



Promoting and Incentivising Federated, Trusted, and Fair Sharing and Trading of Interoperable Data Assets

## D3.2

# Monetisation and Trading services – Alpha version

<b>Editor(s)</b>	Konstantinos Latanis, Sotiris Koussouris
<b>Lead Beneficiary</b>	SUITE5
<b>Status</b>	Final
<b>Version</b>	1.00
<b>Due Date</b>	30/04/2024
<b>Delivery Date</b>	03/05/2024
<b>Dissemination Level</b>	PU



Funded by the European Union under Grant Agreement 101093016. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

<b>Project</b>	PISTIS – 101093016
<b>Work Package</b>	WP3 - PISTIS FAIR Data Trading and Value Exchange/Monetisation Platform Services
<b>Deliverable</b>	D3.2 - Monetisation and Trading services - Alpha version
<b>Contributor(s)</b>	SUITE5, ICCS, UBITECH, EUT, ISI/ATHENA, SPH
<b>Reviewer(s)</b>	FHG, TRAF
<b>Abstract</b>	The deliverable D3.2 "Monetisation and Trading services – Alpha version" comprises a direct outcome of all Tasks of WP3, documenting the design and implementation of the alpha releases of the different components of the PISTIS FAIR Data Trading and Value Exchange/Monetisation Platform.

## Executive Summary

The deliverable D3.2 "Monetisation and Trading services - Alpha version" comprises a direct outcome of Task 3.3 "Data Valuation Assessment and Dynamic Pricing Services ", Task 3.4 "XAI driven Monetisation and Matchmaking Services", Task 3.5 "PISTIS Data Exchange Market, Crypto Operations and Wallets" and Task 3.6 "Immutable Blockchain Network for Data Trading Management and Operations Auditing", documenting the alpha releases of the different components of the PISTIS FAIR Data Trading and Value Exchange/Monetisation Platform (hereafter called "PISTIS Market Exchange Platform" for brevity reasons). More specifically, the first releases of the different components of the PISTIS Market Exchange Platform are described and the technology stack used for their development is provided. User stories covering the set of functionalities of each component and that drive the specification of the respective functional and non-functional requirements are also presented, alongside with the internal architecture of each component and screenshots of the UI of each component.

The PISTIS Market Exchange Platform enables the valuation of datasets and provide meaningful insights to data providers towards sharing their datasets through the creation of blockchain-enabled contracts to interested stakeholders by exploiting NFTs while maintaining the privacy and security of data and transactions. Eventually, it constitutes an exchange market for datasets that will bring together different organisations without the necessity for storing their data outside their premises.

The PISTIS Market Exchange Platform consists of components that belong to 4 different core bundles of the overall PISTIS Platform, as these are described in D4.1 "PISTIS Reference Architecture and API Documentation":

- a) Data Discovery bundle: Provides services for the exploration and discovery of the available datasets that may interest a Data Consumer. It includes the **Matchmaking Services** component of the PISTIS Market Exchange Platform, which proposes datasets to Data Consumers based on their previous searches and transactions.
- b) Data Monetisation bundle: Provides services for the Data Provider to understand the value of their data within the market environment of the platform and to place the data on the PISTIS Data Catalogue using different methods to monetise by engaging in transactions performed with the aid of blockchain technology. It includes the following components of the PISTIS Market Exchange Platform:
  - **Asset Description Bundler**: Puts together a package which fully describes the data asset that is placed for publishing or trading.
  - **FAIR Data Valuation Service**: Provides methods that enable the assignment of a quantitative value to a data asset.
  - **Digital DLT-FIAT Wallet**: Oversees the platform's value transactions in a DLT private network, guaranteeing quick and reliable exchanges.
  - **Data Investment Planner**: Enables the design of an investment plan related to a specific dataset of a Data Provider.
  - **PISTIS Market Insights**: Offers insights about the transactions performed in the PISTIS Marketplace, based on the analysis of the available transaction metadata.

- **NFT Generator:** Generates a data NFT for a given dataset.
  - **Monetisation Plan Designer:** Enables Data Providers to put their datasets into the PISTIS market and specify their desired monetization method.
  - **Data Usage and Intentions Analytics:** Captures the underlying intentions of data owners and users regarding the datasets they own or are interested in, respectively.
- c) **Ledgers bundle:** Provides services for the storage of the data and the monetary transactions. It includes the following components of the PISTIS Market Exchange Platform:
- **PISTIS Data Ledger:** Allows the storage of metadata of datasets transactions performed in the PISTIS Platform.
  - **PISTIS Monetary Ledger:** Allows the storage of monetary information of datasets transactions performed in the PISTIS Platform.
- d) **Transaction Services bundle:** Provides the services to create, execute and validate the different transactions with the use of smart contracts. It includes the following components of the PISTIS Market Exchange Platform:
- **On/Off Platform Contract Inspector:** Performs various checks to ensure that all transactions adhere to the established rules of the active licenses of datasets.
  - **Transaction Auditor:** Enables the monitoring of the validity of the transactions in PISTIS Platform.
  - **Smart Contract Template Composer:** Enables the selection of pre-defined contract templates and transforms them into the corresponding blockchain code to drive their execution.
  - **Smart Contract Execution Engine:** Realizes the execution of the smart contracts for the corresponding datasets transactions.
  - **Data Factory User Wallet:** Enables the secure management of the identity data and credentials of a Data Factory User.

Deliverable D3.2 provides an overview of the Alpha version of the components developed under WP3 "PISTIS FAIR Data Trading and Value Exchange/Monetisation Platform Services". These components can be found in the project's GitHub repository at: <https://github.com/orgs/PISTIS-Platform/>. This deliverable has been based on the D3.1 "Data Valuation, Sharing and Trading Framework" that had initially sketched the methodological and technological scenery of these components. Deliverable 3.2 has also received input from deliverable D1.2 "PISTIS Technical Requirements and MVP - v1" regarding the high-level user stories related to PISTIS Market Exchange Platform and is interconnected with deliverable D2.2 "Data Management and Protection services - Alpha version" for the interrelation with components of WP2 "PISTIS Data Spaces Factory and Trusted Data Management Services", and it will provide feedback to deliverable D4.2 "The PISTIS Product - Alpha version" towards the integration of the different components of the PISTIS Market Exchange Platform to the alpha release of the overall PISTIS Platform. It will also constitute the basis for the deliverable D3.3 "Release of the core PISTIS factory services dealing with data Management and protection, and of the Data Explorer Service - Beta version" for the design and development of the beta release of the PISTIS Market Exchange Platform, where the updates and new features of its different components will be documented.

## Table of Contents

- 1 Introduction ..... 11
  - 1.1 Document structure ..... 13
- 2 Data Discovery Bundle ..... 14
  - 2.1 Matchmaking Services ..... 14
    - 2.1.1 Component Description ..... 14
    - 2.1.2 Technology Background ..... 14
    - 2.1.3 Component Backlog ..... 14
    - 2.1.4 Functional Requirements ..... 15
    - 2.1.5 Non-Functional Requirements ..... 15
    - 2.1.6 Component’s Main Elements and Internal Architecture ..... 15
    - 2.1.7 Mock-ups and Screenshots ..... 16
- 3 Data Monetisation Bundle ..... 17
  - 3.1 Asset Description Bundler ..... 17
    - 3.1.1 Component Description ..... 17
    - 3.1.2 Technology Background ..... 18
    - 3.1.3 Component Backlog ..... 18
    - 3.1.4 Functional Requirements ..... 19
    - 3.1.5 Non-Functional Requirements ..... 19
    - 3.1.6 Component’s Main Elements and Internal Architecture ..... 20
    - 3.1.7 Mock-ups and Screenshots ..... 21
  - 3.2 FAIR Data Valuation Service ..... 22
    - 3.2.1 Component Description ..... 22
    - 3.2.2 Technology Background ..... 22
    - 3.2.3 Component Backlog ..... 22
    - 3.2.4 Functional Requirements ..... 23
    - 3.2.5 Non-Functional Requirements ..... 24
    - 3.2.6 Component’s Main Elements and Internal Architecture ..... 24
    - 3.2.7 Mock-ups and Screenshots ..... 25
  - 3.3 Digital DLT-FIAT Wallet ..... 27
    - 3.3.1 Component Description ..... 27
    - 3.3.2 Technology Background ..... 28
    - 3.3.3 Component Backlog ..... 29
    - 3.3.4 Functional Requirements ..... 33

- 3.3.5 Non-Functional Requirements ..... 34
- 3.3.6 Component’s Main Elements and Internal Architecture ..... 35
- 3.3.7 Mock-ups and Screenshots ..... 36
- 3.4 Data Investment Planner ..... 36
  - 3.4.1 Component Description ..... 36
  - 3.4.2 Technology Background ..... 36
  - 3.4.3 Component Backlog ..... 37
  - 3.4.4 Functional Requirements ..... 38
  - 3.4.5 Non-Functional Requirements ..... 38
  - 3.4.6 Component’s Main Elements and Internal Architecture ..... 38
  - 3.4.7 Mock-ups and Screenshots ..... 39
- 3.5 PISTIS Market Insights ..... 40
  - 3.5.1 Component Description ..... 40
  - 3.5.2 Technology Background ..... 40
  - 3.5.3 Component Backlog ..... 40
  - 3.5.4 Functional Requirements ..... 41
  - 3.5.5 Non-Functional Requirements ..... 41
  - 3.5.6 Component’s Main Elements and Internal Architecture ..... 41
  - 3.5.7 Mock-ups and Screenshots ..... 43
- 3.6 NFT Generator ..... 44
  - 3.6.1 Component Description ..... 44
  - 3.6.2 Technology Background ..... 44
  - 3.6.3 Component Backlog ..... 45
  - 3.6.4 Functional Requirements ..... 45
  - 3.6.5 Non-Functional Requirements ..... 45
  - 3.6.6 Component’s Main Elements and Internal Architecture ..... 45
  - 3.6.7 Mock-ups and Screenshots ..... 46
- 3.7 Monetisation Plan Designer ..... 46
  - 3.7.1 Component Description ..... 46
  - 3.7.2 Technology Background ..... 47
  - 3.7.3 Component Backlog ..... 47
  - 3.7.4 Functional Requirements ..... 48
  - 3.7.5 Non-Functional Requirements ..... 49
  - 3.7.6 Component’s Main Elements and Internal Architecture ..... 49

- 3.7.7 Mock-ups and Screenshots ..... 50
- 3.8 Data Usage and Intentions Analytics ..... 52
  - 3.8.1 Component Description ..... 52
  - 3.8.2 Technology Background ..... 52
  - 3.8.3 Component Backlog ..... 53
  - 3.8.4 Functional Requirements ..... 54
  - 3.8.5 Non-Functional Requirements ..... 54
  - 3.8.6 Component’s Main Elements and Internal Architecture ..... 55
  - 3.8.7 Mock-ups and Screenshots ..... 56
- 4 Ledgers Bundle..... 57
  - 4.1 PISTIS Data Ledger ..... 57
    - 4.1.1 Component Description ..... 57
    - 4.1.2 Technology Background ..... 57
    - 4.1.3 Component Backlog ..... 57
    - 4.1.4 Functional Requirements ..... 58
    - 4.1.5 Non-Functional Requirements ..... 58
    - 4.1.6 Component’s Main Elements and Internal Architecture ..... 59
    - 4.1.7 Mock-ups and Screenshots ..... 59
  - 4.2 PISTIS Monetary Ledger..... 60
    - 4.2.1 Component Description ..... 60
    - 4.2.2 Technology Background ..... 60
    - 4.2.3 Component Backlog ..... 60
    - 4.2.4 Functional Requirements ..... 62
    - 4.2.5 Non-Functional Requirements ..... 63
    - 4.2.6 Component’s Main Elements and Internal Architecture ..... 63
    - 4.2.7 Mock-ups and Screenshots ..... 63
- 5 Transaction Services Bundle ..... 64
  - 5.1 On/Off Platform Contract Inspector..... 64
    - 5.1.1 Component Description ..... 64
    - 5.1.2 Technology Background ..... 65
    - 5.1.3 Component Backlog ..... 66
    - 5.1.4 Functional Requirements ..... 66
    - 5.1.5 Non-Functional Requirements ..... 67
    - 5.1.6 Component’s Main Elements and Internal Architecture ..... 68

- 5.1.7 Mock-ups and Screenshots ..... 69
- 5.2 Transaction Auditor ..... 69
  - 5.2.1 Component Description ..... 69
  - 5.2.2 Technology Background ..... 70
  - 5.2.3 Component Backlog ..... 70
  - 5.2.4 Functional Requirements ..... 70
  - 5.2.5 Non-Functional Requirements ..... 70
  - 5.2.6 Component’s Main Elements and Internal Architecture ..... 71
  - 5.2.7 Mock-ups and Screenshots ..... 72
- 5.3 Smart Contract Template Composer ..... 72
  - 5.3.1 Component Description ..... 72
  - 5.3.2 Technology Background ..... 72
  - 5.3.3 Component Backlog ..... 73
  - 5.3.4 Functional Requirements ..... 73
  - 5.3.5 Non-Functional Requirements ..... 73
  - 5.3.6 Component’s Main Elements and Internal Architecture ..... 73
  - 5.3.7 Mock-ups and Screenshots ..... 74
- 5.4 Smart Contract Execution Engine ..... 74
  - 5.4.1 Component Description ..... 74
  - 5.4.2 Technology Background ..... 75
  - 5.4.3 Component Backlog ..... 75
  - 5.4.4 Functional Requirements ..... 76
  - 5.4.5 Non-Functional Requirements ..... 76
  - 5.4.6 Component’s Main Elements and Internal Architecture ..... 77
  - 5.4.7 Mock-ups and Screenshots ..... 77
- 5.5 Data Factory User Wallet..... 77
  - 5.5.1 Component Description ..... 77
  - 5.5.2 Technology Background ..... 77
  - 5.5.3 Component Backlog ..... 78
  - 5.5.4 Functional Requirements ..... 78
  - 5.5.5 Non-Functional Requirements ..... 79
  - 5.5.6 Component’s Main Elements and Internal Architecture ..... 79
  - 5.5.7 Mock-ups and Screenshots ..... 80
- 6 Conclusions ..... 81



## List of Figures

Figure 1-1: The architecture of PISTIS FAIR Data Trading and Value Exchange/Monetisation Platform within the overall PISTIS architecture .....	11
Figure 2-1: Matchmaking Service’s Internal Architecture .....	16
Figure 2-2: Matchmaking Services integrated with the PISTIS Data Catalogue UI .....	16
Figure 3-1: Asset Description Bundler’s Internal Architecture .....	20
Figure 3-2: Asset Description Bundler UI Mockup, illustrating the necessity of the ADB to collect information from the Data Valuation Services, Monetisation Plan, Access Policies and configure the Smart Contract.....	21
Figure 3-3: FAIR Data Value Service’s Internal Architecture.....	24
Figure 3-4: FAIR Data Valuation Service UI. Definition of Purpose, Business goals and Licenses. ....	25
Figure 3-5: FAIR Data Valuation Service UI. Definition of weights for the data value dimensions. ....	26
Figure 3-6: FAIR Data Valuation Service UI. Data Value Scorecard.....	27
Figure 3-7: Digital DLT-FIAT Wallet Internal Architecture.....	36
Figure 3-8: Data Investment Planner’s Internal Architecture .....	39
Figure 3-9: Data Investment Planner UI.....	39
Figure 3-10: PISTIS Market Insights’ Internal Architecture .....	42
Figure 3-11: Overview Dashboard of PISTIS Market Insights.....	43
Figure 3-12: Dashboard of Market Insights showing Top Performing Assets.....	43
Figure 3-13: View of Sector Sales per Week .....	44
Figure 3-14: NFT Generator’s Internal Architecture .....	46
Figure 3-15: Monetisation Plan Designer’s Internal Architecture .....	50
Figure 3-16: Monetisation Plan Designer – User’s owned datasets (main page).....	50
Figure 3-17: Monetisation Plan Designer – Dataset Selection.....	51
Figure 3-18: Monetisation Plan Designer – One off Sale .....	51
Figure 3-19: Data Usage and Intentions Analytics’ Internal Architecture .....	55
Figure 3-20: Questionnaire Builder .....	56
Figure 3-21: List of available Questionnaires .....	56
Figure 4-1: PISTIS Data Ledger’s Internal Architecture .....	59
Figure 5-1: On chain platform inspection .....	65
Figure 5-2: On Platform Contract Inspector’s Internal Architecture .....	68
Figure 5-3: On Platform Contract Inspector core entities.....	69
Figure 5-4: Transaction Auditor’s Internal Architecture .....	71
Figure 5-5: List of Transactions and Transaction IDs.....	72
Figure 5-6: Smart Contract Template Composer’s Internal Architecture.....	74
Figure 5-7: Smart Contract Execution Engine high-level Architecture .....	75
Figure 5-8: Smart Contract Execution Engine’s Internal Architecture .....	77
Figure 5-9: Data Factory User Wallet’s Internal Architecture.....	79

## Terms and Abbreviations

<b>ADB</b>	Asser Description Bundle
<b>AI</b>	Artificial Intelligence
<b>API</b>	Application Programming Interface
<b>CF</b>	Collaborative filtering
<b>CRUD</b>	Create, Read, Update, Delete
<b>DaVe</b>	Data Value Vocabulary
<b>DC</b>	Dublin Core
<b>DCAT</b>	Data Catalogue Vocabulary
<b>DID</b>	Decentralized Identifier
<b>DLT</b>	Distributed Ledger Technology
<b>DNN</b>	Deep neural networks
<b>DQV</b>	W3C Data quality Vocabulary
<b>DVDs</b>	Data Value Dimensions
<b>DVS</b>	Data Valuation Service
<b>EU</b>	European Union
<b>FAIR</b>	Findable, Accessible, Interoperable, Reusable
<b>GDPR</b>	General Data protection Regulation
<b>GNN</b>	Graph Neural Networks
<b>HTTP</b>	HyperText Transfer Protocol
<b>ID</b>	Identity
<b>IOTA</b>	Internet of Things Application
<b>JSON</b>	JavaScript Object Notation
<b>JWT</b>	JSON Web Token
<b>kNN</b>	k-Nearest Neighbour
<b>LoA</b>	Level of Assurance
<b>ML</b>	Machine Learning
<b>MMS</b>	Matchmaking Services
<b>MVP</b>	Minimum Viable Product
<b>NFT</b>	Non-Fungible Token
<b>ODRL</b>	W3C Open Digital Rights Language
<b>PROV</b>	Provenance
<b>PSD2</b>	Payment Services Directive 2
<b>RDF</b>	Resource Description Framework
<b>REST</b>	Representational state transfer
<b>SQL</b>	Structured Query Language
<b>SSI</b>	Self-Sovereign Identity
<b>VC</b>	Verifiable Credential
<b>WP</b>	Work Package
<b>XAI</b>	eXplainable AI

# 1 INTRODUCTION

The deliverable D3.2 documents the alpha releases of the different components of the PISTIS FAIR Data Trading and Value Exchange/Monetisation Platform, which enables the valuation, trading and monetization of datasets in PISTIS.

The functionalities of each component are described and the technologies that have been exploited for the development of the corresponding alpha releases are presented. The user stories for each component are specified and through them the functional and non-functional requirements of the different components are defined. Moreover, the internal architecture of each component is provided, showcasing the interconnections among the subcomponents of each component. For the components that have a user interface, respective screenshots depicting the components' functionalities are also presented.

The updates and new features of the different components of the PISTIS Market Exchange Platform that will be included in their beta and final releases, will be documented in the corresponding deliverables D3.3 and D3.4.

In the following image, the different highlighted bundles of the PISTIS FAIR Data Trading and Value Exchange/Monetisation Platform are depicted as part of the overall PISTIS Architecture, based on the designed architecture which is described in D4.1.

