in digital environments. FAME aims for secure and auditable pricing, trading, and monetization of digital data assets, ensuring clear ownership and transaction history. Similarly, EBSI focuses on establishing trust and verifiable authenticity for cross-border public services and digital credentials, mitigating fraud and increasing efficiency.
- **Decentralization Principles:** Both projects embrace decentralized models to achieve their objectives. FAME operates as a "federated marketplace," empowering data providers to maintain control over their assets and infrastructure while leveraging shared blockchain-based access rights. EBSI, with its network of distributed blockchain nodes across participating European countries, embodies decentralization by removing reliance on single central authorities for verification and trust.
- **Focus on Trust and Provenance:** Provenance and an immutable record of events are critical to both. FAME employs a "Blockchain-based Data Provenance Infrastructure" to record ownership, payments, and trading operations on the blockchain, providing a clear audit trail for data assets. EBSI similarly utilizes its ledger to create trusted, tamper-resistant digital audit trails for verifiable credentials and supports "Track and Trace" capabilities for provenance in various sectors, underpinning the integrity of shared information.
- **Permissioned Network Approach:** Both FAME and EBSI operate on permissioned blockchain networks. This choice generally offers advantages in terms of higher transaction throughput, enhanced data privacy (as participants are known), and more robust governance structures compared to public, permissionless blockchains. This shared architectural choice simplifies the establishment of trust frameworks and regulatory compliance.

8.2.2 Potential Options to Use EBSI with FAME for Scalability

Leveraging EBSI's established infrastructure and services can significantly enhance FAME's scalability, cross-border reach, and trustworthiness within the European digital landscape. Here are several potential options:

Integrating with EBSI's DIDs and VCs:

FAME can adopt EBSI's Decentralized Identifiers (DIDs) and **Verifiable Credentials (**VCs) to enhance identity verification, authentication, and authorization processes for participants in its federated data marketplace.

- **Enhanced Identity Verification:** Data providers and consumers within FAME could use EBSI-issued Verifiable Credentials (e.g., legal entity status, professional licenses, certifications related to data quality or compliance) to verify their identities and qualifications. This would add a layer of robust, EU-level trusted verification beyond FAME’s internal Federated AAI Infrastructure, streamlining "Know Your Customer/Business" (KYC/KYB) processes.

- **Streamlined Onboarding:** By leveraging existing EBSI identities and credentials, onboarding new participants to the FAME marketplace can be made more efficient and secure, reducing the administrative burden and accelerating market participation.

8.2.2.0 Leveraging EBSI's Ledger for Data Asset Provenance at an EU Level

While FAME already has a blockchain-based provenance infrastructure, integrating with EBSI's ledger for key provenance events can enhance traceability, trust, and auditability at a broader EU level.

- **Enhanced Traceability and Public Trust:** Recording critical provenance events (e.g., initial registration of a data asset, major ownership transfers, significant transformations, or quality certifications) on EBSI's public sector-backed ledger would provide an additional, immutable, and EU-verifiable layer of traceability. This enhances public trust and allows for easier verification by regulators, auditors, and end-users.
- **Interoperable Provenance Records:** Standardizing certain provenance records with EBSI's framework would enable greater interoperability with other EU initiatives or national systems that also utilize EBSI for traceability.
- **"Digital Data Asset Passports":** Extending the concept of EBSI's Digital Product Passports, FAME could create "Digital Data Asset Passports." These would leverage EBSI to record a comprehensive set of provenance data for a data asset, including its source, processing history, quality metrics, and compliance certifications, thereby creating a verifiable and comprehensive lifecycle record for high-value or regulated data.

8.3 FAME and the Digital Euro

8.3.0 Introduction

The European Central Bank (ECB) is actively pursuing the development of a Digital Euro, a project of strategic and geopolitical importance. This project is a key component of the evolving financial landscape within the Euro zone and the European Union globally.

In the proposed implementation, the **digital euro** is **central bank digital currency (CBDC)** issued by the European Central Bank (ECB) alongside euro banknotes and coins — not replacing them. It's essentially **digital cash**: a risk-free, public money equivalent to banknotes that can be used for digital payments, both online and offline. The key features, as publicly available from official documentation provided by ECB, include:

- **Free for users:** it will provide an EU IBAN free of charge
- **Universal acceptance** across the Euro Zone — usable in any shop or app that accepts digital payments.
- **Offline capability**, allowing payments without internet access.
- **Private but not anonymous**: digital euro payment will respect privacy with less traceability for small payments.
- **Guaranteed value** — always worth exactly one euro, backed by the ECB

It differs from cryptocurrencies or stablecoins: there's **no exchange rate risk**, **no speculative value**, and it's not controlled by private banks or commercial firms — it's a **public good, issued and secured by the Eurosystem**.