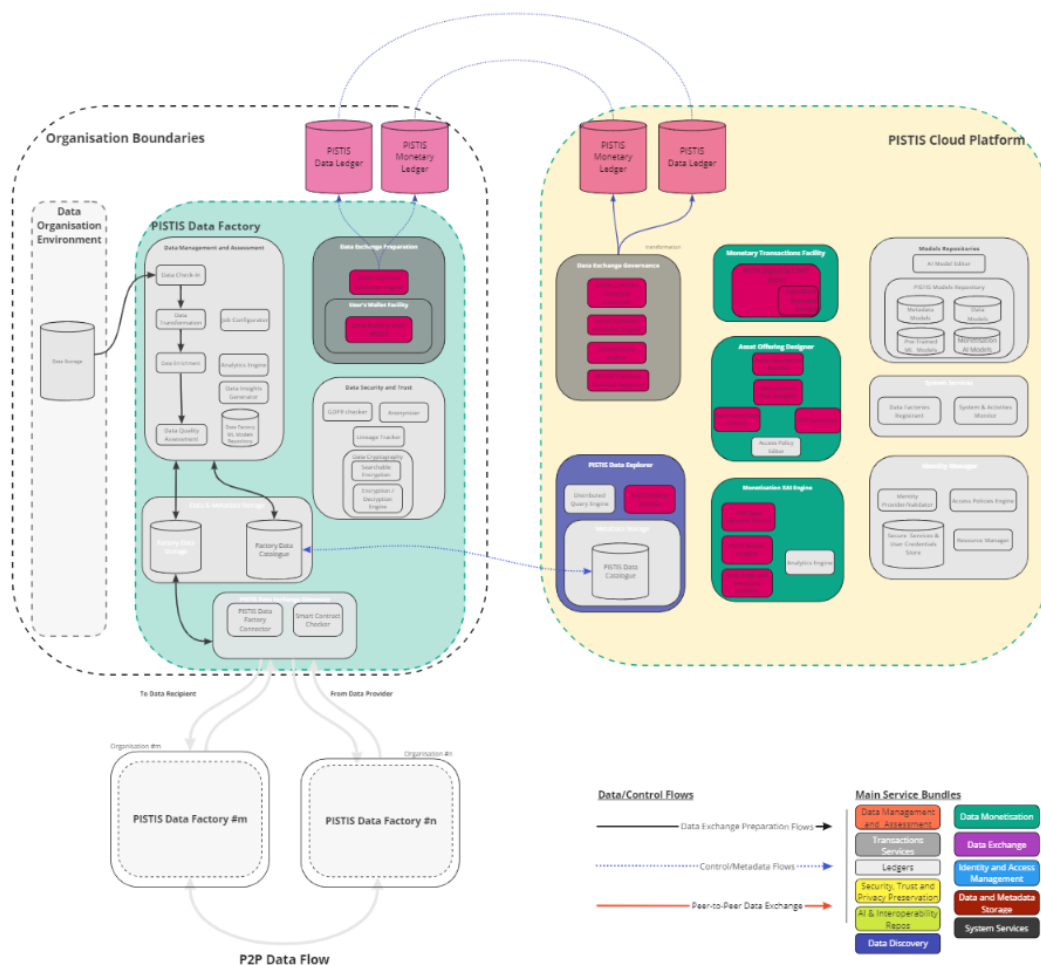


Figure 1-1: The architecture of PISTIS FAIR Data Trading and Value Exchange/Monetisation Platform within the overall PISTIS architecture

The PISTIS Market Exchange Platform consists of components that belong to 4 different core bundles of the overall PISTIS Platform, as follows:

- a) **Data Discovery bundle:** Provides services for the exploration and discovery of the available datasets that may interest a Data Consumer. It includes the **Matchmaking Services** component of the PISTIS Market Exchange Platform, which proposes datasets to Data Consumers based on their previous searches and transactions.
- b) **Data Monetisation bundle:** Provides services for the Data Provider to understand the value of their data within the market environment of the platform and to place the data on the PISTIS Data Catalogue using different methods to monetise by engaging in transactions performed with the aid of blockchain technology. It includes the following components of the PISTIS Market Exchange Platform:
  - **Asset Description Bundler:** Puts together a package which fully describes the data asset that is placed for publishing or trading.
  - **FAIR Data Valuation Service:** Provides methods that enable the assignment of a quantitative value to a data asset.
  - **Digital DLT-FIAT Wallet:** Oversees the platform's value transactions in a DLT private network, guaranteeing quick and reliable exchanges.
  - **Data Investment Planner:** Enables the design of an investment plan related to a specific dataset of a Data Provider.
  - **PISTIS Market Insights:** Offers insights about the transactions performed in the PISTIS Marketplace, by running designated analytics on the available transaction metadata.
  - **NFT Generator:** Generates a data NFT for a given dataset.
  - **Monetisation Plan Designer:** Enables Data Providers to put their datasets into the PISTIS market and specify their desired monetization method.
  - **Data Usage and Intentions Analytics:** Captures the underlying intentions of data owners and users regarding the datasets they own or are interested in, respectively.
- c) **Ledgers bundle:** Provides services for the storage of the data and the monetary transactions. It includes the following components of the PISTIS Market Exchange Platform:
  - **PISTIS Data Ledger:** Allows the storage of metadata of datasets transactions performed in the PISTIS Platform.
  - **PISTIS Monetary Ledger:** Allows the storage of monetary information of datasets transactions performed in the PISTIS Platform.
- d) **Transaction Services bundle:** Provides the services to create, execute and validate the different transactions with the use of smart contracts. It includes the following components of the PISTIS Market Exchange Platform:
  - **On/Off Platform Contract Inspector:** Performs various checks to ensure that all transactions adhere to the established rules of the active licenses of datasets.
  - **Transaction Auditor:** Enables the monitoring of the validity of the transactions in PISTIS Platform.

- **Smart Contract Template Composer:** Enables the selection of pre-defined contract templates and transforms them into the corresponding blockchain code to drive their execution.
- **Smart Contract Execution Engine:** Realizes the execution of the smart contracts for the corresponding datasets transactions.
- **Data Factory User Wallet:** Enables the secure management of the identity data and credentials of a Data Factory User.

The source code of the alpha releases of the different components of the PISTIS Market Exchange Platform resides on the project's GitHub repository and access can be provided upon request:

<a href="https://github.com/orgs/PISTIS-Platform/">https://github.com/orgs/PISTIS-Platform/</a>
---

The deliverable D3.2 has also obtained input from deliverables D1.2 and D3.1 and will provide feedback to deliverables D3.3 and D4.2 towards supporting the development and integration activities of WP3 and WP4.

## 1.1 DOCUMENT STRUCTURE

The present deliverable has been structured as follows:

- Section 1 presents an introduction to the work documented in this deliverable and presents the structure of this deliverable.
- Sections 2, 3, 4, 5 describe the design and development of the alpha releases of the different components of the PISTIS Market Exchange Platform, separated in the corresponding architectural bundles of services of the overall PISTIS Platform where each component belongs to.
- Section 7 provides the conclusions of the work documented in the deliverable.

## 2 DATA DISCOVERY BUNDLE

The Data Discovery bundle enables through its services the exploration and discovery of the available datasets that may interest a Data Consumer. It includes the Matchmaking Services component of the PISTIS Market Exchange Platform.

### 2.1 MATCHMAKING SERVICES

#### 2.1.1 Component Description

Matchmaking services are employed to link, as proactively as possible, data providers to data consumers, based on the latter's interest in data assets (e.g., application domain), as well as complementary data assets, which can result in an increase of data value. This is a user-item recommendation problem, where the items are datasets.

The main functionalities of the Matchmaking Service are:

1. Recommends data assets to Consumers based on their previous purchases and interactions with the Platform.
2. Connects Consumers with Owners of data assets, based on the platform interactions and interests of the former and the properties of the data assets of the latter.
3. Provides interpretability for the user-asset match.

The component will achieve these functionalities by implementing state-of-the-art recommendation algorithms: k-Nearest Neighbour (kNN), collaborative filtering (CF), deep neural networks (DNN) or graph neural networks (GNN). The minimum required data are user-assets interactions, where the concept of "interaction" can refer to "user views an asset", "user enquires about an asset", "user makes a query with certain parameters", "user purchases an asset". The choice of the algorithm will be heavily influenced by the amount of interaction data available for training, with an initial preference for kNN and probably CF, as DNN and GNN-based recommenders require large amounts to train.

#### 2.1.2 Technology Background

API development: Django REST framework

Programming language: Python 3.11

#### 2.1.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	Data Buyer	Find similar data assets to those I view.	Make better decisions about the data I acquire.	Alpha	Top-k datasets are recommended for each data asset.	Done	NA
US_02	Data Buyer	Have a personalized recommendation of data assets, based on my profile.	Better discover data assets which will aid in achieving my goals.	Beta	Top-k most relevant assets to users are recommended.	Upcoming	NA

### 2.1.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The MMS must retrieve data asset description metadata from the PISTIS Data Catalogue.	US_01, US_02	
FR_02	The MMS must compute a similarity score between a given asset and all other data assets in the PISTIS Data Catalogue.	US_01, US_02	
FR_03	The MMS must present those data assets with the highest similarity score, with respect to a given data asset.	US_01, US_02	
FR_04	The MMS must retrieve user interaction records.	US_02	What qualifies as a user-asset interaction will be defined post-Alpha.
FR_05	The MMS recommend data assets based on the user-asset interaction records, with respect to each Data Consumer	US_02	

### 2.1.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
<b>Performance efficiency</b>	NFR1	The MMS should compute recommendations in a timely manner, without causing any delays.
<b>Compatibility</b>	NFR2	The MMS will provide output that is compatible with the PISTIS Data Catalogue and its UI.
<b>Reliability</b>	NFR3	All endpoints of the MMS are functioning, and clear error messages are provided when a request fails.
<b>Portability</b>	NFR4	The MMS is containerized and can be deployed in hardware that supports Docker.

### 2.1.6 Component's Main Elements and Internal Architecture

The MMS consists of 2 main sub-components:

1. The Recommendation Engine implements the recommendation algorithms for the two main tasks: i) recommending data assets similar to a given data asset and ii) recommending data asset according to a user's profile.
2. The Similarity Metric Logic implements the similarity metrics that are needed for the functioning of the algorithms implemented by the Recommendation Engine.
3. The MMS exposes only reading endpoints, making available output consisting in recommended data assets, which are then used by the PISTIS Data Catalogue UI.

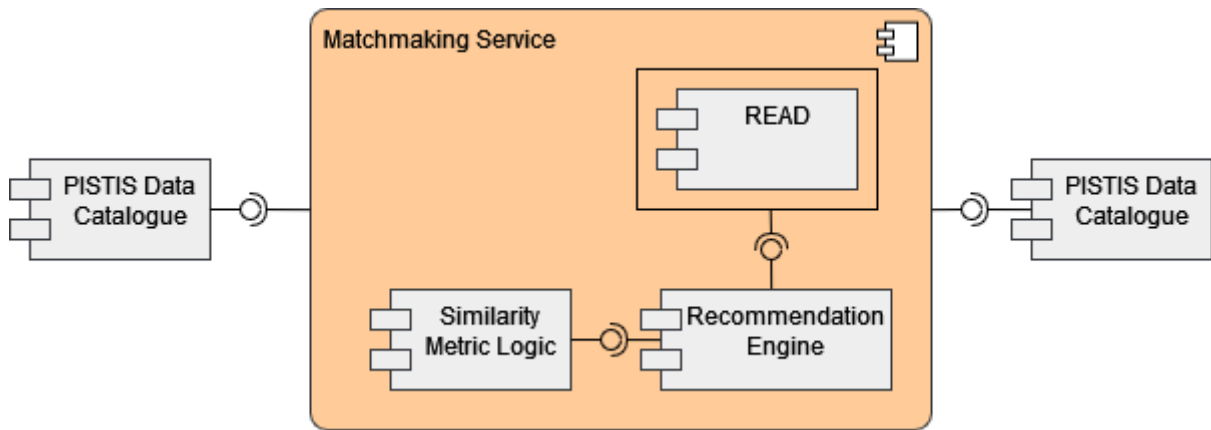


Figure 2-1: Matchmaking Service’s Internal Architecture

### 2.1.7 Mock-ups and Screenshots

The screenshot shows the PISTIS Data Catalogue user interface. At the top, there is a navigation bar with 'PISTIS' and several menu items: 'Home', 'My Data', 'My Transactions', 'Pistis Market', 'Manage Services', and 'Resources and activities monitor'. On the right, there is a user profile for 'Renee McKelvey' with a notification bell icon showing '2' alerts. Below the navigation bar, there are tabs for 'Datasets' and 'Catalogues'. The main content area is titled 'Dataset Berlin' and includes a 'Data Consumer View' badge. The dataset description contains placeholder text. To the right, a metadata box shows: 'Created: Feb 1, 2022', 'Updated: Feb 5, 2023', and 'Publisher: Berlin Pub'. Below this is a 'Distributions (2)' table:

Data	Format	Updated	Preview	Buy	Button
berlin-data-v0	json	Jul 12, 2022	Preview	Buy	Button
berlin-data-v1	csv	Feb 5, 2023	Preview	Buy	Button

Below the table, there is a section 'This data could interest you' with three recommended dataset cards: 'Dataset Weather Subhead', 'Dataset Mobility Subhead', and 'Dataset Energy Subhead'. Each card has a blue circular icon and a light blue background.

Figure 2-2: Matchmaking Services integrated with the PISTIS Data Catalogue UI



### 3 DATA MONETISATION BUNDLE

The Data Monetisation bundle enables through its services the Data provider to understand the value of their data within the market environment of the platform and to place the data on the PISTIS Data Catalogue using different methods to monetise by engaging in transactions performed with the aid of blockchain technology. It includes the following components of the PISTIS Market Exchange Platform:

- 1) Asset Description Bundler
- 2) FAIR Data Value Service
- 3) Digital DLT-FIAT Wallet
- 4) Data Investment Planner
- 5) PISTIS Market Insights
- 6) Monetisation Plan Designer
- 7) Data Usage and Intentions Analytics

#### 3.1 ASSET DESCRIPTION BUNDLER

##### 3.1.1 Component Description

The Asset Description Bundler (ADB) puts together a package which fully describes the data asset that is placed for publishing or trading. The foundation of the bundle is the original metadata, upon which the ADB gathers references to data valuations ran for the data asset (Data Owners may opt for this before publishing, whereas Data Consumers may opt for their own valuation before acquiring), as well as references related to the smart contract (access policies, monetisation option, smart contract ID, execution status).

The ADB is the intermediate layer facilitating the communication between other components and the catalogues.

The functionalities of the ADB are:

1. Provides a semantic model to encapsulate the information needed for data asset publishing and trading.
2. Collects and updates basic asset metadata.
3. Collects and updates references to data valuations ran for a data asset.
4. Collects and updates references to smart contracts built for a data asset: access policies, monetisation option, smart contract ID, execution status.

The metadata bundled or referenced by the ADB:

1. Basic metadata available upon Data Check-In:
  - a. provenance
  - b. license
  - c. lineage
  - d. format
  - e. accessibility
  - f. etc.
2. Data value dimensions:
  - a. valuation context (purpose, weights, business value)
  - b. data quality metrics
  - c. functional data utility (optional) metrics
  - d. bias assessment metrics

- e. anonymisation metrics OR deanonymisation risk
  - f. GDPR check
  - g. aggregate data value score
3. Information related to smart contracts:
- a. access policies
  - b. monetisation option
  - c. smart contract ID
  - d. contract execution status

The ADB will be built with the help of a semantic model suited for the description and exchange of the information previously presented. This will be achieved using a standard for key-value descriptions (JSON, YML) or an RDF-based solution.

### 3.1.2 Technology Background

Data vocabularies: Dublin Core (DC), W3C Data Catalogue Vocabulary (DCAT), W3C Provenance Data Models (PROV-DM), W3C Data quality Vocabulary (DQV), W3C Open Digital Rights Language (ODRL), Data Value Vocabulary (DaVe).

API development: Django REST framework

Programming language: Python 3.11

Database: SQLite 3 (only for Alpha). A permanent decision will be taken together with the developers of the other components, while preparing the Beta version.

### 3.1.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	Data Provider	Retrieve all metadata for an owned data set from the Factory Data Catalogue.	Publish the data asset on the PISTIS Data Catalogue.	Beta	The ADB retrieves a record containing the following type of metadata: basic, GDPR, anonymisation, data quality, transformations.	In progress	PISTIS.OUS.01, PISTIS.OUS.03, PISTIS.OUS.04
US_02	Data Provider	Retrieve all metadata for an owned data asset from the PISTIS Data Catalogue.	Exchange the data asset with another PISTIS User.	Beta	The ADB retrieves a record containing the following type of metadata: basic, GDPR, anonymisation, data quality, transformations.	In progress	PISTIS.OUS.01, PISTIS.OUS.03, PISTIS.OUS.04
US_03	Data Provider	Retrieve the reference to the data valuation relevant to the asset being bundled.	Publish the data asset on the PISTIS Data Catalogue.	Alpha	The ADB retrieves a correct record containing the reference to the data valuation	Done	PISTIS.OUS.07

					corresponding to a data asset.		
US_04	Data Provider	Define monetisation options and access policies, relevant for the current bundle. Different bundles with different monetisation options or access policies can be built for the same asset.	Publish or exchange the data asset.	Beta	The ADB retrieves correct records, each containing the reference to the monetisation plan and access policies of the data asset.	In progress	PISTIS.OUS.05, PISTIS.OUS.06, PISTIS.OUS.07

### 3.1.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The ADB must retrieve basic metadata for the data asset from the proper Data Catalogue.	US_01, US_02	The proper Data Catalogue is dependent on the action being taken. The Factory Data Catalogue for asset publication and the PISTIS Data Catalogue for asset exchange.
FR_02	The ADB must retrieve the data quality assessment for the data asset from the proper Data Catalogue.	US_01, US_02	
FR_03	The ADB must retrieve anonymization metadata from the proper Data Catalogue.	US_01, US_02	
FR_04	The ADB must retrieve GDPR compliance metadata for the data asset from the proper Data Catalogue.	US_01, US_02	
FR_05	The ADB must retrieve provenance and transformation metadata for the data asset from the proper Data Catalogue.	US_01, US_02	
FR_06	The ADB must retrieve the smart contract id from the Smart Contract Composer, associated with the asset bundle.	US_04	
FR_07	The ADB must retrieve the data value assessment for the data asset.	US_03	Only if the asset description bundle is for exchange.

### 3.1.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
Functional Suitability	NFR1	The ADB provides with a data model to collect all information for data asset publication or exchange.

<b>Performance efficiency</b>	NFR2	The ADB should fetch all necessary information from catalogues or other components without causing any delays.
<b>Compatibility</b>	NFR3	The ADB will easily integrate with all directly connected catalogues and components.
<b>Usability</b>	NFR4	The ADB will rely on a UI to launch the processes related to components that need to provide it with the required information (Data Valuation Services, Monetisation Plan, Access Policies Editor).
<b>Reliability</b>	NFR5	All endpoints of the ADB are functioning and clear error messages are provided when a request fails.
<b>Security</b>	NFR6	The ADB can be created only by a registered Data Provider. However, it can be updated by a registered Data Consumer wishing to acquire a published data asset.
<b>Portability</b>	NFR7	The ADB is containerized and can be deployed in hardware that supports Docker.

### 3.1.6 Component's Main Elements and Internal Architecture

The following image depicts the internal architecture of the Asset Description Bundler. The ADB receives metadata from the appropriate catalogue, to which it adds information from several other components. The ADB allows for CRUD operations.

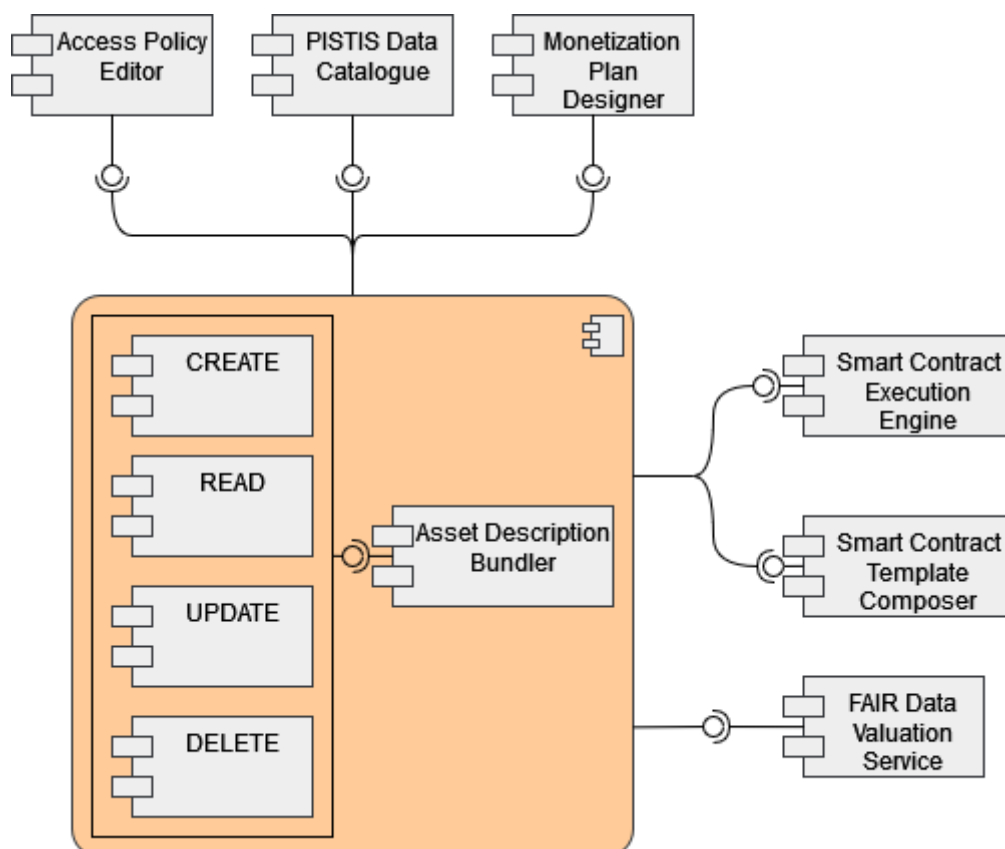


Figure 3-1: Asset Description Bundler's Internal Architecture

### 3.1.7 Mock-ups and Screenshots

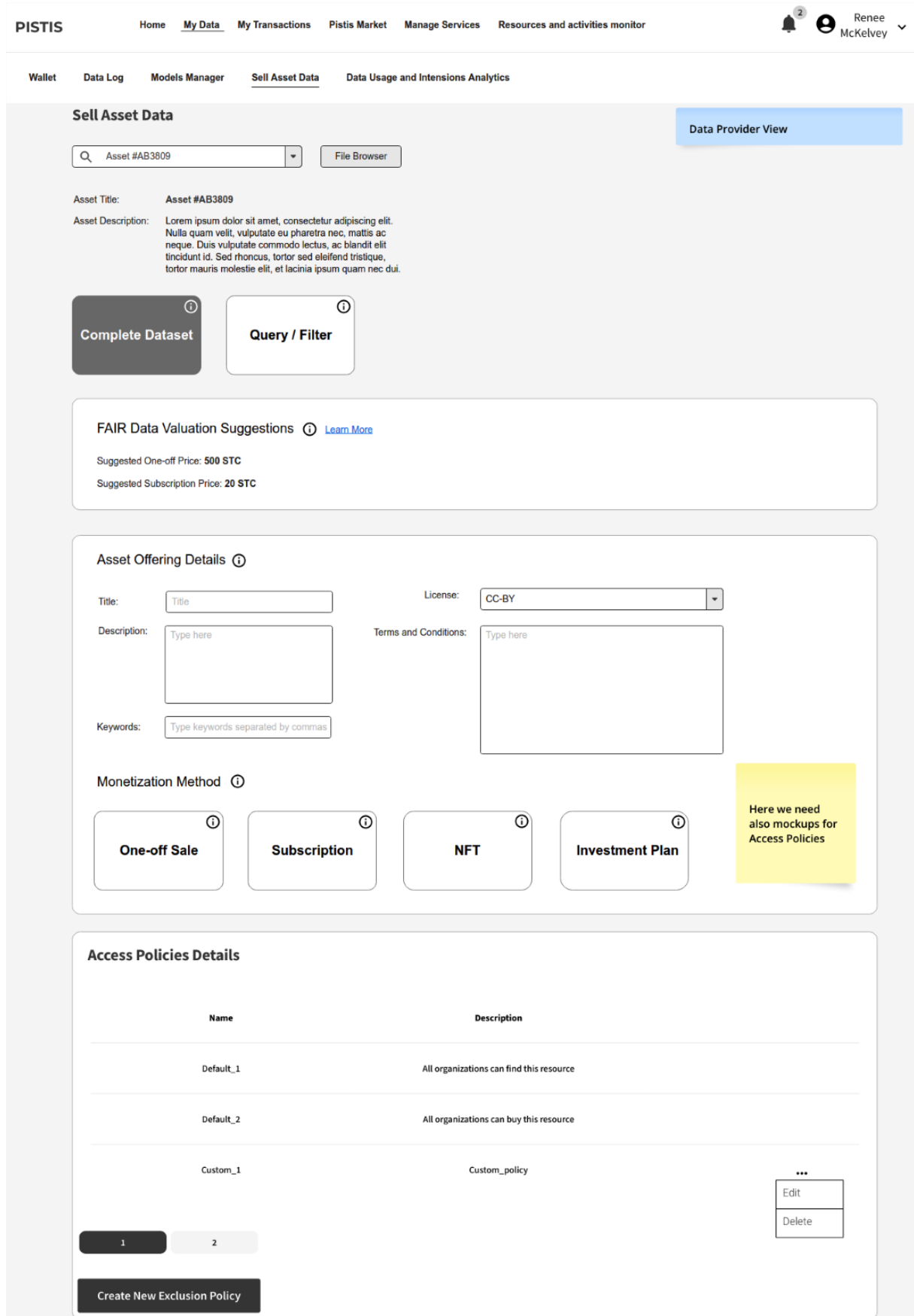


Figure 3-2: Asset Description Bundler UI Mockup, illustrating the necessity of the ADB to collect information from the Data Valuation Services, Monetisation Plan, Access Policies and configure the Smart Contract

## 3.2 FAIR DATA VALUATION SERVICE

### 3.2.1 Component Description

The FAIR Data Valuation Service (DVS) comprises of a set of methods that support the task of data valuation – assigning a quantitative value to a data asset. It is a complex, context-dependent, and multi-dimensional process, built upon several other processes: data and metadata quality assessment, functional data utility assessment, feature bias assessment, legal and ethical assessment, privacy analysis. While the service will mainly support a dimensional approach to data valuation, our ambition is for the component to integrate and/or support market-based and economic models.

The main functionalities of the FAIR Data Valuation Service are:

1. Allow for the User to define the data valuation context: purpose, relevant dimensions, business value/goals, IP requirements. This will be achieved by designing a dedicated UI.
2. Retrieves metadata relevant to the data value dimensions (DVDs) of a selected data asset: basic metadata (provenance, access policy), usage and intentions, data quality metrics, anonymisation metrics, bias metrics, GDPR compliance, market insights. If these are not available, it will query other components' APIs trying to retrieve them (Usage & Intentions Analytics, Data Quality Assessment, Repo for Pre-trained AI Models, Market Insights, Anonymisation, GDPR Checker, Lineage Tracker).
3. Functional utility checks to assess the contextual value of each data point in a structured dataset – currently available for classification or regression problems.
4. Implements a Data Valuation Methodology to aggregate the quantification of each DVD into an aggregate Data Value Score.
5. Reports an aggregate Data Value Score, together with a breakdown along the DVDs.
6. Communicates the results of the data valuation process to instances of the Asset Description Bundler (ADB).

### 3.2.2 Technology Background

Data models: Dublin Core (DC), W3C Data Catalogue Vocabulary (DCAT), W3C Provenance Data Models (PROV-DM), W3C Data quality Vocabulary (DQV), W3C Open Digital Rights Language (ODRL), Data Value Vocabulary (DaVe).

API development: Django REST framework

Programming language: Python 3.11

Database: SQLite 3 (only for Alpha). A permanent decision will be taken together with the developers of the other components, while preparing the Beta version.

### 3.2.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>	so that <Reason>				
US_01	Data Provider	Retrieve the scores of each data value dimensions.	Compute their aggregate score (average).	Beta	Receive a record containing the score for each data value dimension.	In progress	PISTIS.OUS.01, PISTIS.OUS.03, PISTIS.OUS.04, PISTIS.OUS.07

US_02	Data Buyer	Configure a set of weights for each data value dimension.	Use them to weight the aggregate data value score.	Beta	Send to the ADB of interest a set of weights, reflecting the contextual interest into each data value dimension.	In progress	PISTIS.OUS.07
US_03	Data Buyer	Retrieve the scores of each data value dimensions.	Compute aggregate data value score as a weighted average of each data value dimension score.	Alpha	Retrieve the aggregate score according to an agreed aggregation formula and update the corresponding ADB accordingly.	Done	PISTIS.OUS.01, PISTIS.OUS.03, PISTIS.OUS.04, PISTIS.OUS.07
US_04	Data Buyer	View the aggregate data value and the dimensional scores.	Understand the strong and weak points of the dataset I want to buy.	V.1.0.0	UI with aggregate DV score and dimensional breakdown.	Upcoming	PISTIS.OUS.07

### 3.2.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The DVS must allow a Data Buyer to configure importance weights (one-per-data value dimension). This will be done through a form-like UI.	US_02	Only if the data asset has already been published.
FR_02	The DVS must retrieve basic metadata from either the Factory Catalogue or the PISTIS Catalogue.	US_01, US_03	Depending on whether the asset was published or not.
FR_03	The DVS must retrieve the data quality assessment of the data asset, from the appropriate Catalogue.	US_01, US_03	
FR_04	The DVS must call the data utility assessment sub-module, for the data asset.	US_01, US_03	
FR_05	The DVS must retrieve the results of the GDPR Checker, from the appropriate catalogue.	US_01, US_03	
FR_06	The DVS must retrieve the results of the privacy and anonymisation tool, from the appropriate catalogue.	US_01, US_03	
FR_07	The DVS must call the data bias assessment sub-module for the data asset.	US_01, US_03	
FR_08	The DVS must compute a data valuation score, considering the weights and the dimensional scores.	US_01, US_03	
FR_09	The DVS must present the data valuation scores (aggregate and dimensional), together with their explanations in a dedicated UI.	US_04	

### 3.2.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
<b>Functional Suitability</b>	NFR1	The DVS will run the data valuation process only after all necessary information is collected.
<b>Performance efficiency</b>	NFR2	The DVS should fetch the necessary information and run the data valuation process in a timely manner. This can be challenging, especially because of the lengthy data utility assessment.
<b>Compatibility</b>	NFR3	The DVS will easily integrate with the ADB and the data catalogues for I/O operations.
<b>Usability</b>	NFR4	The DVS will provide a UI that will allow users to easily understand the data value score and its dimensional breakdowns, thus understanding the main drivers of the value of their data.
<b>Reliability</b>	NFR5	All endpoints of the DVS are functioning, and clear error messages are provided when a request fails.
<b>Security</b>	NFR6	The DVS will be secured with the Identity and Authorization Manager and Access Policies Manager.
<b>Portability</b>	NFR7	The DVS is containerized and can be deployed in hardware that supports Docker.

### 3.2.6 Component’s Main Elements and Internal Architecture

The FAIR Data Value Service’s internal architecture is presented below. It illustrates its connection to components that contribute to Data Valuation, as well as its sub-components: the Bias Assessment and the Data Utility Assessment. It supports CRUD operations.

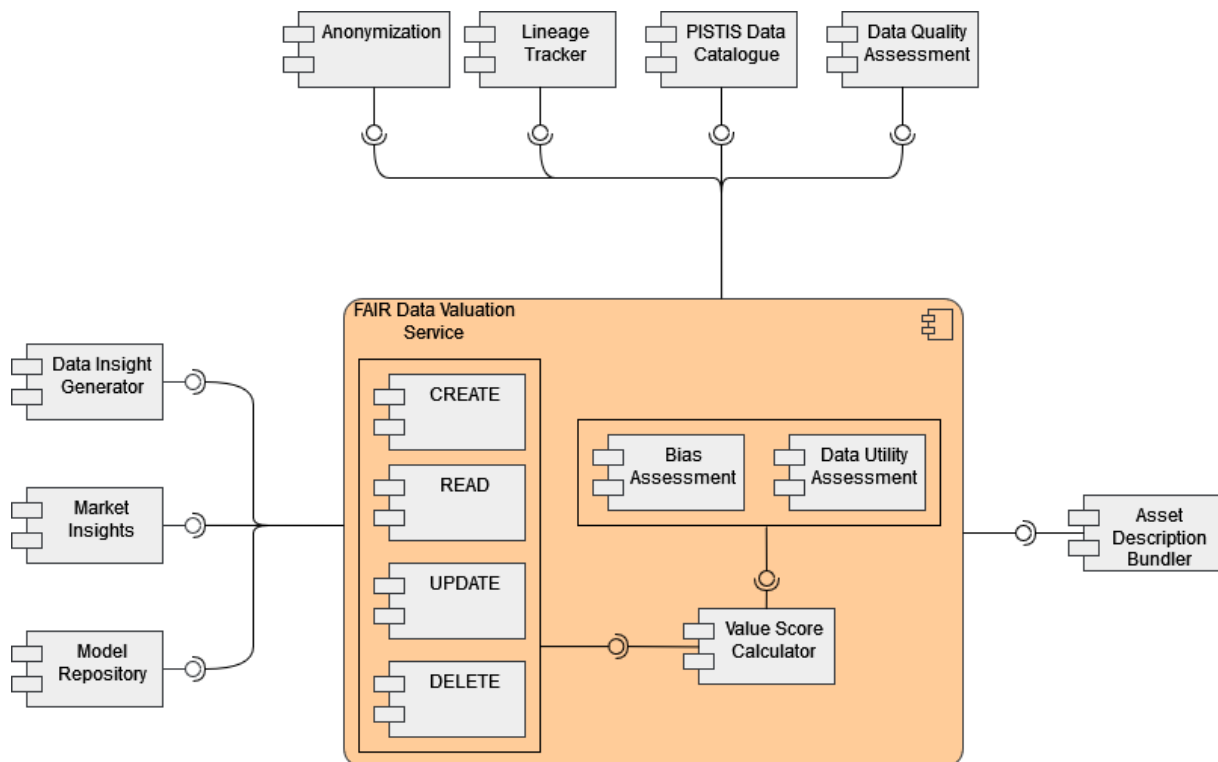


Figure 3-3: FAIR Data Value Service’s Internal Architecture



### 3.2.7 Mock-ups and Screenshots

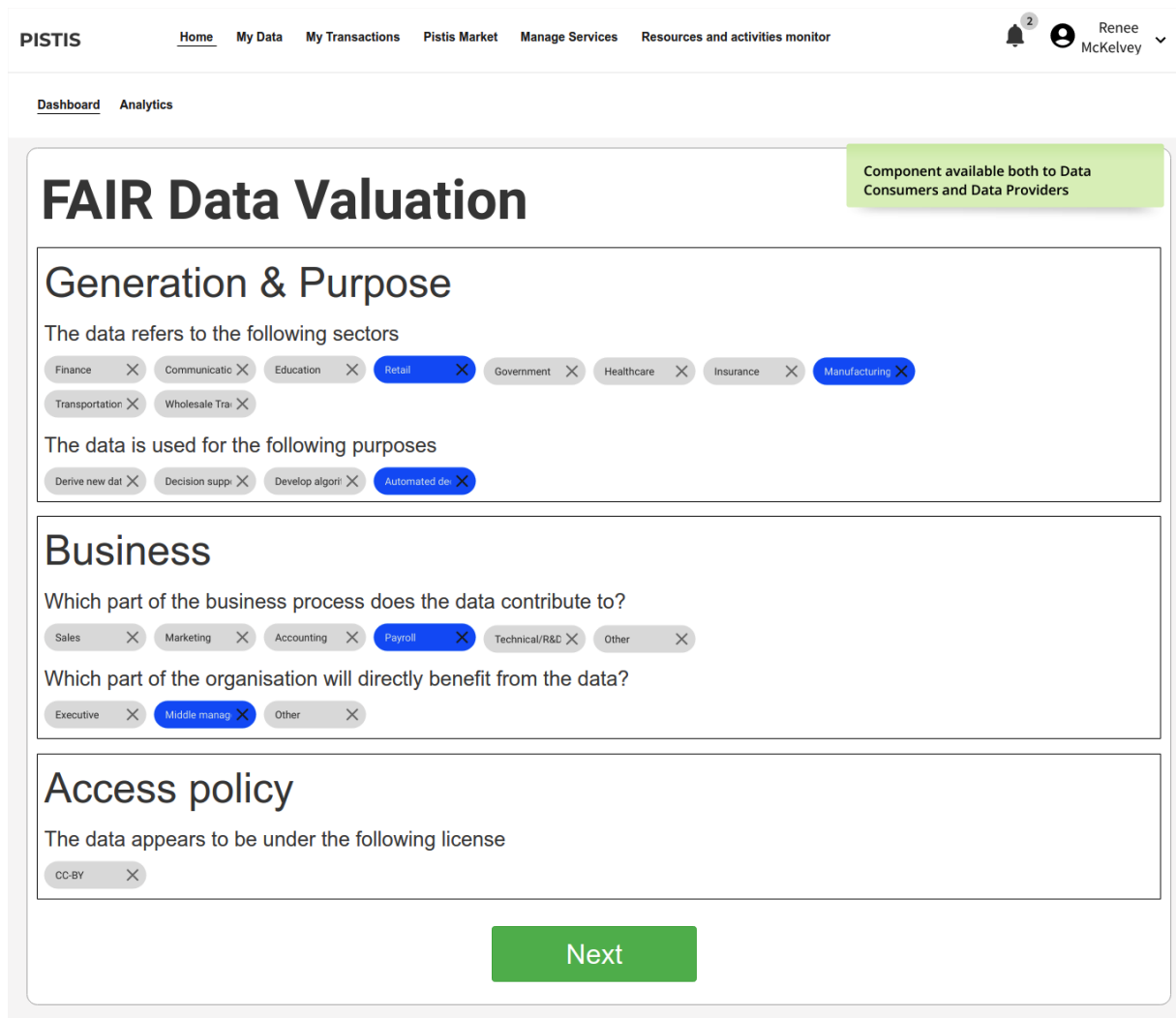


Figure 3-4: FAIR Data Valuation Service UI. Definition of Purpose, Business goals and Licenses.

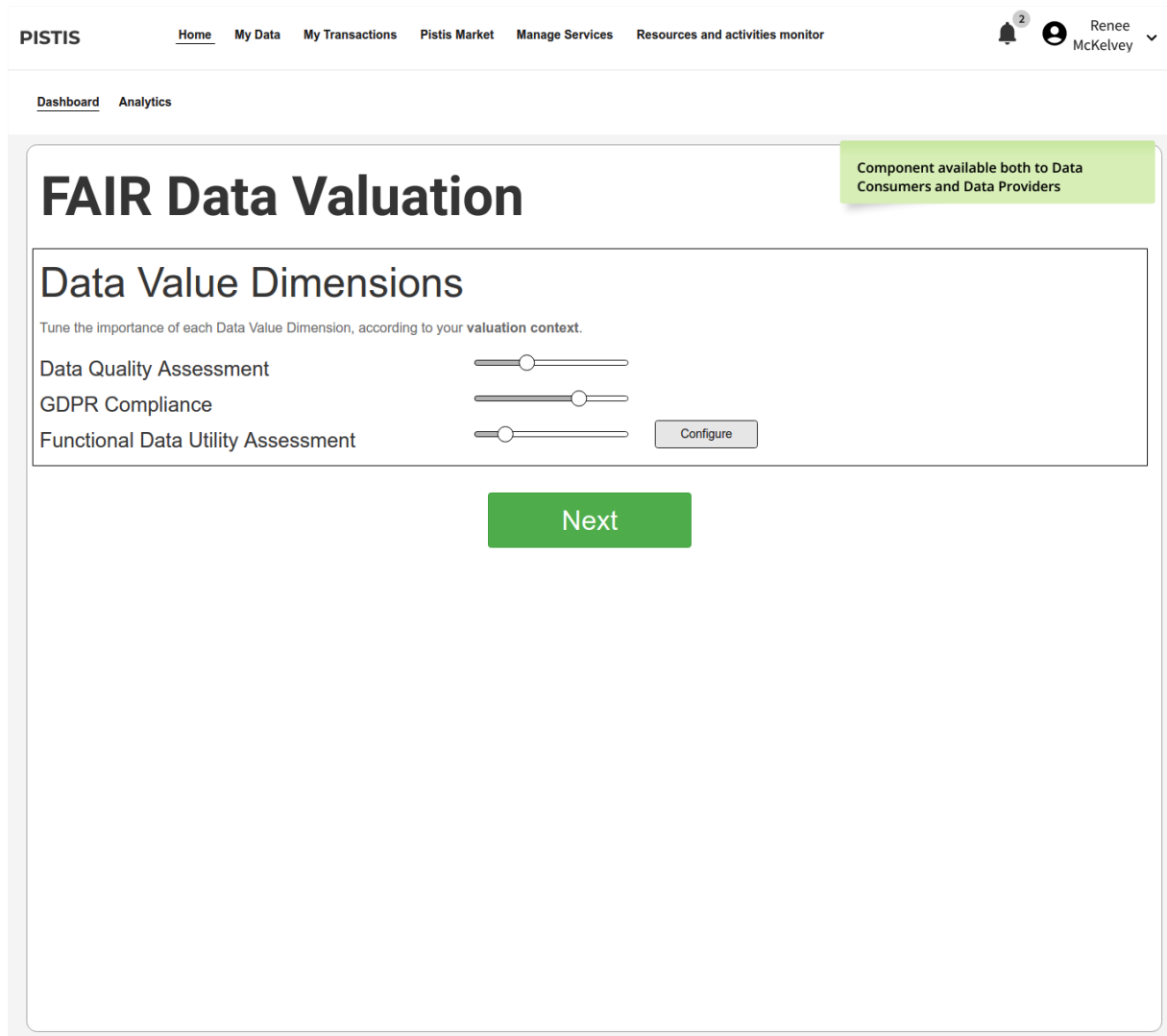


Figure 3-5: FAIR Data Valuation Service UI. Definition of weights for the data value dimensions.

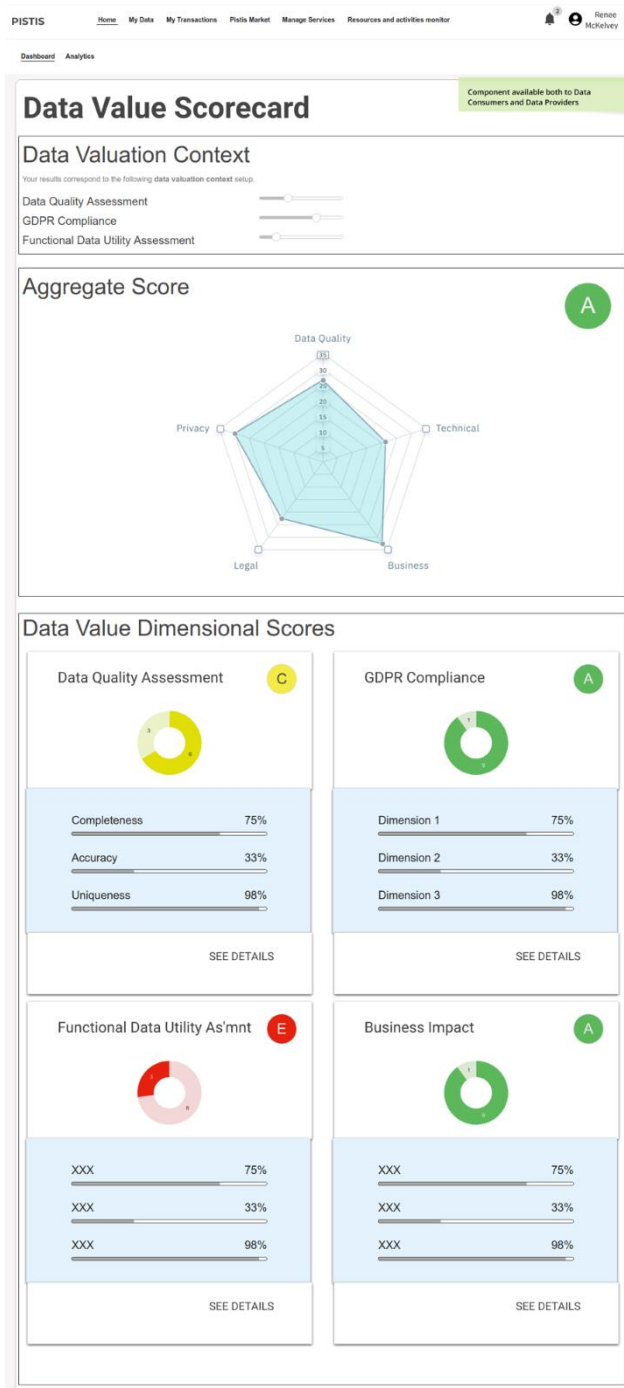


Figure 3-6: FAIR Data Valuation Service UI. Data Value Scorecard.

### 3.3 DIGITAL DLT-FIAT WALLET

#### 3.3.1 Component Description

This component is responsible for overseeing the platform’s value transactions in a DLT private network, guaranteeing quick and reliable exchanges. It also serves as a crucial interface with traditional FIAT banking accounts, enabling seamless fund transfers and effortless conversion between fiat and digital tokens. DLT transactions are processed within a private network that maintains a stable 1:1 exchange rate between FIAT money and PISTIS tokens. To uphold the fixed exchange rate, the component is tasked with providing a collateralized stable DLT coin,

supported by the underlying DLT implementation and regulated by a liquidity mechanism that will protect the exchange rate from the fluctuations in the currency's value.