Historical Context and Ongoing Experimentation

The ECB's engagement with a potential Digital Euro dates back to 2018, when it started an initial exploration phase. This ongoing research and development phase aims to thoroughly assess the technical feasibility, practical implications, and potential benefits of a central bank digital currency for the Eurozone.

Regulatory Progress and Imminent Launch:

Significant progress is being made on the regulatory front. It is anticipated that the legal framework for the Digital Euro likely will receive approval from the EU Parliament within 2025. Following this crucial legislative step, the launch of the Digital Euro is expected to be imminent, marking a new era in European payments.

GFT's Contribution to Digital Euro Experimentation:

In a recent development, GFT, a prominent technology and consulting company, has actively participated in the experimentation phase of the Digital Euro. Their involvement specifically focused on the implementation of a Payment System leveraging a Digital Euro Application Programming Interface (D€ API) for conditional payments. This collaboration highlights the practical strides being made in developing real-world applications and functionalities for the future Digital Euro.

8.3.1 Digital Euro integration in FAME

A solution for converting real money (i.e., Euro) into FDE ("Fame Digital Euros") within the marketplace is comprehensively outlined in Deliverable D4.3, specifically within section 3.2.4, titled "Procurement of FDE coins."

However, this proposed mechanism, as detailed in the deliverable, necessitates a process managed by humans (payer, payee, admins, etc.). It starts with **a ticket raised by a user to the FAME marketplace administrator** who manages the transformation from real money to FAME digital tokens. This implies a controlled and auditable procedure, ensuring that the conversion of traditional fiat currency into FDE is not an instantaneous, unmonitored action, and rather requires administrative oversight and approval. Such a system would likely involve verification steps and potentially designated points of contact to manage the flow of funds and maintain system integrity within the FDE ecosystem.

With **the Digital Euro**, in its current design, this operation can be managed through an intermediary third party who acts as a Payment Service Provider (PSP). Below, there is the description of how Digital Euro can be integrated in FAME.

1. The Federation Application provides PSP services:

The Federation Application is responsible for the comprehensive management of the Digital Euro (D€) accounts for all participating organizations within the Federation. A key structural element is that each organization operating within this Federated system will have a unique European Union International Bank Account Number (EU IBAN) directly associated with its Federation association. This linkage ensures traceability, compliance with existing financial regulations, and seamless integration with the broader European banking infrastructure.

2. Exchange Mechanism: Digital EURO with FDE (Acquire/Redeem):

A fundamental feature of this system is the bidirectional exchange capability. Organizations are empowered to exchange their Digital Euros for Federated Digital Euros (FDE) and vice versa. This process encompasses both the 'acquiring' of FDE (converting Digital EURO into FDE) and the 'redeeming' of FDE (converting FDE back into Digital EURO. This flexibility is crucial for maintaining liquidity and enabling organizations to manage their digital assets efficiently based on their operational needs within and outside the Federation.

3. Digital Euro management in the FAME EU Bank Account:

To allow transactions with FDE, Digital EURO purchases of FDE are securely held in a dedicated FAME EU Bank Account. This centralisation of money in a regulated bank account provides the layer of trust and stability, ensuring all FDE circulating in the FAME Marketplace are backed by an equivalent amount of Digital EURO. This design mitigates risks associated with unbacked digital currencies and aligns with established financial principles.

4. Users' Transactions:

Within this framework, end-users are able to conduct transactions using the FDE. While the document primarily focuses on organizational functions, it implies that once organizations have acquired FDE, they can facilitate various payment scenarios for their customers or internal operations. These transactions would leverage the established FDE network, offering a new means of digital payment within the specified ecosystem.

5. Organization Redemption of FDE into D€ via FEDERATION:

Finally, to complete the cycle and ensure full interoperability, organizations can redeem anytime their FDE back into Digital EUROS, through the FEDERATION PSP services. This redemption process reinforces the convertibility of FDE and provides a clear off-ramp for organizations that may need to convert their digital holdings back into the central bank digital currency for broader use or settlement

outside the immediate Federated system. This final step underscores the full lifecycle management of digital assets within this innovative framework.