### 3.3.2 Technology Background

**Generic:** Wallet application is a web based portable binary application that offers an advanced RESTful API capable to provide support for both the FIAT and the DLT ecosystem. Although a standalone component, internally it is separated into a DLT agnostic intermediate layer on top of a DLT, FIAT aware one.

**DLT Agnostic Layer:** The Digital DLT-FIAT Wallet component will not provide a graphical user interface itself. It is a Rust web-based application that will offer the RESTful API with all the needed functionalities such as authentication and authorization, combined with a minimal Postgres 16 database to store various important information like monetary transactions history, account details and log events. Access by unauthorized entities to the provided functionalities will be limited through the central Keycloak instance, which offers security mechanisms along with user authentication and management. It will also be responsible for implementing the PSD2 Open Banking module, acting as an intermediate protective layer. The former, for security reasons, will not have any public endpoints exposed to other PISTIS modules.

**DLT Aware Layer:** A Rust based application responsible for DLT integration, serving as a gateway for interactions with the PISTIS DLT Ledger. It will also be responsible, acting as an intermediate protective layer, for implementing the PSD2 Open Banking module. Although the app implements a DLT agnostic interface, it will be based on the IOTA DLT protocol. Interaction with the underlying DLT nodes is achieved through the utilization of the IOTA SDK Rust module. DLT-FIAT Wallet and all its accompanying services, like firewall and database/DLT management routines, will be offered as a standalone virtual machine (VM) image. Additionally, a set of dockerised images to be easily deployed in docker based infrastructure will be provided.

**DLT Integration:** The PISTIS wallet solution will be based on the latest IOTA Stardust version, that supports **Smart Contracts** and asset **Tokenization**. In the first development stage, the public IOTA Testnet will be used. The final solution will provide a dedicated PISTIS private permissioned Monetary Ledger, offering both L1 (native token) and L2 (smart contracts) network solutions and a dedicated permanode, a server that will store the entire history of transactions in the system.

**L1 Node Management:** Hornet is an IOTA DLT node software written in Go. It provides the full node capabilities, including full support of the latest network updates.

**L2 Smart Contracts:** Smart Contracts are accessible in a L2 network, based on IOTA's WASP plugin that is deployed alongside the main Hornet node. ISC (IOTA Smart Contracts) support contracts written both in EVM/Solidity and Rust/Wasm (WebAssembly).

#### **System Hosting IOTA nodes:**

- Development Phase: Qemu based VMs on ICCS infrastructure
- Final Deliverable to WP4: Alpine Linux based PODS

### 3.3.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	Data Consumer	Create/manage a FIAT and DLT wallet	I can convert FIAT money to PISTIS tokens and purchase/invest on/subscribe to a dataset.	Alpha	Access wallets' empty balance	Done	PISTIS.OUS.02 PISTIS.OUS.10
US_02	Data Owner	Create/manage a FIAT and DLT wallet	I can convert received PISTIS tokens, from a dataset ownership transfer/investment/subscription to FIAT money	Alpha	Access wallets' empty balance	Done	PISTIS.OUS.02 PISTIS.OUS.10
US_03	PISTIS Administrator	Create/manage a FIAT and DLT wallet	I can receive PISTIS commission tokens from a dataset transaction and convert them to FIAT money	Alpha	Access wallets' empty balance	Done	PISTIS.OUS.02 PISTIS.OUS.10
US_04	Data Consumer	Transfer PISTIS tokens to Data Owner and commission fees to PISTIS Admin	I can purchase/invest on/subscribe to a dataset	Alpha	Transaction is visible in DLT wallet's recent activity	Done	PISTIS.OUS.10
US_05	Data Owner	Receive PISTIS tokens	I can sell an owned dataset	Alpha	Transaction is visible in DLT wallet's recent activity	Done	PISTIS.OUS.10
US_06	PISTIS Administrator	Receive PISTIS commission fees	I can validate a dataset purchase/investment/subscription	Alpha	Transaction is visible in DLT wallet's recent activity	Done	PISTIS.OUS.10
US_07	Data Owner	Create NFTs	I can prove the ownership of a dataset	Beta	Newly created NFT is visible in DLT wallet's digital assets	Upcoming	PISTIS.OUS.06 PISTIS.OUS.10
US_08	Data Owner	Transfer NFT's	I can sell a dataset for which i hold 100 percent of equity	Beta	Transaction is visible in DLT wallet's recent activity	Upcoming	PISTIS.OUS.06 PISTIS.OUS.10

US_09	Data Consumer	Receive NFTs	I can buy a dataset and prove that I am the sole owner.	Beta	Received NFT is visible in DLT wallet's digital assets	Upcoming	PISTIS.OUS.10
US_10	Data Owner	View my wallet's digital assets (NFTs), balance and recent activity	I can monitor my account and plan possible future actions	Alpha	Display FIAT and DLT wallet's balance	Done	PISTIS.OUS.11
US_11	Data Consumer	View my wallet's digital assets (NFTs), balance and recent activity	I can monitor my account and plan possible future actions	Alpha	Display FIAT and DLT wallet's balance	Done	PISTIS.OUS.11
US_12	PISTIS Administrator	View my wallet's digital assets, balance and recent activity	I can monitor my account and plan possible future actions	Alpha	Display FIAT and DLT wallet's balance	Done	PISTIS.OUS.11
US_13	PISTIS Administrator	View ledger's history-activity	I can facilitate services like prices forecasting, market insights or dataset quality/pricing	Beta	Display specific transaction information upon request	Upcoming	PISTIS.OUS.11
US_14	Data Consumer	Receive notifications about the outcome of a transaction	I can track my wallet's balance in real-time	Alpha	Display notification message	Done	PISTIS.OUS.10 PISTIS.OUS.11
US_15	Data Owner	Receive notifications about the outcome of a transaction	I can track my wallet's balance in real-time	Alpha	Display notification message	Done	PISTIS.OUS.10 PISTIS.OUS.11
US_16	PISTIS Administrator	Receive notifications about the outcome of a transaction	I can track my wallet's balance in real-time	Alpha	Display notification message	Done	PISTIS.OUS.10 PISTIS.OUS.11
US_17	PISTIS Platform	Reverse a transaction	PISTIS tokens/NFTs can be returned to owner in case of failure or on platform's demand	Beta	New transaction is visible in DLT wallet's recent activity	Upcoming	PISTIS.OUS.06
US_18	Data Owner	Set smart contract terms	I can set the data purchase/investment/subscription details	v.1.00	New smart contract terms are available in smart contract view entry	Upcoming	PISTIS.OUS.10 PISTIS.OUS.06

					point call after smart contracts state modification		
US_19	PISTIS Administrator	Deploy a smart contract	The data purchase/investment/subscription can be carried out based on the Data Owner's terms	Beta	Contract deployment transaction is visible in DLT wallet's recent activity	Upcoming	PISTIS.OUS.06
US_20	PISTIS Consumer	View a smart contract's terms	I can decide whether to accept them and proceed with the dataset purchase/investment/subscription/ or not	V1.00	Smart contract terms are visible in smart contract view entry point call	Upcoming	PISTIS.OUS.10
US_21	Data Owner	Sign a smart contract	I can authorize the smart contract's execution	Beta	Stakeholder smart contract signatures are visible in smart contract view entry point call	Upcoming	PISTIS.OUS.10
US_22	Data Consumer	Sign a smart contract	I can authorize the smart contract's execution	Beta	Stakeholder smart contract signatures are visible in smart contract view entry point call	Upcoming	PISTIS.OUS.10
US_23	PISTIS Administrator	Sign a smart contract	I can authorize the smart contract's execution	Beta	Stakeholder smart contract signatures are visible in smart contract view entry point call	Upcoming	PISTIS.OUS.10
US_24	PISTIS Platform	Execute a smart contract	The transaction amount may be transferred, or the new ownership status may be recorded	Beta	Be notified of executed smart contract function outcome	Upcoming	PISTIS.OUS.10

US_25	Data Owner	Receive notification for the smart contract execution	I can monitor the value transaction and track my wallet's balance in real-time	Beta	Display notification message	Upcoming	PISTIS.OUS.11
US_26	Data Consumer	Receive notification for the smart contract execution	I can monitor the value transaction	Beta	Display notification message	Upcoming	PISTIS.OUS.12
US_27	PISTIS Administrator	Check the components health status	I can monitor the Platform's status	Alpha	Display component status	Done	PISTIS.SOUS.01
US_28	PISTIS Owner	Receive notification for a digital asset (NFT) creation	I can sell a newly created dataset with the NFT Monetization method	Alpha	Display notification message and transaction is visible in DLT wallet's recent activity	Done	PISTIS.OUS.11
US_29	Data Consumer	Receive notification for a digital asset (NFT) transfer	I can track my digital assets in real-time	Alpha	Display notification message and transaction is visible in DLT wallet's recent activity	Done	PISTIS.OUS.11
US_30	PISTIS Owner	Manage multiple addresses per dlt wallet	I can manage my digital assets (NFTs) or tokens and make my activity harder to track	Beta	New address is visible in wallet information and can receive tokens-assets	Upcoming	PISTIS.OUS.10 PISTIS.OUS.02
US_31	Data Provider	Manage multiple addresses per dlt wallet	I can manage my digital assets (NFTs) or tokens and make my activity harder to track	Beta	New address is visible in wallet information and can receive tokens-assets	Upcoming	PISTIS.OUS.10 PISTIS.OUS.02
US_32	Data Consumer	Manage multiple addresses per dlt wallet	I can manage my digital assets (NFTs) or tokens and make my activity harder to track	Beta	New address is visible in wallet information and can	Upcoming	PISTIS.OUS.10 PISTIS.OUS.02



					receive tokens-assets		
--	--	--	--	--	-----------------------	--	--

### 3.3.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The component creates a wallet for managing FIAT currency.	US_01, US_02, US_03	
FR_02	The component creates a wallet for managing the DLT cryptocurrency.	US_01, US_02, US_03	
FR_03	The component allows the destruction of the FIAT wallet.	US_01, US_02, US_03	
FR_04	The component allows the destruction of the DLT wallet.	US_01, US_02, US_03	
FR_05	The component implements a wallet backup and restore mechanism.	US_01, US_02, US_03	In case of failure uses a previous wallet backup file to restore the wallet's state.
FR_06	The component tracks the current FIAT wallet's balance.	US_10, US_11, US_12	
FR_07	The components tracks the current DLT wallet's balance.	US_10, US_11, US_12	Refers to the total wallet's balance and digital assets.
FR_08	The components tracks the DLT wallet's digital assets (NFTs).	US_10, US_11, US_12	Refers to NFT's or smart contract ownership.
FR_09	The component provides the history of past DLT wallet's activity.	US_13	Transactions are divided into incoming and outgoing.
FR_10	The component provides the history of past FIAT wallet's activity.	US_13	Transactions are divided into incoming and outgoing.
FR_11	The component provides the history of all previous ledger activity.	US_13	This functionality is available only to the PISTIS Owner
FR_12	The component allows the creation of multiple receive and transfer addresses.	US_30, US_31, US_32	An initial DLT receive address will be created at wallet creation. New receive and transfer addresses can be created later.
FR_13	The component allows the transfer of PISTIS tokens or digital assets (NFTs) between addresses within the same wallet.	US_01, US_02, US_03, US_30, US_31, US_32, US_04, US_05, US_06, US_07, US_08	Refers the L1 transactions to support the usage of multiple wallet addresses.
FR_14	The component allows the deletion of wallet's addresses.	US_01, US_02, US_03, US_30, US_31, US_32	
FR_15	The component provides a listing of all current active wallet's addresses.	US_01, US_02, US_03	

FR_16	The component provides information about the state of a specific wallet address.	US_01, US_02, US_03, US_10, US_11, US_12, US_30, US_31, US_32	Refers to balance and digital assets of a specific wallet's address.
FR_17	The component allows the conversion of FIAT currency to PISTIS tokens.	US_01	An initial version of Liquidity Mechanism Component will be provided in beta version. A Fully working version will be provided in the final version.
FR_18	The component allows the conversion of PISTIS tokens to FIAT currency.	US_02	Same as above
FR_19	The component allows the transfer of PISTIS tokens from seller's to buyer's wallet address.	US_04	In the alpha version value transactions will be executed in the DLT Layer 1 (Transferred directly to the stakeholder's wallets)
FR_20	The component allows the transfer of NFT's and digital assets from seller's to buyer's wallet address.	US_08	In the alpha version digital asset transactions will be executed in the DLT Layer 1 (Transferred directly to the stakeholder's wallets)
FR_21	The component is responsible for deploying a new smart contract.	US_19	Layer 2 smart contract functionality will be added in the beta version.
FR_22	The component enables the review of the terms of a deployed smart contract.	US_20	This functionality will be introduced in later development stages.
FR_23	The component enables the modification of the terms of a deployed smart contract.	US_18	This functionality will be introduced in later development stages.
FR_24	The component allows a wallet to sign a smart contract.	US_21	Layer 2 smart contract functionality will be added in the beta version.
FR_25	The component creates a notification about the outcome of a transaction.	US_14, US_15, US_16	Refers to L1 functionality in alpha version.
FR_26	The component creates a notification about the outcome of a digital asset (NFT) transfer.	US_28, US_29	Refers to L1 functionality in alpha version.
FR_27	The component creates a notification about the execution of a smart contract function.	US_25, US_26	Layer 2 smart contract functionality will be added in the beta version.
FR_29	The components provides information about current health status	US_27	Status information mainly refer to the availability of the IOTA DLT Hornet and Wasp nodes.

### 3.3.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement category	Sub-Id	Description (Detailed description of the requirement)
<b>Performance efficiency</b>	NFR1	The overall transaction should be performed without significant delays
<b>Reliability</b>	NFR2	IOTA DLT wallet should synchronize before each action.
<b>Reliability</b>	NFR3	The component should wait for transaction finalization/inclusion in the DLT before notifying the user for the transaction/smart contract execution status
<b>Functional Suitability</b>	NFR4	FIAT wallet deletions should be allowed only on empty wallets
<b>Functional Suitability</b>	NFR5	DLT address and wallet deletion should be allowed only on empty addresses or wallets
<b>Security</b>	NFR6	Demand wallet seed or passphrase to initiate a transaction
<b>Security</b>	NFR7	IOTA Stronghold will be used for secure secret management
<b>Portability</b>	NFR8	The component will be isolated using Docker in order to run in architecture-agnostic infrastructures
<b>Reliability, Usability</b>	NFR9	User's wallet backups will be created automatically at regular intervals, in addition to user-initiated requests, ensuring a seamless and proactive backup process.
<b>Maintainability, Availability</b>	NFR10	Application supports dynamic on-runtime configuration changes.
<b>Security</b>	NFT11	The application restricts access for unauthorized users.
<b>Recoverability</b>	NFT12	The component provides a restore mechanism for synchronizing IOTA DLT nodes to the current ledger status after critical failures.

### 3.3.6 Component's Main Elements and Internal Architecture

The internal architecture of the component comprises the following primary subcomponents:

- The wallet backend serves as the primary gateway to access the functionalities of the component. It carries the responsibility of interacting with the Distributed Ledger Technology (DLT) and handling actions related to managing both the DLT and FIAT Wallet.
- The FIAT wallet component is responsible for FIAT PSD2 Open Banking module integration.
- The database is utilized to store crucial information and log specific events for record-keeping and reference purposes.
- The IOTA Hornet node allows access to the core functions of the IOTA DLT.
- The IOTA Wasp node is a plugin to the main Hornet node. It is responsible for handling all L2 smart contract functionalities.

A UML component diagram of the subcomponents is presented below.

## Digital DLT - FIAT Wallet

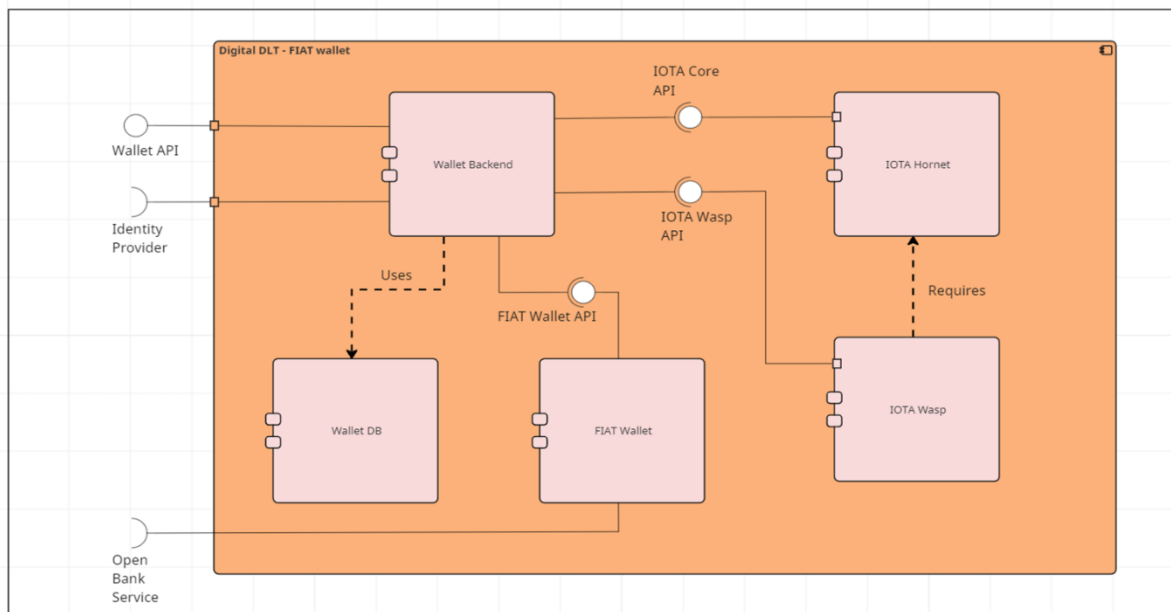


Figure 3-7: Digital DLT-FIAT Wallet Internal Architecture

## 3.3.7 Mock-ups and Screenshots

This component does not provide a corresponding UI Front End application.

## 3.4 DATA INVESTMENT PLANNER

## 3.4.1 Component Description

The Data Investment Planner represents an innovative approach poised to transform the way the financing of data set management and trading is approached. By enabling participants in the supply chain to create an entirely different financing framework by sharing capital, data owners and providers are allowed to raise investment by offering a share of the potential profits generated from their data.

The Data Investment Planner is responsible for designing an investment plan related to a specific dataset of a data provider. Once a dataset is registered in PISTIS, the user (data provider) enters the Monetisation Plan Designer and inserts information about the minimum and total equity percentage, the equity price and the maximum number of investors that can purchase an equity percentage.

The information is sent to the Data Investment Planner which, after analyzing this information, generates a suggested investment plan that can be saved for further editing later and upon finalization of selection, it can be submitted towards enabling the data provider to trade his/her dataset and obtain monetary gains.

Alternatively, the data provider has the option to select an investment type from a list of ready-made investment templates and edit the details of this template. The data provider can also in this case select to save and/or submit his/her data investment plan.

## 3.4.2 Technology Background

The Data Investment Planner will offer a frontend service with a UI, developed in Nuxt and Vue.js, for displaying to the users the investment plan details or selecting ready-made

templates and allowing them to make any necessary edits. For the backend services of this component, Nest.js will be exploited, while intra-component communication will be facilitated with Rest APIs designed with Swagger.

### 3.4.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>	so that <Reason>				
US_01	PISTIS User	Be directed to the Data Investment Planner when selecting this method of monetization	I may continue setting the desired parameters	Beta	The settings for the Data Investment Plan are displayed under the selection of the Data Investment Planner	Upcoming	PISTIS.OUS.07
US_02	PISTIS User	Create a new investment plan for my dataset by specifying the equity percentage, the price for it and the minimum amount of equity one can get	I can use this plan	Beta	Create an investment plan by setting these details	Upcoming	PISTIS.OUS.07
US_03	PISTIS User	I want to limit the number of "investors" that can buy an equity percentage	I can control how many people can co-own a dataset	Beta	Create an investment plan by setting the number of investors	Upcoming	PISTIS.OUS.07
US_04	PISTIS User	I want to save the investment plan I generated	I can reuse it later	Beta	The investment plan is saved	Upcoming	PISTIS.OUS.07
US_05	PISTIS User	I want to select an investment type from a list of ready-made templates including explanations	I may monetise my dataset	Beta	There is a list of ready-made investment type templates to select from	Upcoming	PISTIS.OUS.07
US_06	PISTIS User	I want to edit the details of a ready-made investment type template	I can create a new custom investment plan	Beta	The user can edit these details	Upcoming	PISTIS.OUS.07

### 3.4.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The Data Investment Planner should allow users to view the settings of the data investment plan, when selecting this monetization method	US_01	
FR_02	The Data Investment Planner should enable users to specify the equity percentage, the price for it and the minimum amount of equity that a buyer of the dataset can get	US_02	
FR_03	The Data Investment Planner should enable users to define the maximum number of investors that can purchase an equity percentage	US_03	
FR_04	The Data Investment Planner should allow users to save their data investment plan selections to revisit them later	US_04	
FR_05	The Data Investment Planner should provide to the users a list of ready-made templates including explanations to select from	US_05	
FR_06	The Data Investment Planner should enable users to edit the details of a selected ready-made investment type template	US_06	

### 3.4.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
<b>Performance efficiency</b>	NFR1	The overall process shall be performed without delays and should not consume unnecessary system resources
<b>Reliability</b>	NFR2	The Data Investment Planner shall operate in a reliable manner, delivering valid investment plans
<b>Security</b>	NFR3	The overall process shall be made through secure communication channels

### 3.4.6 Component's Main Elements and Internal Architecture

The Data Investment Planner comprises the following components:

- **Frontend Service:** A user interface that enables the users to fill-in and edit the details of a data investment plan for a dataset.
- **Backend Service:** This component receives input from the Monetization Plan Designer for the creation of a data investment plan, and can retrieve specific investment type templates from the list of already stored templates.
- **Ready-made Investment Type templates:** This component provides the ready-made templates to the backend to be used by the users when they want to create and/or edit a specific type of investment plan.

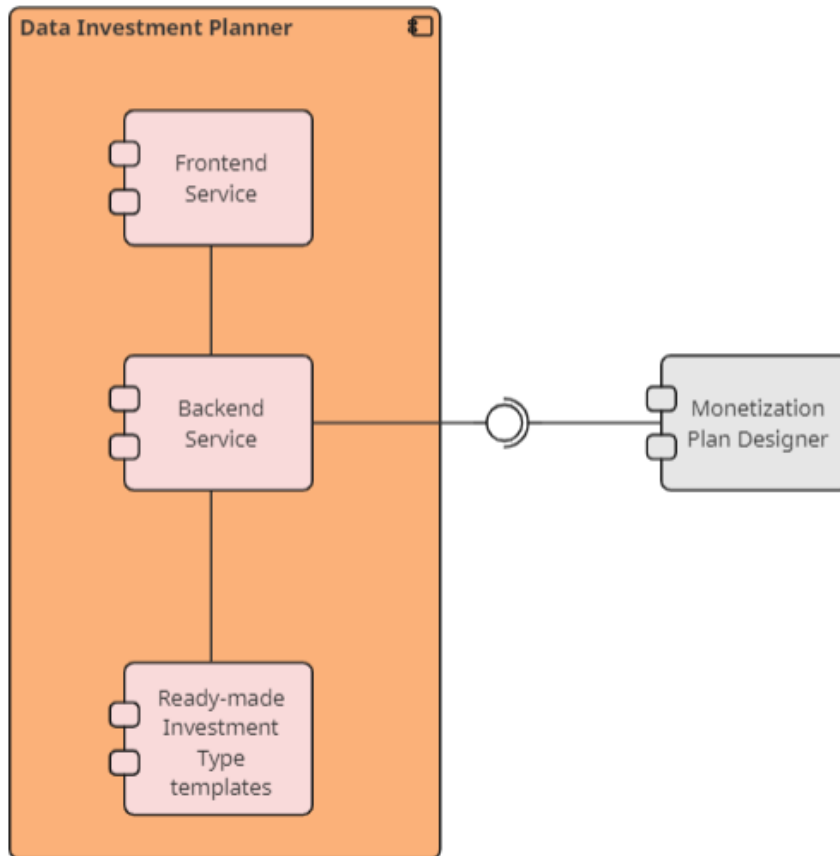


Figure 3-8: Data Investment Planner’s Internal Architecture

### 3.4.7 Mock-ups and Screenshots

**Monetization Method**  
Info here

---

**One-off Sale**

Info here

**Subscription**

Info here

**NFT**

Info here

**Investment Plan**

Info here ✓

Search for / select / edit / create new plan

or [create new plan](#)

**Investment Plan Title \***

**Total Equity Percentage \***  
 %

**Minimum Equity Percentage \***  
 %

**Equity Price \***  
 STC

**Max No of Investors \***

Figure 3-9: Data Investment Planner UI

### 3.5 PISTIS MARKET INSIGHTS

#### 3.5.1 Component Description

The PISTIS Market Insights component is responsible for offering insights about the transactions performed in the PISTIS Marketplace, by running designated analytics on the available transaction metadata residing in the different protected data ledgers of PISTIS stakeholders.

The PISTIS Market Insights component provides to the users predefined dashboards for exploring market relevant information (e.g. the PISTIS ecosystem transactions). The retrieved related transactions are indexed and specific models from the PISTIS Models Repository are exploited for the execution of analytics, based on the user query, while the analytics results are displayed in dashboards, allowing the user to understand in depth the specificities of transactions in PISTIS.

#### 3.5.2 Technology Background

The PISTIS Market Insights component will offer a frontend with the Insights Dashboard, built with Nuxt and Vue.js, that will constitute the UI with which the user will interact to create queries and view analytics results. The backend services are enabled with the Indexing Service and the Insights Engine, developed in Nest.js, that will search and retrieve the necessary transaction data and execute the related analytics on them respectively.

#### 3.5.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	Data Provider / Data Consumer	View the overall assets available on the platform, in relation to time	I can better understand the PISTIS market context	Alpha	Platform assets are displayed in dashboards	Done	PISTIS.OUS.08
US_02	Data Provider / Data Consumer	filter the available graphs based on some criteria (dataset domains, sellers countries, sellers size (large/SME/micro)	I can analyse the insights more effectively based on my preferences.	Alpha	Filters available to be applied on the graphs	Done	PISTIS.OUS.08
US_03	Data Provider / Data Consumer	View the sales I have made on my data and the income generated from them	I can keep records and monitor their value	Alpha	User's assets' sales and generated income displayed in dashboards	Done	PISTIS.OUS.08
US_04	Data Provider / Data Consumer	View the assets acquired over the platform, in relation to time	I can monitor their growth or decline	Alpha	Assets acquired over the platform are displayed in dashboards	Done	PISTIS.OUS.08
US_05	Data Provider / Data Consumer	filter the available graphs for the acquired assets based on some criteria (value, type of buyer, domain	I can analyse the insights more effectively based on my preferences.	Alpha	Filters available to be applied on the graphs of acquired assets.	Done	PISTIS.OUS.08



		of dataset, license) etc					
US_06	Data Provider / Data Consumer	View projections regarding the market value in the near future	I can take informed decisions about investments strategies on new assets	Beta	Market value projections are displayed in dashboards	Upcoming	PISTIS.OUS.08

### 3.5.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The PISTIS Market Insights should provide a user-friendly interface to allow users view all the assets available on the PISTIS Market and those acquired over the platform	US_01	
FR_02	The PISTIS Market Insights should provide a user interface to enable data providers view the data asset sales they have made and the generated income	US_02	
FR_03	The PISTIS Market Insights should provide a user-friendly interface to allow data consumers to see the data assets they acquired over the platform	US_03	
FR_04	The PISTIS Market Insights should provide a user interface that will enable users to filter the insights provided based on some criteria	US_04	
FR_05	The PISTIS Market Insights should provide a user interface that will present to users projections of the market value of their assets in the near future	US_05	

### 3.5.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement category	Sub-Id	Description (Detailed description of the requirement)
Performance efficiency	NFR1	The overall process shall be performed without delays and should not consume unnecessary system resources
Reliability	NFR2	The PISTIS Market Insights shall operate in a reliable manner, providing reliable analytics to the data provider.
Security	NFR3	The overall process shall be made through secure communication channels

### 3.5.6 Component's Main Elements and Internal Architecture

The PISTIS Market Insights consists of the following components:

- **Insights Dashboard:** The user interface that enables the users to see all of the available data assets in the PISTIS Market and search for specific datasets and view corresponding insights.

- Analytics/Forecasting Engine: The AI engine of the component that executes analytics for selected datasets to provide insights based on pre-defined models, such as forecasting trends for the market, etc.
- Indexing Component: It enables the indexing of transactions that happen in PISTIS, covering both the dataset acquisition transactions, and the placement of assets on the market.

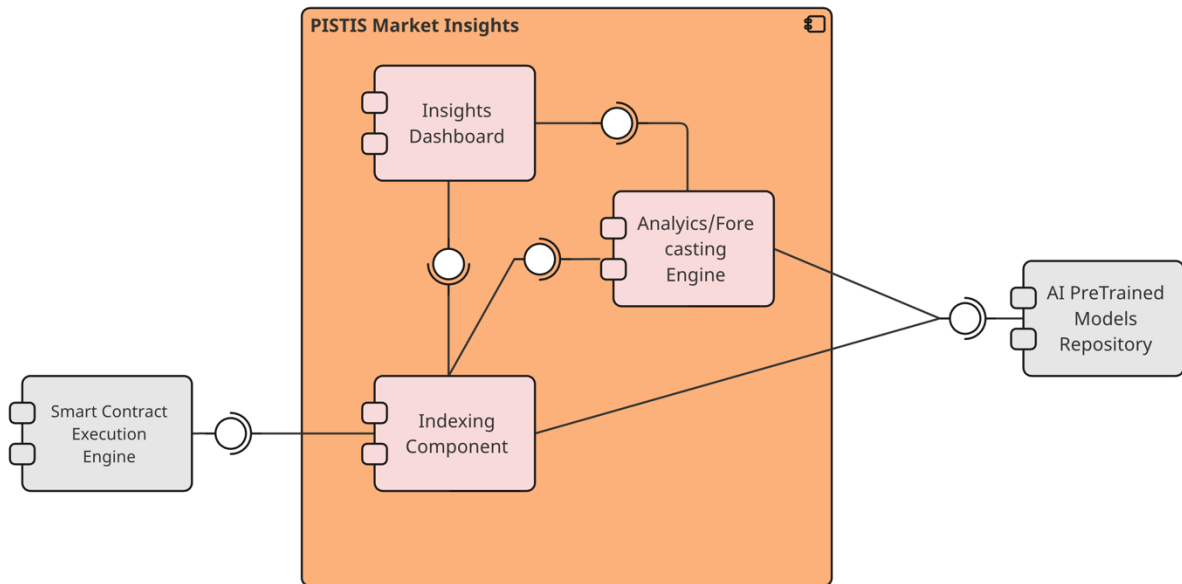


Figure 3-10: PISTIS Market Insights' Internal Architecture

### 3.5.7 Mock-ups and Screenshots

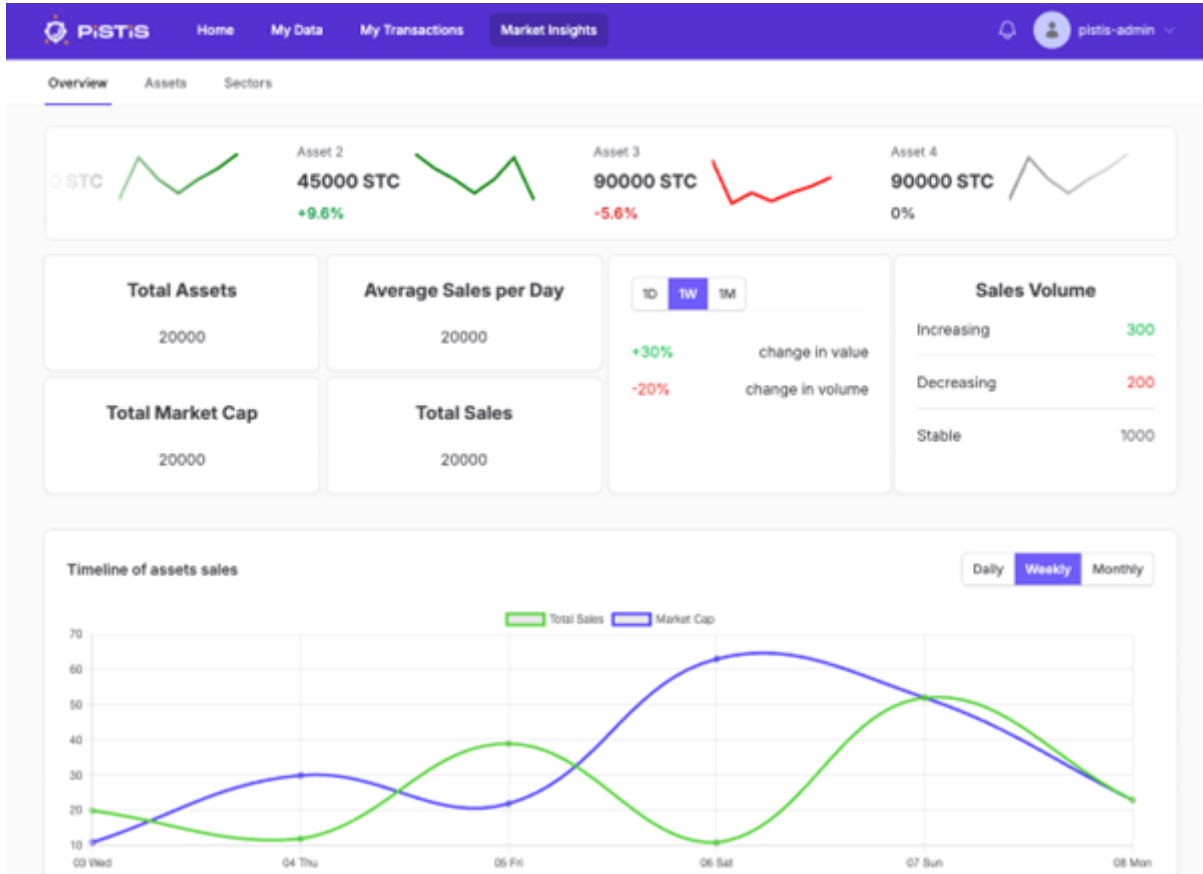


Figure 3-11: Overview Dashboard of PISTIS Market Insights

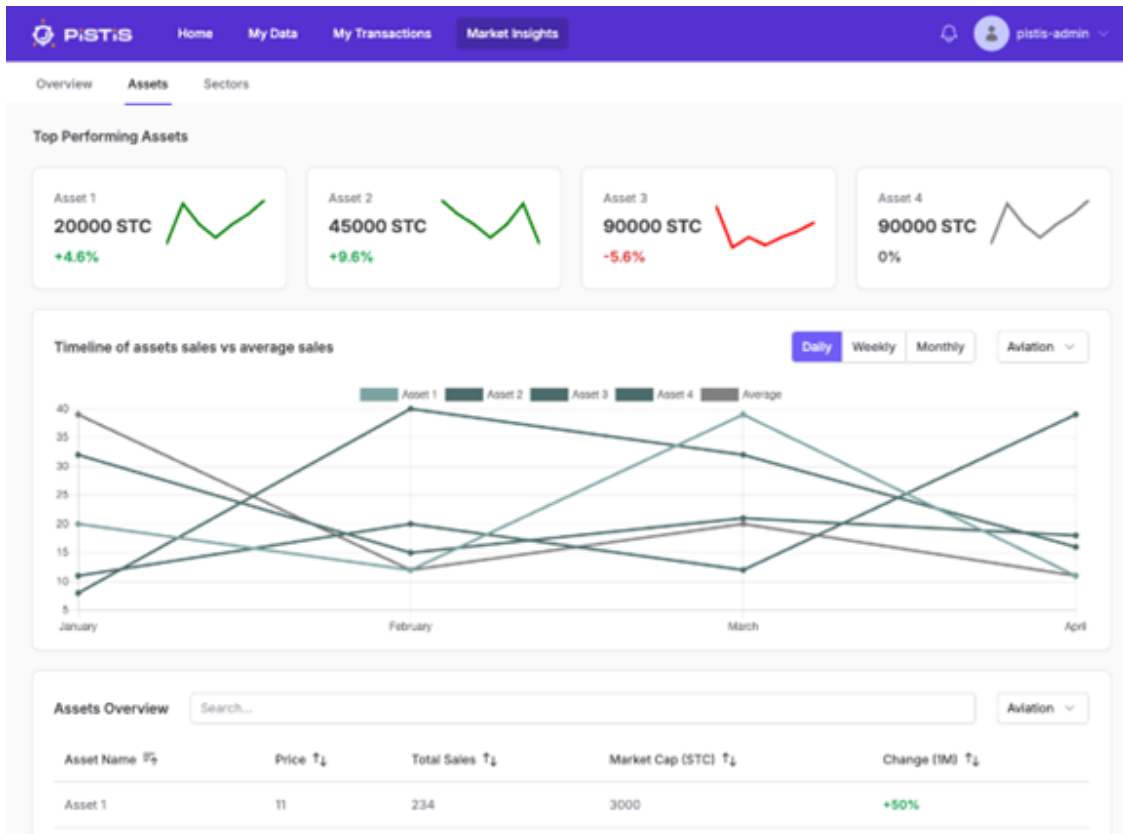


Figure 3-12: Dashboard of Market Insights showing Top Performing Assets

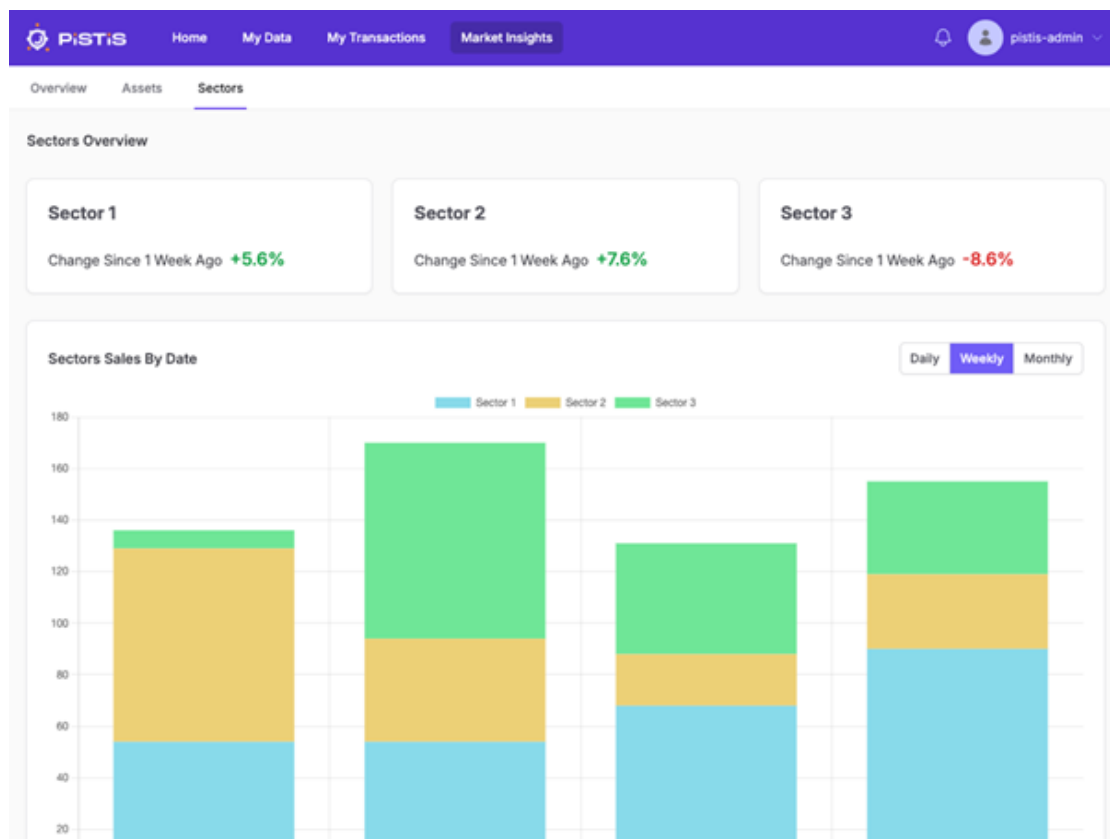


Figure 3-13: View of Sector Sales per Week

## 3.6 NFT GENERATOR

### 3.6.1 Component Description

The NFT Generator lies within the Data Monetisation bundle of the PISTIS Cloud platform and is responsible for generating the appropriate Data NFT. Upon user selection of NFT as the preferred monetization method for a given dataset, the NFT Generator component is triggered to create a new digital asset (NFT) that will be directly aligned with the corresponding dataset.

This component is triggered by the Monetisation Plan Designer, which provides all the required information for generating a digital asset (NFT). Utilizing this input, the NFT Generator will communicate with the Digital DLT-FIAT Wallet of the PISTIS Administrator on the PISTIS Cloud platform. The latter will mint and transfer the NFT to the Data Owner's wallet.

### 3.6.2 Technology Background

NFT Generator is a minimal standalone Rust web-based application, utilizing the underlying PISTIS Monetary Ledger which is based on IOTA. It can be considered a thin layer above PISTIS DLT-FIAT Cloud Wallet and especially around its digital assets' web API.

It resides only on the PISTIS Cloud infrastructure so that the PISTIS Administrator will be recorded as the NFT's immutable issuer. The minted digital asset will be automatically transferred to the Data Owner's wallet. The Digital DLT-FIAT Wallet is responsible for notifying

the end user about the completed transaction. The final NFT will be uniquely associated with the selected dataset by assigning appropriate immutable values as NFT properties.

### 3.6.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_1	Data Owner	Create an NFT representation of my dataset(s)	I can assure uniqueness, ownership, digital rights of them	Alpha	Generated NFTs are visible on wallet and can be transferred.	Done	PISTIS.OUS.06

### 3.6.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_1	Creates new unique NFT for the requested dataset and assigns it to the Data Owner's DLT-FIAT Wallet.	US_1	

### 3.6.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
<b>Performance efficiency</b>	NFR1	The overall transaction shall be performed without delays and should not consume unnecessary system resources
<b>Reliability</b>	NFR2	The component shall operate in a reliable manner, providing a reliable NFT to the data provider
<b>Security</b>	NFR3	The generation of the NFT shall be made through secure communication channels

### 3.6.6 Component's Main Elements and Internal Architecture

The component is comprised of a backend service which receives input from the Monetisation Plan designer toward generating an NFT for a given dataset. The component will act as an intermediate between users and the Digital DLT-FIAT Wallet, the latter will issue the NFT. Through the IOTA protocol and its consensus mechanism, the newly created NFT will be visible from the user's digital wallet.