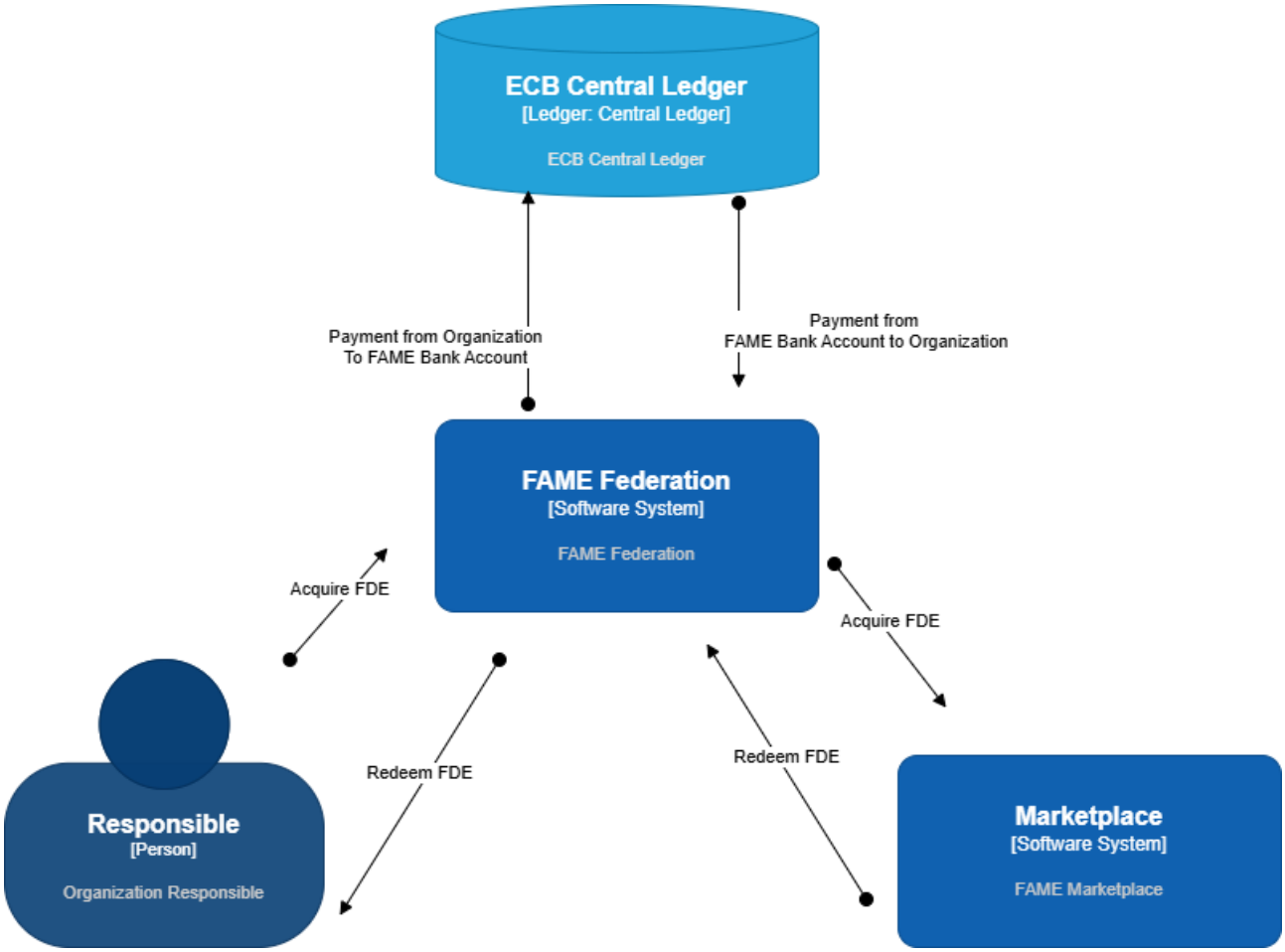


Figure 71 : Logical Flow

8.3.2 Design implementation hypothesis

These two sequence diagrams highlight the two main operations performed in the Digital Euro Wallet.