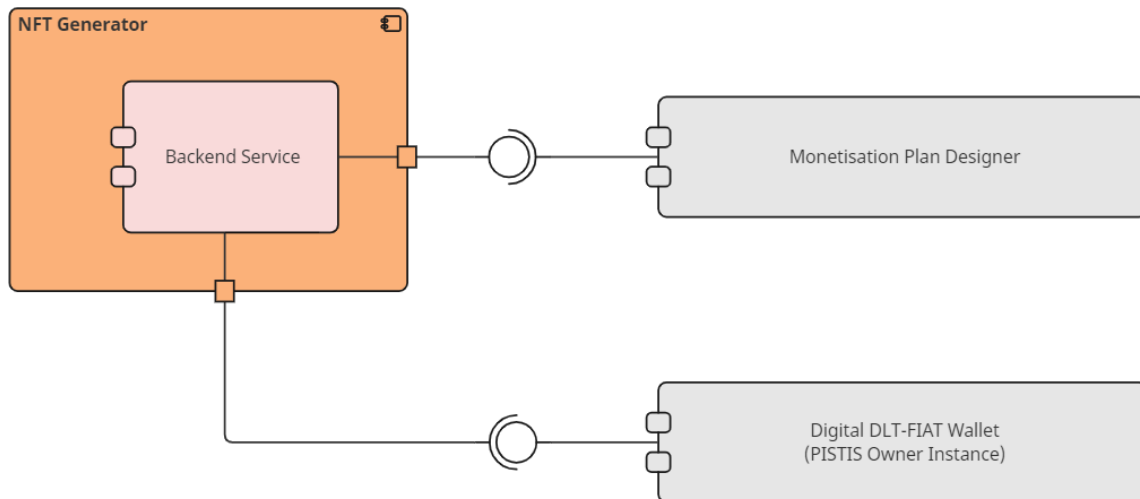


Figure 3-14: NFT Generator's Internal Architecture

### 3.6.7 Mock-ups and Screenshots

This component will not provide a corresponding UI Front End application.

## 3.7 MONETISATION PLAN DESIGNER

### 3.7.1 Component Description

The Monetisation Plan Designer, is an integral component of the Data Monetisation Bundle, located on the PISTIS Cloud Platform that is essential for seamless data interactions within the PISTIS ecosystem. It essentially enables data providers to put their datasets into the market and specify their desired monetization method, with the ultimate scope to simplify the complexities of the data transactions creating a streamlined and effective system that lays the foundation for a thriving data-centric ecosystem.

This component embodies a sophisticated mechanism that orchestrates two key options for data dissemination:

- I. *One-off purchase*, which enables instantaneous availability (i.e., on-demand access) of specific data assets or services, catering to immediate requirements of users without the need for a long-term commitment.
- II. *Subscription Plans*, which involves the provision of data or data-related services through tailored subscription plans (i.e. through a predefined subscription fee); towards ensuring a flexible and scalable approach to data accessibility. By offering various subscription models, data providers are empowered to structure their offerings with granularity, meeting the diverse needs of data stakeholders.

Two additional monetization methods are also provided to the data providers, including the selling of the NFT of his assets, where the data provider specifies the price of the NFT; and the creation of a Data Investment plan for the asset, where users shall specify the max number of investors, the total equity (%), the minimum equity (%) and the equity price.

To fulfil its functionalities the Monetisation Plan Designer communicates with a variety of data monetisation, trading services and value design components, depending on the users preferred monetisation route, including: the Asset Description Bundler where user can select

the dataset for which he/she will define the monetisation method and where the final user's monetisation plan details will be stored; the data FAIR Data valuation Service towards receiving data valuation suggestions. In the case of an NFT plan the component sends information to the NFT Generator to generate the relevant NFT; while in the case of an Investment Plan the component communicates with the Data Investment Planner towards sending the required information for the configuration of the appropriate Investment plan.

### 3.7.2 Technology Background

The Monetisation Plan Designer will offer a frontend service, developed in Nuxt.js and Vue.js, delivering the UI where the users will be able to select the preferred monetization route for their data. For the backend services of this component, Nest.js will be exploited, while intra-component communication will be facilitated with Rest APIs designed with Swagger.

### 3.7.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>	so that <Reason>				
US_01	Data Owner	I want to see valuation recommendations from FAIR Data Valuation Services	I may leave them as is or edit them to my liking	Alpha	Update if needed, the valuation recommendations	Done	PISTIS.OUS.06
US_02	Data Owner	I want to select the way to monetize my data	I can place this in the marketplace	Alpha	Select monetization method	Done	PISTIS.OUS.06
US_03	Data Owner	I want to be linked/taken to the Once-Off Sales Planner	I place this asset on sale as a simple commodity	Alpha	nPlace asset on sale as a simple commodity through the Once-Off Sales Planner	Done	PISTIS.OUS.06
US_04	Data Owner	I want to input the price for a once-off sale to my data	I define the price for it (can be also free)	Alpha	Define the price for once-off sale	Done	PISTIS.OUS.06
US_05	Data Owner	I want to choose how many times a buyer can download the dataset per day/week/month in the once-off sale	I place some constraints to save system resources	Alpha	Select the times a buyer can download the dataset per day/week/month	Done	PISTIS.OUS.06
US_06	Data Owner	I want to be linked/taken to the Subscription Planner	I place this asset on sale as a subscription	Alpha	Place the asset on sale as a subscription through the Subscription Planner	Done	PISTIS.OUS.06
US_07	Data Owner	I want to input the price for a subscription to my data	I define the price for it (could be also free)	Alpha	Define the price for subscription plan	Done	PISTIS.OUS.06

US_08	Data Owner	I want to choose the subscription period for my data	I can let the buyers know for how long they can get the data	Alpha	Set specific period in the subscription plan for the data	Done	PISTIS.OUS.06
US_09	Data Owner	I want to choose how many times a buyer can download the dataset per day/week/month in the subscription plan	I place some constraints to save system resources	Alpha	Select the times a buyer can download the dataset per day/week/month	Done	PISTIS.OUS.06
US_10	Data Owner	I want to be linked/taken to the NFT Generator when selecting this method of monetization	I may continue setting the desired parameters	Beta	View the details of NFT monetization upon selection of this method	Upcoming	PISTIS.OUS.06
US_11	Data Owner	I want to be linked/taken to the Investment Planner when selecting this method of monetization	I may continue setting the desired parameters	Beta	View the details of Investment Planner upon selection of this method	Upcoming	PISTIS.OUS.06

### 3.7.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The Monetisation Plan Designer should enable data owners to view the valuation recommendations from FAIR Data Valuation Services for their datasets	US_01	
FR_02	The Monetisation Plan Designer should enable data owners to edit the valuation recommendations from FAIR Data Valuation Services for their datasets	US_01	
FR_03	The Monetisation Plan Designer should allow data owners to select the monetisation method for their datasets	US_02	
FR_04	The Monetisation Plan Designer should enable data owners to select the Once-Off Planner to sell their datasets as simple commodities	US_03	
FR_05	The Monetisation Plan Designer should allow data owners to define the price for the once-off sale of their datasets	US_04	
FR_06	The Monetisation Plan Designer should enable data owners to select how many times a buyer can download their dataset per day/week/month in the once-off sale	US_05	
FR_07	The Monetisation Plan Designer should enable data owners to select the Subscription Planner to sell their datasets in a subscription plan	US_06	
FR_08	The Monetisation Plan Designer should allow data owners to define the price for the subscription plan of their datasets	US_07	



<b>FR_09</b>	The Monetisation Plan Designer should allow data owners to define the subscription period in the subscription plan of their datasets	US_08	
<b>FR_10</b>	The Monetisation Plan Designer should enable data owners to select how many times a buyer can download their dataset per day/week/month during the subscription plan	US_09	
<b>FR_11</b>	The Monetisation Plan Designer should enable data owners to select the NFT Generator as monetisation method for their datasets	US_10	
<b>FR_12</b>	The Monetisation Plan Designer should enable data owners to select the Investment Planner as monetisation method for their datasets	US_11	

### 3.7.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
<b>Performance efficiency</b>	NFR1	The overall process shall be performed without delays and should not consume unnecessary system resource
<b>Reliability</b>	NFR2	The Monetisation Plan Designer shall operate in a reliable manner, applying the data provider's preferred monetisation plan to his/her datasets
<b>Security</b>	NFR3	The overall process shall be made through secure communication channels

### 3.7.6 Component's Main Elements and Internal Architecture

The Monetisation Plan Designer comprises the following components:

- **Frontend Interface:** The user interface (dashboard) that enables users to configure the dataset's monetisation plan.
- **Monetisation Plan Constructor:** This is a backend service that receives input from the Asset Description Bundler, and depending on the user's selected monetisation route, it communicates with the FAIR Data Valuation Service, the NFT Generator or the Data Investment planner to fulfil its operations. Finally, a monetisation plan is constructed, and this is sent back to the Asset Description Bundler

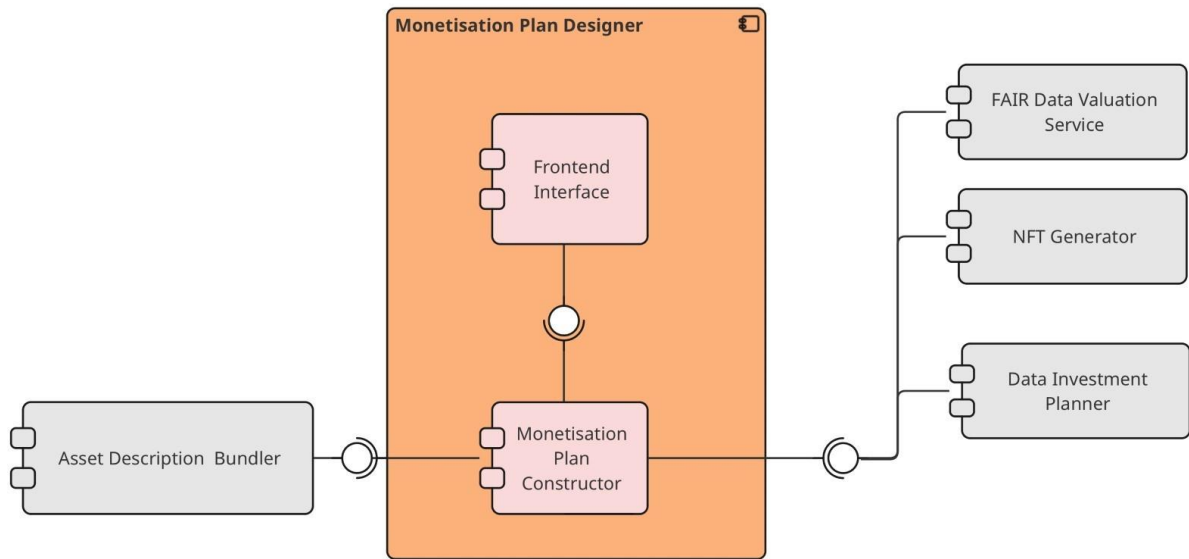


Figure 3-15: Monetisation Plan Designer’s Internal Architecture

### 3.7.7 Mock-ups and Screenshots

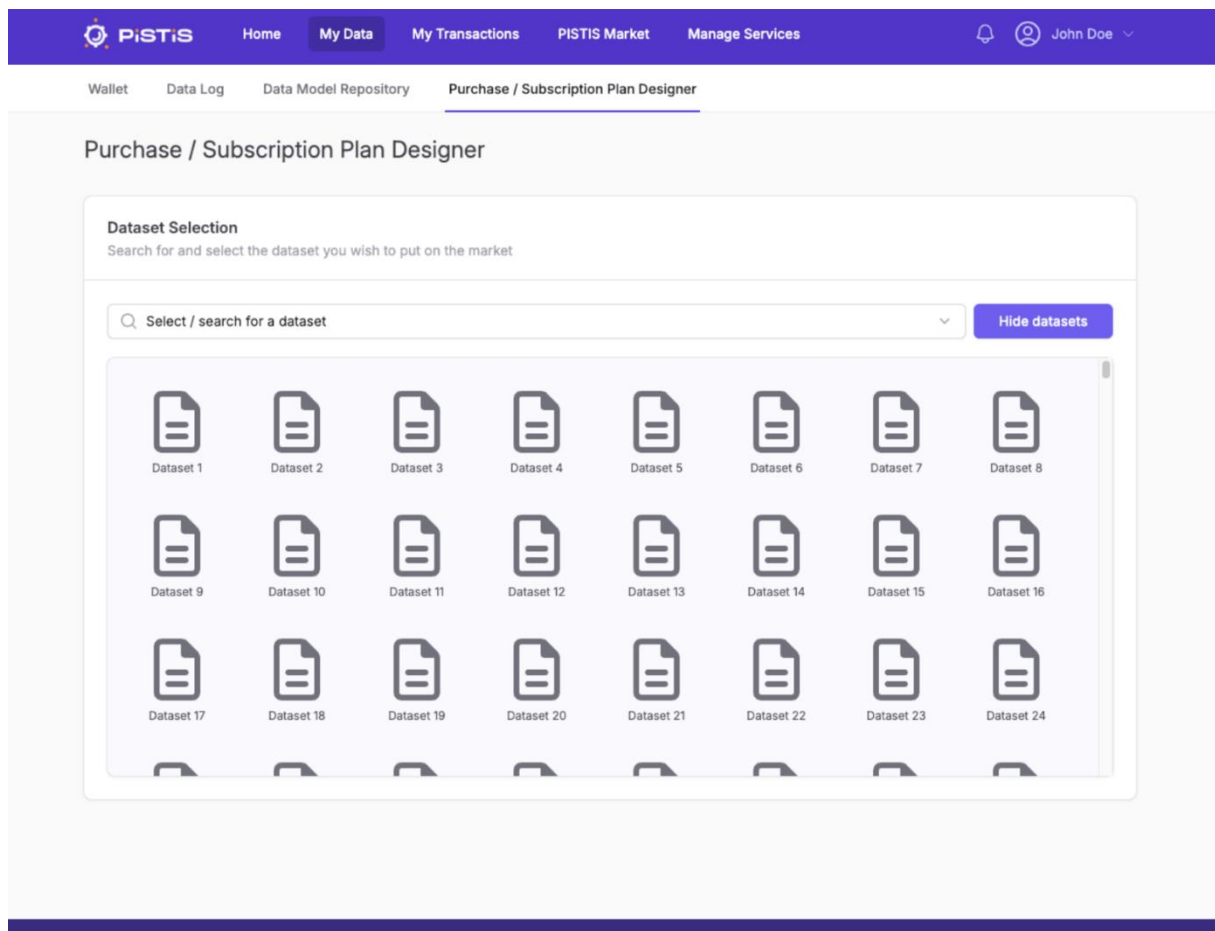


Figure 3-16: Monetisation Plan Designer – User’s owned datasets (main page)

Wallet Data Log Data Model Repository Purchase / Subscription Plan Designer

### Purchase / Subscription Plan Designer

**Dataset Selection**  
Search for and select the dataset you wish to put on the market

[← Select a different dataset](#)

Asset Title: **Dataset 5**

Asset Description: 5Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui.

**Complete Dataset**  
Select the complete dataset

**Query / Filter**  
Select a subset of the dataset

**Asset Offering Details**  
More info here

**Title \***  
Title of the asset

**Description \***  
Type a description for the asset here

**Keywords \***  
Type keywords separated by commas or use the Enter button

**FAIR Data Valuation Suggestions** [Learn More](#)  
Find out more here

Suggested One-off Price: **500 STC**

Suggested Subscription Price: **20 STC per month**

Figure 3-17: Monetisation Plan Designer – Dataset Selection

PISTIS Home My Data My Transactions PISTIS Market Manage Services John Doe

Wallet Data Log Data Model Repository Purchase / Subscription Plan Designer

**Asset Offering Details**  
More info here

**Title \***  
Title of the asset

**Description \***  
Type a description for the asset here

**Keywords \***  
Type keywords separated by commas or use the Enter button

**FAIR Data Valuation Suggestions** [Learn More](#)  
Find out more here

Suggested One-off Price: **500 STC**

Suggested Subscription Price: **20 STC per month**

**Monetization Method**  
Info here

**One-off Sale**  
Info here

**Subscription**  
Info here

**NFT**  
Info here

**Investment Plan**  
Info here

**One-off Sale Price \***  
Price of the asset  STC

**License \***  
Select license

**Download limit \***  
Number of times

**Frequency \***  
Select a frequency

**Terms and conditions \***  
Type the terms and conditions of your license here

Figure 3-18: Monetisation Plan Designer – One off Sale

## 3.8 DATA USAGE AND INTENTIONS ANALYTICS

### 3.8.1 Component Description

The Data Usage and Intentions Analytics located within the Monetization XAI Engine of the PISTIS cloud platform is designed to unravel the complexities of data utilisation within an organisation by delving into the underlying intentions of data owners and users.

The Data Usage and Intentions Analytics enables the administrators to create two different types of questionnaires towards capturing the intended usage or value of a dataset: (a) a questionnaire for verified buyers that have already purchased the dataset and can provide information about the way they exploit the dataset, (b) a questionnaire for any user that tracks the intentions of potential consumers of the dataset. This component allows the administrators to create multiple versions of a questionnaire in order to modify any questions and then select which of the questionnaire versions is active by defining also the user group(s) that may reply to it. The Data Usage and Intentions Analytics will allow any user to answer to a questionnaire for a dataset only one time. The replies of the questionnaires for a specific dataset will be analysed and the accumulated results for this dataset will be provided through comprehensive dashboards by the component to the dataset owner. The Data Usage and Intentions Analytics will facilitate the filtering of the accumulated results based on the user groups that replied to the questionnaire, while it will also provide the option for displaying these results in a timeline to show the evolution of value and intentions for the dataset during time. Moreover, by utilizing lineage information, the Data Usage and Intentions Analytics will allow data owners to view how their sold datasets are used and hence, understand better their usage.

In general, through the analysis of the questionnaires' replies this component enables data owners to clearly understand their datasets and the contextual landscape in which these assets may be employed. The insights gained from these visualisations empower data owners to make efficient decisions not only about how their data is processed but also about the manner in which it is shared, based on a comprehensive understanding of their data landscape; and altogether contributes to enhancing data quality and security.

The Data Usage and Intentions Analytics communicates with the PISTIS Data Catalogue for the retrieval of datasets and their metadata. In addition, the component communicates with the Data Ledger to retrieve the relevant datasets lineage and any recorded transactions made on similar datasets. By processing this input along with the answers to the questionnaires, the Data Usage and Intentions Analytics presents to users optimal methods for data exploitation.

### 3.8.2 Technology Background

The Data Usage and Intentions Analytics will offer a frontend service, developed in Nuxt.js and Vue.js, delivering the UI where the administrators will create questionnaires, the users will answer to them, and the data owners will view dashboards about the data usage and intentions for their datasets. For the backend services of this component, Nest.js will be exploited, while intra-component communication will be facilitated with Rest APIs designed with Swagger.

### 3.8.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	PISTIS Platform Administrator	Create a questionnaire to be answered by any user	I can capture information about intended usage (or value) of a dataset	Alpha	Create a questionnaire and receive answers by any user	Done	PISTIS.OUS.07/ PISTIS.OUS.08
US_02	PISTIS Platform Administrator	Create a questionnaire to be answered by any verified buyers	I can capture information about intended usage (or value) of a dataset	Alpha	Create a questionnaire and receive answers by any verified buyer	Done	PISTIS.OUS.07/ PISTIS.OUS.08
US_03	PISTIS Platform Administrator	Create multiple versions of the questionnaires	I can change/amend questions	Alpha	Create multiple questionnaire versions and update questions	Done	PISTIS.OUS.07/ PISTIS.OUS.08
US_04	PISTIS Platform Administrator	Select which questionnaire version is active at any time (and for which user group)	I can change its scope	Alpha	Define which questionnaire version is active	Done	PISTIS.OUS.07/ PISTIS.OUS.08
US_05	PISTIS User	Answer only once to the questionnaires for a specific dataset	I provide my viewpoint	Alpha	Only one answer can be provided by a user	Done	PISTIS.OUS.07/ PISTIS.OUS.08
US_06	Data Owner	View the accumulated scores of questionnaires for a given asset, filtered by the stakeholder group	I understand their value/intentions	Beta	Understand value/intentions of user groups for dataset based on accumulated questionnaire scores	Upcoming	PISTIS.OUS.07/ PISTIS.OUS.08
US_07	Data Owner	View the accumulated scores of questionnaires for a given asset, in a timeline graph	I understand how the value/intentions for a dataset have changed over time	Beta	Understand value/intentions of dataset based on accumulated questionnaire scores	Upcoming	PISTIS.OUS.07/ PISTIS.OUS.08
US_08	Data Owner	See information regarding the	I understand how it is used by different buyers	Beta	Accumulated results to the	Upcoming	PISTIS.OUS.07/ PISTIS.OUS.08

		usage of a sold data asset			questionnaire		
--	--	----------------------------	--	--	---------------	--	--

### 3.8.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The Data Usage and Intentions Analytics should allow administrators to create questionnaire for a dataset to be replied by any user	US_01	
FR_02	The Data Usage and Intentions Analytics should allow administrators to create questionnaire for a dataset to be replied by verified buyers of the dataset	US_02	
FR_03	The Data Usage and Intentions Analytics should capture information about intended usage/value based on the replies of the questionnaires	US_01, US_02	
FR_04	The Data Usage and Intentions Analytics should allow administrators to create multiple versions of questionnaires	US_03	
FR_05	The Data Usage and Intentions Analytics should allow administrators to modify questions in the questionnaires	US_03	
FR_06	The Data Usage and Intentions Analytics should enable administrators to activate a specific version of the questionnaire	US_04	
FR_07	The Data Usage and Intentions Analytics should enable administrators to define the user groups that can reply to the active version of a questionnaire	US_04	
FR_08	The Data Usage and Intentions Analytics should enable users to reply only once to the questionnaire for a specific dataset	US_05	
FR_09	The Data Usage and Intentions Analytics should analyse the replies of the questionnaires and provide accumulated results to the data owners	US_06, US_07	
FR_10	The Data Usage and Intentions Analytics should allow data owners to filter the accumulated questionnaire results based on the user groups that replied	US_06, US_07	
FR_11	The Data Usage and Intentions Analytics should allow data owners to view the accumulated questionnaires results in a timeline graph	US_06, US_07	
FR_12	The Data Usage and Intentions Analytics should enable data owners to view information about the usage of their sold dataset	US_08	

### 3.8.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
Performance efficiency	NFR1	The overall process shall be performed without delays and should not consume unnecessary system resources

<b>Reliability</b>	NFR2	The component shall operate in a reliable manner, providing reliable analytics to the data provider
<b>Security</b>	NFR3	The overall process shall be made through secure communication channels

### 3.8.6 Component's Main Elements and Internal Architecture

The Data Usage and Intentions Analytics consists of the following components:

- Frontend Service: The user interface (dashboard) that displays the information from the results of the questions and of the usage tracking to owners of datasets.
- The Questionnaire Builder that allows the PISTIS Platform Administrator that allows to create and update questionnaires.
- The Questionnaire Forms, that is responsible for rendering the questionnaires to users and accepting their input.
- The Questionnaire Repository, that is used to save the results of the questionnaires.
- The Usage Calculation Service, that is used to calculate the metrics for the selected datasets towards providing insights.