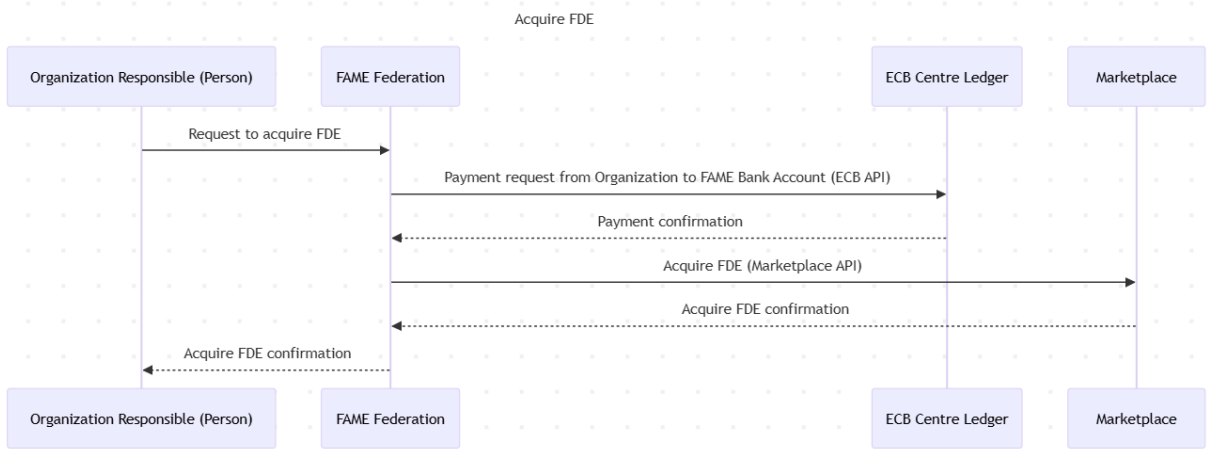


Figure 72: Acquire sequence diagram

Figure 72: Acquire sequence diagram illustrates the procedural interaction between four key entities involved in the process of acquiring Federated Digital Euro (FDE): the **Organization Responsible**, the **FAME Federation**, the **ECB Central Ledger**, and the **Marketplace**. The diagram outlines the end-to-end flow from the initial request to the final confirmation, highlighting the transactional logic and inter-system communication that support the operation.

The process begins when the Organization Responsible—typically a user or entity seeking to obtain FDE—initiates the workflow by submitting a request to the FAME Federation. This request expresses the intent to acquire a specific quantity of FDE and marks the entry point into the coordinated interaction between the participating systems.

Upon receiving the acquisition request, the FAME Federation proceeds to initiate the necessary financial transaction. It communicates with the **ECB Central Ledger**, invoking the appropriate API to request a payment transfer from the Organization's account to the designated FAME bank account. This step ensures that the financial commitment is fulfilled prior to the issuance of the digital asset. Once the ECB confirms that the payment has been successfully processed, the FAME Federation is authorized to continue the operation.

Following payment confirmation, the FAME Federation engages with the **Marketplace**, leveraging its API to formally acquire the corresponding amount of FDE. This interaction ensures that the digital assets are provisioned through the standardized marketplace mechanisms, maintaining transparency, traceability, and alignment with federated procedures. Upon successful acquisition, the Marketplace responds with a confirmation, indicating that the FDE tokens have been successfully allocated.

Finally, the FAME Federation communicates back to the originating Organization with a confirmation of the completed acquisition. This closing interaction signals that the requested FDE has been successfully obtained and that all underlying transactions—both monetary and digital—have been executed in accordance with the defined protocol

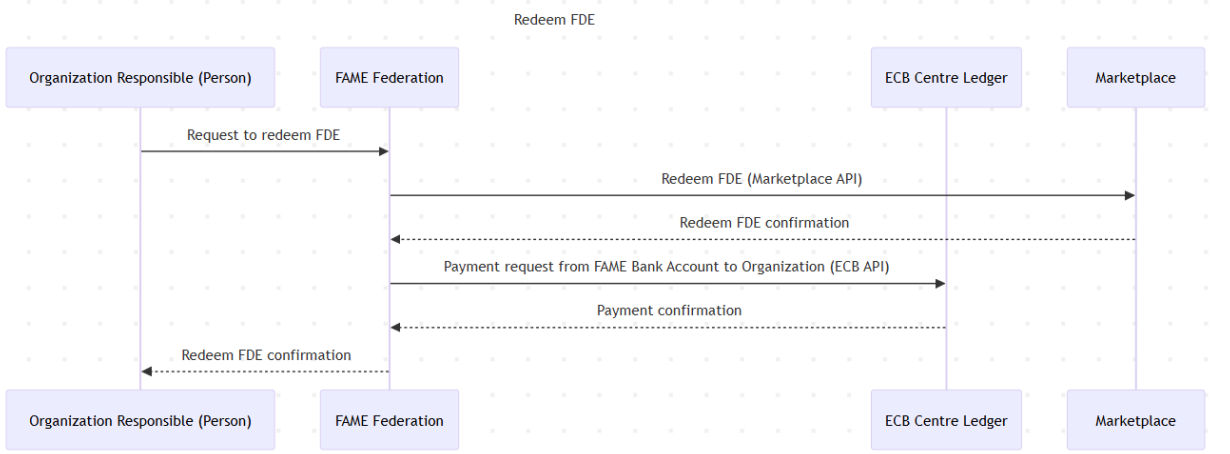


Figure 73: Redeem sequence diagram

Following the process of acquiring FDE, the reverse operation—**Redeeming FDE**—is equally structured and governed by clear transactional steps. The diagram outlines how an organization initiates the redemption of previously acquired FDE, resulting in the return of the corresponding monetary value.

The sequence begins with the organization submitting a **redemption request** to the FAME Federation. This signals the intent to convert a specific amount of FDE back into euros. Upon receipt of the request, the FAME Federation coordinates directly with the Marketplace through its API to execute the redemption. This ensures that the tokens are correctly and securely removed from the organization's digital balance.

Once the Marketplace confirms that the redemption has been completed, the FAME Federation proceeds to initiate the **financial settlement**. It contacts the ECB Central Ledger, instructing a transfer of funds from the FAME bank account back to the organization. This transaction mirrors the earlier acquisition flow, reinforcing a full-cycle logic of traceability and accountability.

Upon confirmation from the ECB that the payment has been executed, the FAME Federation delivers a final acknowledgment to the organization. This final step confirms that the FDE has been successfully redeemed and that the corresponding euro amount has been reimbursed.

Together with the acquisition workflow, this process completes the lifecycle of FDE within the platform, offering users a transparent, reversible, and institutionally integrated mechanism for managing digital currency.

8.3.3 Federation PSP Mockup User interface

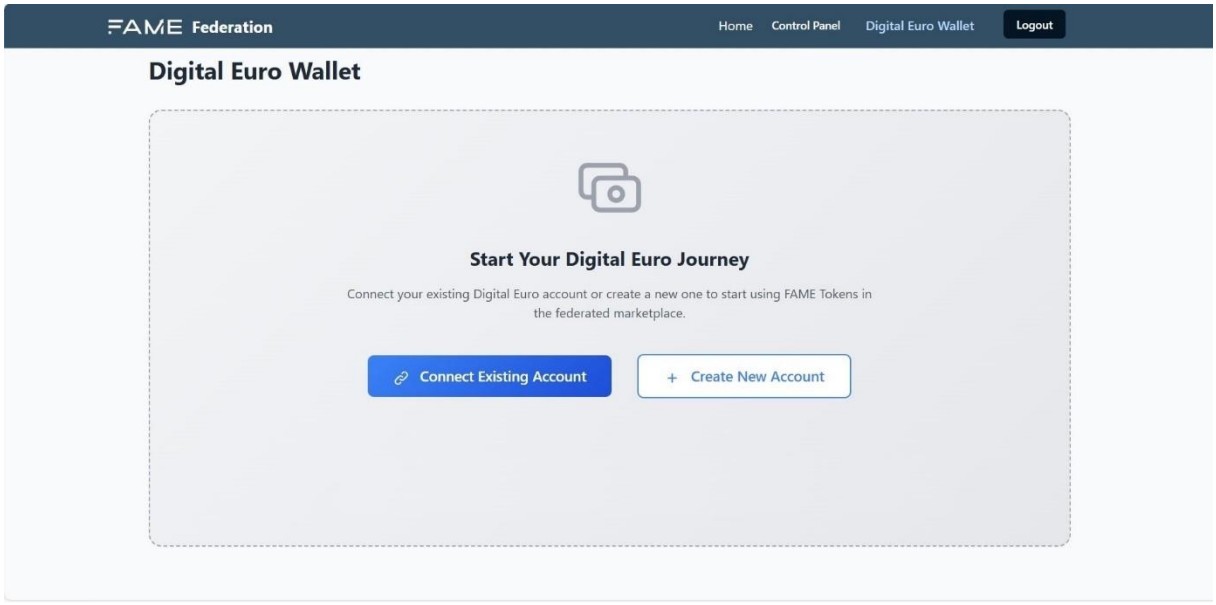


Figure 74: Login page

The user experience design for the FAME Digital Euro integration prioritizes seamless integration with the existing Federation platform while introducing powerful new financial capabilities. The implementation follows established FAME design principles, ensuring consistency with the broader ecosystem while addressing the specific requirements of digital currency management.

User Journey Implementation

The Digital Euro user journey begins within the familiar environment of the Federation Dashboard, where organization representatives access their existing profile and administrative functions. The integration point is strategically positioned within the "My Profile" section of the Federation Dashboard, providing a natural pathway to wallet functionality without disrupting established user workflows.

When an organization representative navigates to the wallet section for the first time, they encounter a streamlined onboarding experience that clearly communicates the value proposition of Digital Euro integration. The system presents two primary pathways: connecting an existing Digital Euro account or creating a new one through the DESP service. This approach accommodates organizations at different stages of Digital Euro adoption while maintaining consistency with the Federation's inclusive philosophy.