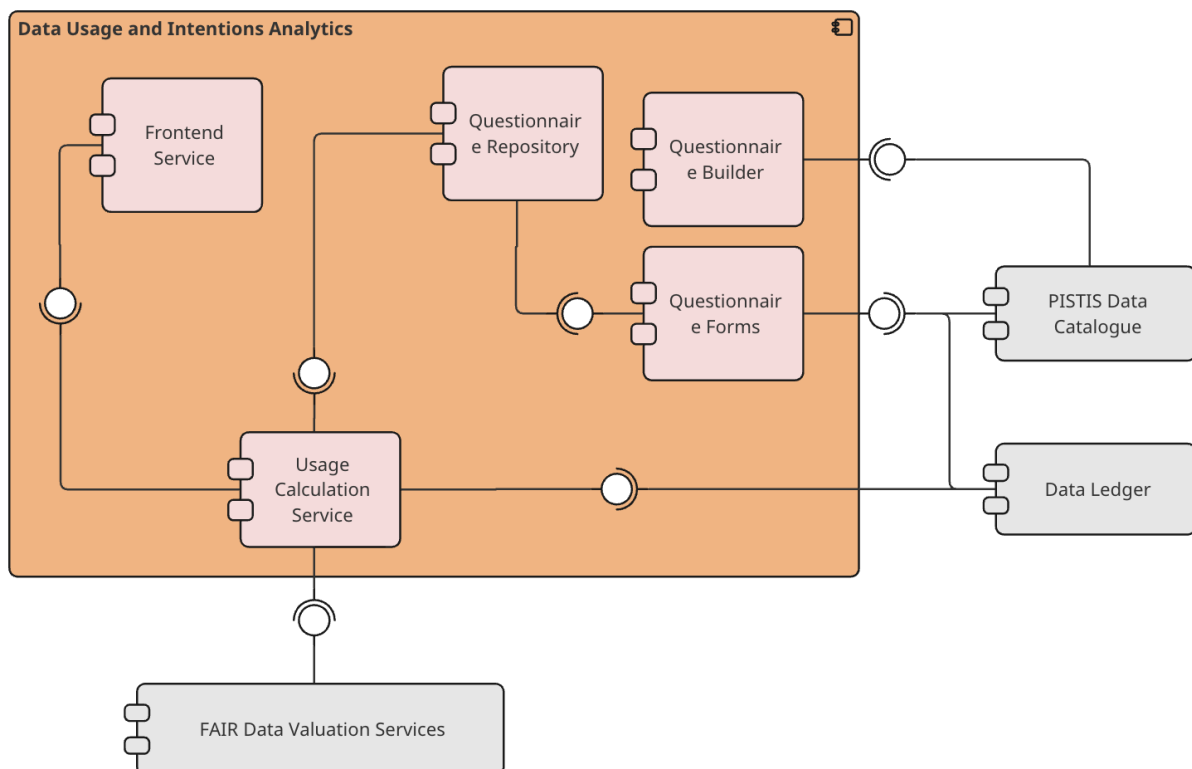


Figure 3-19: Data Usage and Intentions Analytics' Internal Architecture

### 3.8.7 Mock-ups and Screenshots

**Create New Questionnaire**

Title \*  
Questionnaire test 123

Description  
description

For Verified Buyers

**Question 1** Remove Question

Select Question Type \*  
Text

Title \*  
Question test

Is Optional

**Question 2** Remove Question

Select Question Type \*  
Checkbox

Title \*  
Checkbox question

Options \*

1

2

3

Figure 3-20: Questionnaire Builder

**Questionnaires**

Search Create

Title ↑	Version ↑	Publication Date
XYZ Test <span>For Verified Buyers</span>	1	Mar 21, 2024
Questionnaire Test <span>For Verified Buyers</span>	4	—
ABCD Questionnaire <span>For Verified Buyers</span>	2	Mar 21, 2024
Questionnaire for verified buyers <span>For Verified Buyers</span> <span>Is Active</span>	3	Mar 28, 2024
General users questionnaire <span>Is Active</span>	10	Mar 29, 2024
New general users questionnaire	8	Mar 28, 2024

Figure 3-21: List of available Questionnaires



## 4 LEDGERS BUNDLE

The Ledgers bundle facilitates through its services the storage of data and monetary transactions. It includes the PISTIS Data Ledger and the PISTIS Monetary Ledger of the PISTIS Market Exchange Platform.

### 4.1 PISTIS DATA LEDGER

#### 4.1.1 Component Description

Distributed Ledger Technologies (DLT) have received growing attention in recent years as an innovative method of storing data. It is a fully decentralized record-keeping system used to record information. Thus, such a technology is of paramount importance for storing the smart contracts, lineage information and data transactions in the context of PISTIS. The adoption of Blockchain in PISTIS symbolizes a commitment to leveraging cutting-edge technology for secure, transparent, and efficient data management. It aligns with modern trends in digital transformation, where trust, security, and decentralization are paramount. The integration of DLT not only enhances the PISTIS platform's operational efficiency but also fortifies trust among its users, making PISTIS a pioneering platform in the realm of secure and transparent data exchange and management. In addition, PISTIS Data Ledger component includes a private channel for storing sensitive data such as pointers that need to be protected under access control policies defined by the user or the organization and are attached to the specific data bundle.

#### 4.1.2 Technology Background

**Hyperledger Besu** Blockchain is an advanced Ethereum client that facilitates confining a permissioned and privacy enabled Ethereum network using the privacy group feature. Besu will be used as the underlying Blockchain technology since it is scalable, reliable, and offers secured off-chain privacy.

Besu's compatibility with the **Quorum network** enhances our platform's ability to manage privacy-focused agreements and ensures compatibility with a broad range of Ethereum-based tools and standards.

For the development and deployment of smart contracts, we use **Solidity**, the primary programming language for writing decentralized applications on Ethereum. This allows for the creation of complex contractual agreements that can be automatically executed within the blockchain environment.

#### 4.1.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	Data Consumer	have access on the GDPR compliance certificates	I can be sure that the dataset does not reveal any sensitive information	Alpha	GDPR certificate	Done	PISTIS.OUS.4 & PISTIS.OUS.7
US_02	Data Consumer	search datasets efficiently based	I can buy them	Alpha	Search results	Done	PISTIS.OUS.9

		on specific keywords					
US_03	PISTIS Administrator	review all transactions and query data	I can audit and monitor system activity	Alpha	Transaction auditability	Done	PISTIS.OUS.12
US_04	Data Provider	view history of data changes and purchases	I can stay informed about the usage and value of assets I provide	Alpha	Historical data transparency	Done	PISTIS.OUS.12
US_05	Data Consumer	purchase data securely	I can obtain the necessary data for my operations without compromising security	Alpha	Secure transaction processing	Done	PISTIS.OUS.12
US_06	Data Provider	list data for sale	I can monetize the data I own or manage	Alpha	Listing and pricing functionalities	Done	PISTIS.OUS.12
US_07	PISTIS Administrator	track dataset downloads	I can understand data consumption patterns	Alpha	Download metrics	Done	PISTIS.OUS.12
US_08	Data Consumer	acquire licenses for data usage	I can utilize the data in compliance with the terms	Alpha	Licensing clarity and enforcement	Done	PISTIS.OUS.12
US_09	Data Provider	store assets	I can store the assets to be sold in the blockchain	Alpha	Asset Storing	Done	PISTIS.OUS.12

#### 4.1.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	Data Ledger supports storing data from PISTIS components.	US_05, US_09	PISTIS components should authenticate themselves to the Data Ledger.
FR_02	Data Ledger supports querying data from PISTIS components.	US_02, US_03, US_04, US_06, US_07, US_08	PISTIS components should authenticate themselves to the Data Ledger.
FR_03	Data Ledger allows checking datasets based on GDPR compliance status.	US_01	Enhances the trust and compliance aspect by allowing users to verify GDPR status.

#### 4.1.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
Performance efficiency	NFR1	Storing and querying the Data Ledger should be performed in efficient way.
Usability	NFR2	Authorized entities should easily store or search data.
Reliability	NFR3	Store information should not be altered during time.
Security	NFR4	Only authorised entities should store or query data.
Portability	NFR5	Both PISTIS distributed ledgers should have synchronized information.

#### 4.1.6 Component's Main Elements and Internal Architecture

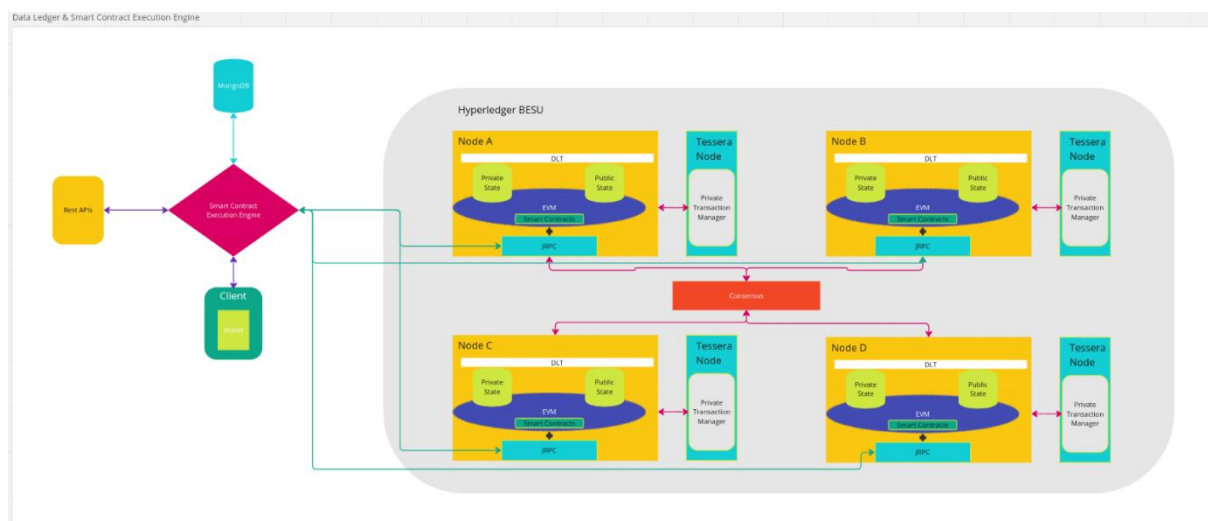


Figure 4-1: PISTIS Data Ledger's Internal Architecture

The architecture represents the PISTIS Data Ledger and the Smart Contract Execution Engine. The network is composed of four primary Besu nodes (A, B, C, D) (as an example, the nodes could be as many as we need), each associated with a corresponding Tessera node to manage private transactions and data. This setup enables a hybrid ledger where transactions can be processed in both public and private states, ensuring confidentiality when required. These nodes are orchestrated by a consensus mechanism to ensure data integrity and consistency across the network. The system's backbone is the Smart Contract Execution Engine that facilitates the deployment and operation of smart contracts through the Ethereum Virtual Machine (EVM) present in each Besu node. Clients (in our case, components), equipped with wallets, interact with the network through REST APIs that communicate with the execution engine, enabling seamless transaction submissions and smart contract interactions. Additionally, the architecture includes MongoDB as an ancillary database, allowing for the storage and retrieval of off-chain data which complements the on-chain elements of PISTIS, thereby offering an integrated data management and smart contract execution environment.

#### 4.1.7 Mock-ups and Screenshots

No UI is provided as this is a backend component.

## 4.2 PISTIS MONETARY LEDGER

### 4.2.1 Component Description

DLTs (Distributed Ledger Technologies) are rapidly gaining popularity, transforming the way organizations operate in the digital world. It is a system for storing and managing data and digital assets in a secure, transparent and decentralized manner. It ensures that data is tamper-proof and allows for a high level of transparency by making information available to everyone on the network while, at the same time, offering several advantages over traditional databases. The IOTA protocol bypasses the natural bottlenecks of conventional DLTs, commonly used by other implementations and allows for a more scalable and efficient network. It eliminates the need for centralized validation and offers a decentralized and secure environment for transactions and data exchange. Unlike traditional cryptocurrencies, IOTA requires no fees for transaction processing. The amount deducted from the sender's wallet is the same as the amount added to the recipient's wallet, making transactions seamless and cost-effective.

This component will be the foundational infrastructure of the Digital DLT-FIAT Wallet application, implementing the essential functionalities regarding digital asset management. It is a vital component facilitating monetary transactions for data trading, using cryptocurrency within a private network.

The selection of IOTA DLT network in the monetary management of the PISTIS platform, represents an adoption of an open, feeless and scalable technology, designed to support value transfers. Its primary objective is to establish trust between users, without the need for any central monetary management authority, while enhancing the operational efficiency of the PISTIS platform.

### 4.2.2 Technology Background

IOTA is a DLT (Distributed Ledger Technology) that empowers individuals to have control over their private data and engage in asset ownership and quick, reliable trading without intermediary involvement. IOTA will be used as the background DLT technology that supports and implements the functional requirements of the PISTIS DLT-FIAT Wallet component. Each Wallet Component will intergrade an IOTA node that serves as an access point in the PISTIS Monetary Ledger. The latest IOTA DLT protocol version, named Stardust, will be used to support both Smart Contracts and Asset Tokenization. During the initial development stages, the deployed IOTA nodes will participate in the public Testnet. In the final solution, a dedicated PISTIS permissioned IOTA DLT will be provided, offering both L1 (native token) and L2 (smart contracts) network solutions as well as a dedicated permanode, a server that will store the entire history of transactions in the system.

### 4.2.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>	so that <Reason>				
US_01	Data Owner	View my wallet's assets, balance and recent activity	I can monitor my account and plan possible future actions	Alpha	Display FIAT and DLT wallet's balance	Done	PISTIS.OUS.11

US_02	Data Consumer	View my wallet's assets, balance and recent activity	I can monitor my account and plan possible future actions	Alpha	Display FIAT and DLT wallet's balance	Done	PISTIS.OUS.11
US_03	PISTIS Administrator	View my wallet's assets, balance and recent activity	I can monitor my account and plan possible future actions	Alpha	Display FIAT and DLT wallet's balance	Done	PISTIS.OUS.11
US_04	PISTIS Administrator	View ledger's history-activity	I can facilitate services like prices forecasting, market insights or dataset quality/pricing	Alpha	Display specific transaction information upon request	Done	PISTIS.OUS.11
US_05	Data Consumer	Transfer PISTIS tokens to Data Owner and commission fees to PISTIS Admin	I can purchase/invest on/subscribe to a dataset	Alpha	Transaction is visible in DLT wallet's recent activity	Done	PISTIS.OUS.10
US_06	Data Owner	Create NFTs	I can prove the ownership/share of a dataset	Alpha	Newly created NFT is visible in DLT wallet's digital assets	Done	PISTIS.OUS.06 PISTIS.OUS.10
US_07	Data Owner	Transfer NFT's	I can transfer/share ownership of a dataset	Beta	Transaction is visible in DLT wallet's recent activity	Upcoming	PISTIS.OUS.06 PISTIS.OUS.10
US_08	PISTIS Administrator	Deploy a smart contract	The data purchase/investment/subscription can be carried out based on the Data Owner's terms	Beta	Contract deployment transaction is visible in DLT wallet's recent activity	Upcoming	PISTIS.OUS.06
US_09	PISTIS Consumer	View a smart contract's terms	I can decide whether to accept them and proceed with the dataset purchase/investment/subscription/ or not	V1.00	Smart contract terms are visible in smart contract view entry point call	Upcoming	PISTIS.OUS.10
US_10	Data Owner	Sign a smart contract	I can authorize the smart contract's execution	Beta	Stakeholder smart contract signatures are visible in smart	Upcoming	PISTIS.OUS.10

					contract view entry point call		
US_11	Data Consumer	Sign a smart contract	I can authorize the smart contract's execution	Beta	Stakeholder smart contract signatures are visible in smart contract view entry point call	Upcoming	PISTIS.OUS.10
US_12	PISTIS Administrator	Sign a smart contract	I can authorize the smart contract's execution	Beta	Stakeholder smart contract signatures are visible in smart contract view entry point call	Upcoming	PISTIS.OUS.10
US_13	PISTIS Platform	Execute a smart contract	The transaction amount may be transferred, or the new ownership status may be recorded	Beta	Be notified of executed smart contract function outcome	Upcoming	PISTIS.OUS.10

#### 4.2.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	Monetary Ledger supports up to date digital asset ownership record.	US_01, US_02, US_03	The monetary ledger is responsible for keeping up to date records of the digital assets.
FR_02	Monetary Ledger supports storing previous transactions.	US_01, US_02, US_03, US_04	
FR_03	Monetary Ledger supports retrieving previous transaction information.	US_01, US_02, US_03, US_04	Data providers and consumers can only access the previous activity of their wallet. The PISTIS Administrator can oversee the whole ledger's history
FR_04	Monetary Ledger supports cryptocurrency and digital asset transactions.	US_05, US_06, US_07	
FR_05	Monetary Ledger supports smart contracts	US_08, US_09, US_10, US_11, US_12, US_13	

#### 4.2.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
<b>Performance efficiency</b>	NFR1	The overall transactions shall be performed without significant delays
<b>Reliability</b>	NFR2	Monetary Ledgers synchronizes do that all nodes are up to date and in consensus
<b>Functional Suitability, Reliability, Security</b>	NFR3	Information recorded on the ledger are immutable and cannot be changed or deleted
<b>Security</b>	NFR4	Only authorized entities (wallet owners) should be able to perform currency trade-related actions.

#### 4.2.6 Component's Main Elements and Internal Architecture

As stated in the introduction section, the PISTIS Monetary Ledger consists of multiple IOTA nodes running in the user's physical premises. These nodes constitute just one element within the entirety of the Digital DLT-FIAT Wallet component. More information can be found in the corresponding section 3.3.6.

#### 4.2.7 Mock-ups and Screenshots

This component will not provide a UI Front End application.

## 5 TRANSACTION SERVICES BUNDLE

The Transaction Services bundle allows through its services the creation, execution and validation of the different transactions with the use of smart contracts. It includes the following components of the PISTIS Market Exchange Platform:

- 1) On/Off Platform Contract Inspector
- 2) Transaction Auditor
- 3) Smart Contract Template Composer
- 4) Smart Contract Execution Engine
- 5) Data Factory User Wallet

### 5.1 ON/OFF PLATFORM CONTRACT INSPECTOR

#### 5.1.1 Component Description

The “on” part of the on/off-platform contract inspector leverages the information gathered by the lineage tracking component in conjunction with usage tracking to fulfil its core functions. Triggered by the smart contract execution engine, it retrieves usage tracking information, active licenses, and lineage tracking information and performs a variety of checks to ensure that all transactions adhere to the established rules of the active licenses.

Upon activation, the inspector engages in several activities. It fetches the current active license for a particular asset and the user's organization from the ledger, ensuring that any operations performed on the asset align with the license's stipulations. It also retrieves the transaction history of the asset from the blockchain ledger, which is essential for validating the claims regarding its past use or ownership with respect to the active license. Additionally, the inspector obtains lineage data from a lineage tracker, encompassing the asset's provenance, lifecycle events, and any transformations. The inspector then runs an inspection service to scrutinize the collected data against the license terms.

The component concludes its process by delivering a response to the smart contract execution engine. This response confirms the legitimacy of the submitted action or signals potential breaches of licensing terms. Overall, the On Platform Contract Inspector ensures transparency and adherence to licensing agreements in the digital asset trade.

The source code of the component is maintained in GitHub as a git repository under the PISTIS project and can be accessed via the respective link:

<https://github.com/PISTIS-Platform/contract-inspector-on-platform>



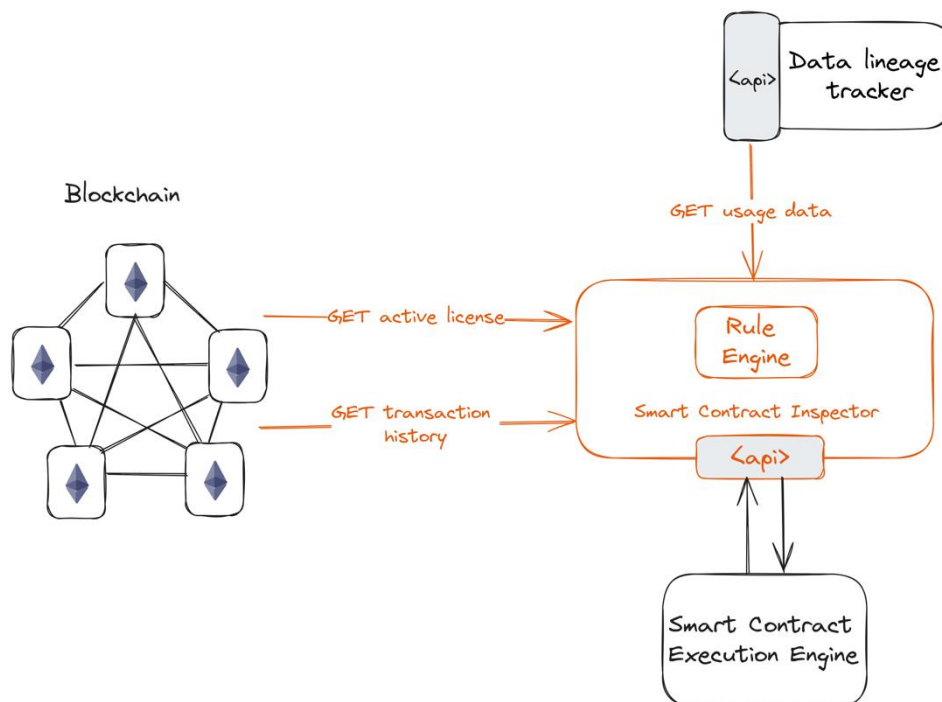


Figure 5-1: On chain platform inspection

The Off-Chain Contract Inspector is a component that targets the examination of datasets to ascertain their uniqueness across various granularity spectrums by employing 'fingerprinting'. This inspector operates autonomously, distinct from the on-platform counterpart, engaging in its activities external to the main data repository during the pivotal phase of data admission.