The account creation process leverages the established relationship between the organization and the Federation platform. Rather than requiring complex verification procedures, the system utilizes existing organizational credentials and verification status to streamline the Digital Euro account provisioning. This approach significantly reduces friction while maintaining the security and compliance standards required for financial services.

Once the Digital Euro account is established, the user experience transitions to a comprehensive wallet dashboard that serves as the central hub for all digital currency operations. The dashboard design maintains visual consistency with the Federation platform while introducing specialized financial interface elements that clearly communicate account status, transaction capabilities, and operational controls.

Wallet Dashboard Design

The wallet dashboard represents a sophisticated balance between financial functionality and user accessibility. The interface design follows the established Federation visual language while incorporating specialized elements necessary for financial operations. The primary dashboard view presents a clear hierarchy of information, with balance displays, transaction controls, and historical data organized in a logical and intuitive manner.

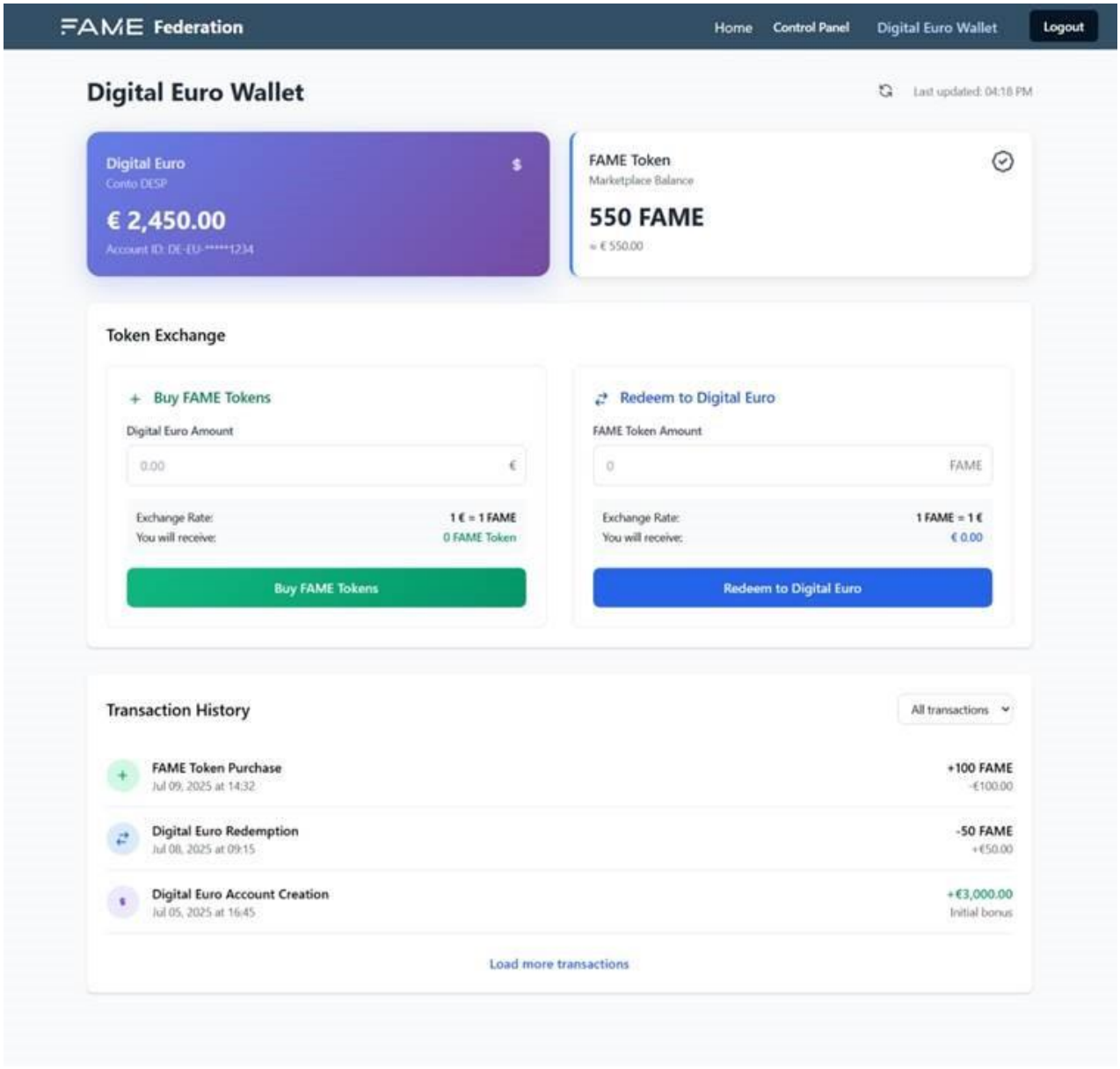


Figure 75: Wallet Page

The balance presentation distinguishes between Digital Euro holdings and FAME Token balances, providing users with immediate visibility into their available resources across both currency types. The interface design employs visual cues and color coding to differentiate between currency types while maintaining accessibility standards. Real-time balance updates ensure that users always have current information about their financial position.

The exchange interface represents the core functional element of the wallet dashboard, providing users with straightforward controls for converting between Digital Euro and FAME Token currencies. The design emphasizes clarity and transparency, with real-time exchange rate displays and clear

confirmation of transaction details before execution. The interface includes preview functionality that allows users to understand the exact outcome of proposed transactions before commitment.

Transaction history integration provides users with comprehensive visibility into their Digital Euro and FAME Token operations. The historical view includes detailed transaction records with timestamps, amounts, and operation types clearly identified. The interface design supports filtering and sorting capabilities, enabling users to quickly locate specific transactions or analyze patterns in their digital currency usage.

Transaction Management

The transaction management system prioritizes immediacy and reliability, with Digital Euro to FAME Token exchanges processed instantaneously upon user confirmation. This approach eliminates the complexity of pending states while ensuring that users receive immediate feedback on their operations. The system architecture supports this instantaneous processing through direct integration with both the DESP service and the FAME marketplace infrastructure.

The exchange process begins with user input specifying the desired transaction amount and direction. The interface provides immediate validation of available balances and exchange rates, ensuring that users have complete information before proceeding. The confirmation process includes a clear summary of the transaction details, including the exact amounts involved and the resulting balance changes.

Upon transaction execution, the system provides immediate confirmation through dashboard updates and notification messages. The balance displays refresh automatically to reflect the new financial position, while the transaction history updates to include the completed operation. This immediate feedback cycle ensures that users maintain complete visibility into their financial operations without delays or uncertainty.

The marketplace synchronization occurs seamlessly through API integration, ensuring that FAME Token balances are immediately available for use in marketplace transactions. This integration eliminates the need for manual reconciliation or separate balance management, providing users with a unified view of their digital assets across the entire FAME ecosystem.

Error handling within the transaction management system focuses on clear communication and user guidance. When insufficient balances or other operational constraints prevent transaction completion, the system provides specific error messages that guide users toward resolution. The interface design ensures that error conditions are immediately apparent without compromising the overall user experience.

The transaction management system also incorporates comprehensive logging and audit capabilities, ensuring that all operations are properly recorded for compliance and troubleshooting purposes. These backend capabilities operate transparently to users while providing the necessary infrastructure for system reliability and regulatory compliance.

User support for wallet and transaction operations integrates with the existing FAME support infrastructure, providing users with access to comprehensive help resources and assistance channels. The support system includes detailed FAQ sections and direct access to support personnel through established communication channels, ensuring that users can resolve issues quickly and efficiently.

The overall user experience design recognizes that Digital Euro integration represents a significant expansion of FAME platform capabilities while maintaining the approachable and professional interface standards that define the Federation platform. This balance ensures that users can

confidently engage with advanced financial functionality while operating within a familiar and trusted environment.

8.3.4 Advantages and Disadvantages

Advantages in adopting Digital Euro for FAME Marketplace

- Digital Euro is a strategic European project and FAME wants to support it from the beginning
- Digital Euro solves the problem of exchange of real money with the FAME token used in the marketplace
- Keeping separate Digital Euro to FAME token transactions is a more robust and safer mechanism

Disadvantages

- Digital Euro is not ready yet and there is the risk that it will not be implemented (low risk)
- The actual implementation of the DE in the FAME marketplace will only be a Proof of Concept and when it will be deployed by ECB there could be more development to the ultimate technology framework.

9 Conclusions

As previously mentioned, FAME aims to be a realized Data Space, specifically a Finance Data Space, one of the types of Data Space defined by the EU among the Common European Data Spaces [4].

Compared to traditional market players, FAME, as a Data Space, aims for a lighter version of implementation constraints for the valuation, sharing, and commercialization of digital assets (Assets). In fact, no specific implementation constraints are imposed on the exchange of data.

FAME's federated model aims to guarantee the rules, principles, and control of participation, safeguarding the values of commercial exchange, rather than imposing the creation of a mere interchange platform through connectors. This is to allow asset producers the freedom to decide how to provide/exchange assets with consumers, as long as they are acceptable within the federation.

The motivation is to enable even SMEs to participate in and benefit from the Data Economy in the Finance sector through FAME.

FAME focuses on the regulation of economic exchanges of assets, aiming to enrich the reference architectures of Data Spaces with value-added components for data assets, such as trading and monetization modules, which are currently missing. To achieve this objective, FAME aims to define three frameworks: a Business Framework, a Legal/Organizational Framework, and a Technological Framework for the realization of Data Spaces in the Finance sector.

This document outlines the integration architecture of the FAME Marketplace's technical framework, detailing its two core components: the Federation Application, which functions as the manager of the Data Space and the Marketplace platform, which is dedicated to the evaluation and trading of data assets (including the FAME Connector for secure, scalable, and transparent data consumption between data providers and consumers). This integration architecture allows the various technical modules of the solution to integrate, enabling interaction with users and integration with other systems/marketplaces/Data Spaces. The goal of this integration is to allow the inclusion of assets (digital assets of various types) in the FAME "showcase," enhancing their value through their integration, commercialization, and exchange. In this document, the term "asset" refers to digital assets of various types such as Data Assets, AI models, and static analytics or their runtime made available through services accessible to consumers.

In addition to the integration of the various components, the document describes the infrastructure on which the FAME Data Space will be built during the project phase. This DevOps infrastructure includes applications/tools for managing code or pre-existing work-in-progress (Docker images) and all deployment processes according to best CD/CD practices, as well as an environment for releasing the solution at runtime.

In this version, we have introduced a new section that delves into the potential for FAME to evolve in line with the latest regulatory and technological developments currently being shaped at the European level. Specifically, it considers key initiatives such as eIDAS 2, the European Blockchain Services Infrastructure (EBSI), and the Digital Euro. These initiatives are anticipated to play a pivotal role in defining the future landscape of digital identity, decentralised data exchange, and digital finance. FAME is strategically positioned to respond to these changes, ensuring its continued relevance and alignment with the broader European digital agenda.

References

1. European data strategy, https://digital-strategy.ec.europa.eu/en/policies/strategy-data
2. European Data Governance Act, https://digital-strategy.ec.europa.eu/en/policies/datagovernance-act
3. Data Act, https://digital-strategy.ec.europa.eu/en/policies/data-act
4. Common European Data Spaces, https://digital-strategy.ec.europa.eu/en/policies/data-spaces
5. FAME, D2.6 - Technical Specifications and Platform Architecture II (2024) https://zenodo.org/records/15341321
6. FAME, D2.5 - Requirements Analysis, Specifications and Co-Creation II (2024) https://zenodo.org/records/15341334
7. FAME, D3.5 - Federated Data Assets Catalogue II (2025)
8. FAME, D4.4 - Blockchain-based Data Provenance Infrastructure II (2025) https://zenodo.org/records/15341360
9. FAME, D4.5 - Pricing, Trading and Monetization Techniques II (2025) https://zenodo.org/records/15341421
10. FAME, D3.4 - Secure Federated Data Management II (2025) https://zenodo.org/records/15341348
11. FAME, D5.1 - Trusted and Explainable AI Techniques I (2024) https://zenodo.org/records/16620533
12. FAME, D5.2 - Energy Efficient Analytics Toolbox I (2024) https://zenodo.org/records/16620626
13. European AI Act, AI Act | Shaping Europe's digital future (europa.eu)
14. Octopus Visual Sitemap tool, https://octopus.do/
15. FIGMA, https://www.figma.com/
16. Angular, https://github.com/angular/angular/releases?page=5
17. Docker, https://www.docker.com/
18. HTML, https://html.com/html5/
19. SCSS, https://www.npmjs.com/package/sass/v/1.70.0
20. TypeScript, https://www.npmjs.com/package/typescript
21. Angular Material, https://v16.material.angular.io/
22. WSL, https://learn.microsoft.com/en-us/windows/wsl/install
23. Git, https://www.simplilearn.com/tutorials/git-tutorial/git-installation-on-windows
24. Node.js, https://nodejs.org/en/download/package-manager
25. Docker installation, https://www.docker.com/products/docker-desktop/
26. Node Version Manager, https://github.com/coreybutler/nvm-windows/releases
27. NPM, https://phoenixnap.com/kb/install-node-js-npm-on-windows
28. API-First approach, https://www.postman.com/api-first/
29. Postman, https://www.postman.com/
30. SwaggerHub, https://swagger.io/tools/swaggerhub/
31. Online Swagger editor, https://editor.swagger.io/
32. FAME, D3.4 - Secure Federated Data Management II
33. FAME, D3.6 - Mechanisms and Tools for Regulatory Compliance II