With its principal function centered on the scanning and juxtaposition of dataset records, this component analyzes the distinct fingerprints of each entry. The adopted approach will be adept at identifying any instances of exact or substantially similar records. This examination is instrumental in ensuring the datasets' veracity and safeguarding against any forms of unauthorized exploitation.

### 5.1.2 Technology Background

Both aspects of the PISTIS contract inspector (on and off) will be implemented based on the Java and Spring ecosystem. The implementation will follow a stateless approach for enhancing scalability by allowing requests to be processed by any instance of the service, facilitating easy scaling out. Stateless services are easier to manage, scale, and can recover more gracefully from failures, making them ideal for handling variable loads and ensuring high availability.

The identified integration interfaces with other components require extensive use of the RESTful paradigm. A REST will be also implemented for triggering the contract inspections. For the on-platform variation of the inspector, an integration with a Besu based blockchain is also required. Besu is an Ethereum client designed for enterprise use for both public and private permissioned blockchain networks. Integrating with the Besu blockchain allows the off-platform inspector to interact with blockchain data and smart contracts.

The main technologies that will be the utilized for implementing and delivering the on/off platform contract inspector according to the identified functionalities are the following:

- Java: versatile, object-oriented programming language known for its portability, performance, and robustness. It enables the development of cross-platform applications and is widely used for building enterprise-scale applications. Java's

extensive standard library and the rich ecosystem of third-party libraries and frameworks make it an ideal choice for developing complex data processing and analysis tools like the off-platform contract inspector.

- Spring: a comprehensive framework for developing Java applications. It provides a wide array of features that facilitate the development of high-performing, reusable, and easily testable components.
- Web3j: a lightweight, reactive Java library for integrating applications with Ethereum blockchains.

### 5.1.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	On Platform Contract Inspector	Fetch active license for an asset and user group	To run the inspection service	Alpha	License is fetched	Done	PISTIS.OUS.10
US_02	On Platform Contract Inspector	Fetch usage data from the blockchain for an asset	To run the inspection service	Alpha	Usage data is fetched	Done	PISTIS.OUS.10
US_03	On Platform Contract Inspector	Fetch lineage data from the blockchain for an asset	To run the inspection service	Alpha	Lineage data is fetched	Done	PISTIS.OUS.10
US_04	On Platform Contract Inspector	Run the inspection service	Respond to the smart contract execution engine that triggered the inspector	Alpha	Response is returned to the smart contract execution engine	Done	PISTIS.OUS.10

### 5.1.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_ON_01	The inspector gets triggered by the smart contract execution engine by passing the asset id and the id of the user that wants to perform a trading action	US_01, US_02, US_03, US_04	On platform functional requirements
FR_ON_02	The inspector retrieves from the ledger the active license for a particular asset and organization	US_01, US_02, US_03, US_04	
FR_ON_03	The inspector retrieves from the ledger the transaction history for a particular asset	US_01, US_02, US_03, US_04	
FR_ON_04	The inspector retrieves from the lineage tracker the lineage information of a particular asset	US_01, US_02, US_03, US_04	
FR_ON_05	The inspector evaluates whether the terms of license are satisfied based on the data acquired by the ledger and the lineage tracker and emits respective alarms/events if not	US_01, US_02, US_03, US_04	

ID	Description	Related Use Cases	Comments
FR_OFF_01	The inspector must generate unique cryptographic hashes (fingerprints) for each dataset entry upon receipt.	US_01, US_02, US_03, US_04	Off platform functional requirements
FR_OFF_02	The inspector should support different data formats (e.g., text, images, and binary data) for fingerprint generation.	US_01, US_02, US_03, US_04	
FR_OFF_03	The inspector must compare the fingerprint of each new or updated dataset entry against existing hashes in the dataset to identify duplicates.	US_01, US_02, US_03, US_04	
FR_OFF_04	The inspector should implement an efficient comparison algorithm to minimize processing time and resources.	US_01, US_02, US_03, US_04	
FR_OFF_05	The inspector must be capable of identifying both exact match duplicates and closely matching records based on a predefined similarity threshold.	US_01, US_02, US_03, US_04	

### 5.1.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
<b>Performance efficiency</b>	NFR1	Response times must not exceed 2 seconds under standard operating conditions.
<b>Reliability</b>	NFR2	The system must have an uptime of 99.9%, with redundancy mechanisms in place to handle failover scenarios without data loss.
<b>Reliability</b>	NFR3	The system should perform consistently under varying loads, ensuring stable operation during peak usage times.
<b>Security</b>	NFR4	All data transfers between the contract inspector components and the blockchain must be secured using the latest encryption standards.
<b>Monitoring and Logging Requirements</b>	NFR5	Comprehensive logging of all operations must be implemented to facilitate auditing and troubleshooting.
<b>Monitoring and Logging Requirements</b>	NFR6	The system must have monitoring capabilities that allow for real-time tracking of its performance and health.
<b>Scalability</b>	NFR7	The system must be able to scale out horizontally to accommodate increases in workload without a significant increase in latency.

### 5.1.6 Component's Main Elements and Internal Architecture

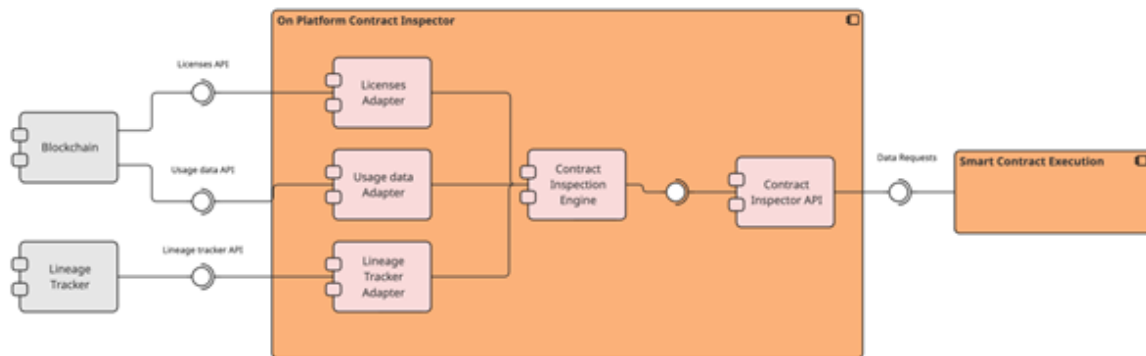


Figure 5-2: On Platform Contract Inspector's Internal Architecture

At its heart, the Inspector component orchestrates the workflow, managing interactions among the different modules and exposing a REST API for external activation. Integral to the architecture are the adapters: the Lineage Tracker Adapter connects with the Lineage Tracker API to fetch asset lineage data; the Blockchain License Adapter interfaces with the blockchain to retrieve active license data; and the Blockchain Usage Data Adapter extracts the transaction history of assets from the blockchain. Each adapter encapsulates the specifics of API communication, response handling, and error management. The rule engine, also developed in Java, processes the data retrieved by the adapters. It evaluates the data against established licensing terms and, if it finds violations, the term of violation is returned to the smart contract execution as response.

The REST API serves as the entry point for the smart contract execution engine, providing endpoint that triggers the inspection process. Upon receipt of a request, including asset and user group identifiers, the Inspector component begins its workflow. It calls upon the adapters to gather the necessary data, which is then analysed by the rule engine. After processing, the Inspector compiles the results and issues a response through the REST API to the smart contract execution engine, marking the process's completion.

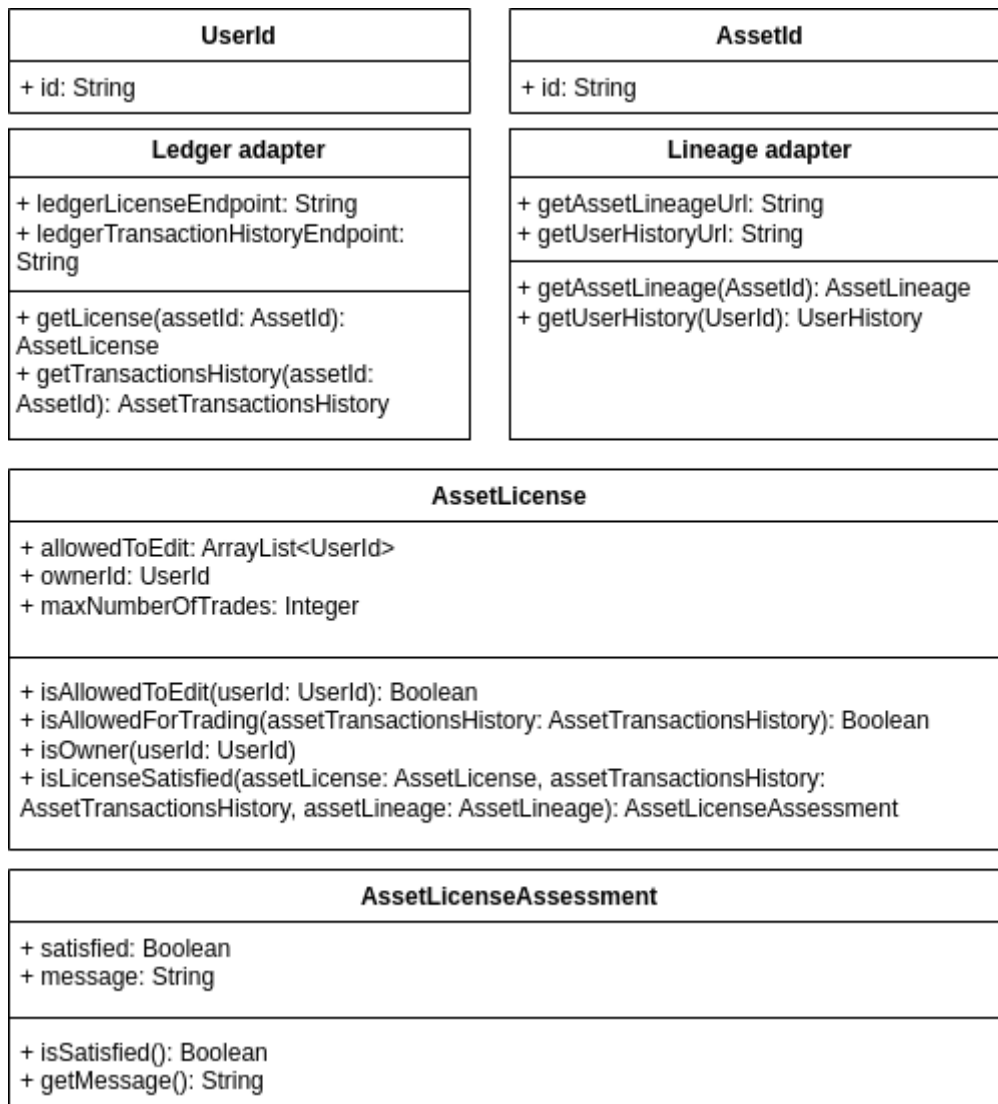


Figure 5-3: On Platform Contract Inspector core entities

### 5.1.7 Mock-ups and Screenshots

This component does not have a UI.

## 5.2 TRANSACTION AUDITOR

### 5.2.1 Component Description

The Transaction Auditor is responsible for monitoring the validity of the transactions in PISTIS. It enables users to search through different criteria (date/seller/buyer/transactionID) and select a transaction from the PISTIS ledgers and accompanies it with metadata requested and retrieved from the PISTIS Data Catalogue. In addition, the Transaction Auditor retrieves monetary details from the Monetary Ledger.

By analysing the aforementioned information about the selected transaction based on the corresponding smart contract, the Transaction Auditor evaluates the transaction and displays to the user the auditing info with all the related details for the selected transaction, which can be also downloaded as a report in a PDF format.

### 5.2.2 Technology Background

The Transaction Auditor provides a user interface that enables the users to search and select transactions for auditing. This frontend service will be implemented with Nuxt and Vue.js. The backend services of the component will be built with Nest.js.

### 5.2.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	PISTIS User	Search for a specific transaction (by date/seller/buyer/transactionID)	I can view the full details about it (coming from both the Data and the Monetary Ledger)	Alpha	View the full details of the selected transaction	Done	PISTIS.OUS.12
US_02	PISTIS User	download a PDF report regarding all the details about a specific transaction	I can use it for future reference or logging purposes	Beta	Successfully download a PDF file with all transaction details	Upcoming	PISTIS.OUS.12

### 5.2.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The Transaction Auditor should enable users to search for a specific transaction through different search criteria, such as the date, the seller or the buyer of the transaction, the transactionID, etc.	US_01	
FR_02	The Transaction Auditor should enable users to view the full details for the transaction they have search for	US_01	
FR_03	The Transaction Auditor should allow users to download a PDF report including all the details of a searched transaction	US_02	

### 5.2.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement category	Sub-Id	Description (Detailed description of the requirement)
Performance efficiency	NFR1	The overall audit process shall be performed without delays and should not consume unnecessary system resources
Reliability	NFR2	The Transaction Auditor shall operate in a reliable manner, delivering credible audit results tot he users.
Reliability	NFR3	The Transactions Auditor shall provide notifications to the users
Security	NFR4	The overall data audit shall be made through secure communication channels

### 5.2.6 Component's Main Elements and Internal Architecture

The Transactions Auditor consists of the following components:

- **Frontend Service:** The user interface for the interaction of users with Transactions Auditor, where they may search with different search criteria for specific transactions and view their details.
- **Backend Audit Service:** The audit mechanisms of the component that provide the audit results for the data transactions.
- **Notification Service:** Generates the notifications that are sent to the users regarding the status of their data transactions after audit process. It also notifies auditors about the history of data transactions.

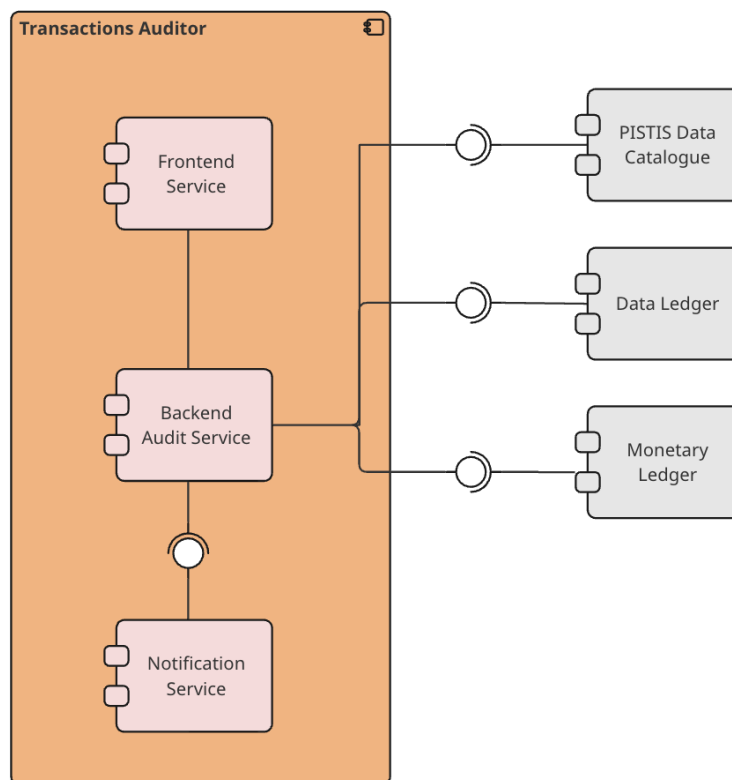
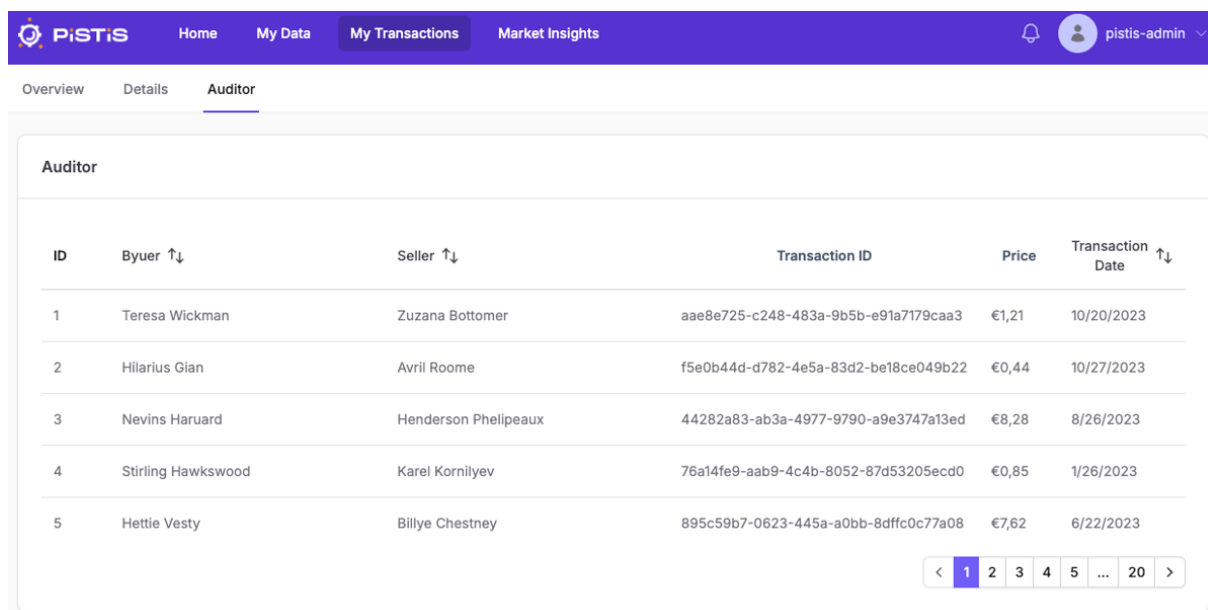


Figure 5-4: Transaction Auditor's Internal Architecture

## 5.2.7 Mock-ups and Screenshots



ID	Buyer ↑↓	Seller ↑↓	Transaction ID	Price	Transaction Date ↑↓
1	Teresa Wickman	Zuzana Bottomer	aae8e725-c248-483a-9b5b-e91a7179caa3	€1,21	10/20/2023
2	Hilarius Gian	Avril Roome	f5e0b44d-d782-4e5a-83d2-be18ce049b22	€0,44	10/27/2023
3	Nevins Haruard	Henderson Phelipeaux	44282a83-ab3a-4977-9790-a9e3747a13ed	€8,28	8/26/2023
4	Stirling Hawkswood	Karel Kornilyev	76a14fe9-aab9-4c4b-8052-87d53205ecd0	€0,85	1/26/2023
5	Hettie Vesty	Billye Chestney	895c59b7-0623-445a-a0bb-8dff0c77a08	€7,62	6/22/2023

Figure 5-5: List of Transactions and Transaction IDs

## 5.3 SMART CONTRACT TEMPLATE COMPOSER

### 5.3.1 Component Description

The Smart Contract Template Composer overall purpose is to allow the Asset Description Bundler to select the appropriate templates and transform the readily available smart contract templates into the corresponding code, so it can be then sent to the blockchain, triggering the execution of the associated transactions.

As such this composer's responsibility lies in delivering a ready-to-execute contract (by selecting and filling in pre-designed contract templates), that is able capture the details of the agreements pertaining to the exchange of datasets, and that will be presented to the Data Provider prior to finalising the publication of a dataset in the PISTIS Data Catalogue.

Additionally, the Smart Contract Template Composer empowers data providers to personalise these templates, drafting smart contracts aligned to their specific requirements.

### 5.3.2 Technology Background

For the backend services of this component, Nest.js will be exploited, while intra-component communication will be facilitated with Rest APIs designed with Swagger.

The Smart Contract Template Composer will also offer a frontend service with a UI, developed in Nuxt.js and Vue.js, for displaying the list of pre-existing contract templates which users can utilise/edit to generate specific smart contracts satisfying their needs.



### 5.3.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	Data Provider	all the information that the Data Asset Bundler holds for the publication of a Dataset to be automatically filled in into a "template"	It can be executed as a smart contract	Alpha	Display of filled-in template	Done	PISTIS.OUS.06

### 5.3.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	The Smart Contract Template Composer shall enable a data provider to create a new transaction data contract using automatically filled-in templates.	US_01	
FR_02	The Smart Contract Template Composer shall enable a data provider to edit/confirm the preselected parameters in the contract template	US_01	

### 5.3.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
<b>Performance efficiency</b>	NFR1	The overall process shall be performed without delays and should not consume unnecessary system resources
<b>Reliability</b>	NFR2	The component shall operate in a reliable manner, providing reliable and validated contracts to the data provider
<b>Security</b>	NFR3	The overall process shall be made through secure communication channels

### 5.3.6 Component's Main Elements and Internal Architecture

The Smart Contract Template Composer consists of the following components:

- **Draft Contract Display Service:** A service that provides the draft contract back to the Asset Description Bundler to enable users to preview new contracts and edit specific pre-selected parameters, prior to finalising the Asset placement on the PISTIS Catalogue.
- **Asset Details to Contract Service:** This component executes the necessary function towards transforming the asset details into a ready to be executed smart contract. This is done by receiving input from the Asset Description Bundler that described the asset (e.g. asset metadata, monetisation plan, etc) and selecting the appropriate already available smart contract template to fill in.

- The Pre-Defined Smart Contract Templates, that correspond to different monetisation methods supported by the Monetisation Plan Designer, in order to be filled in by the previous service.

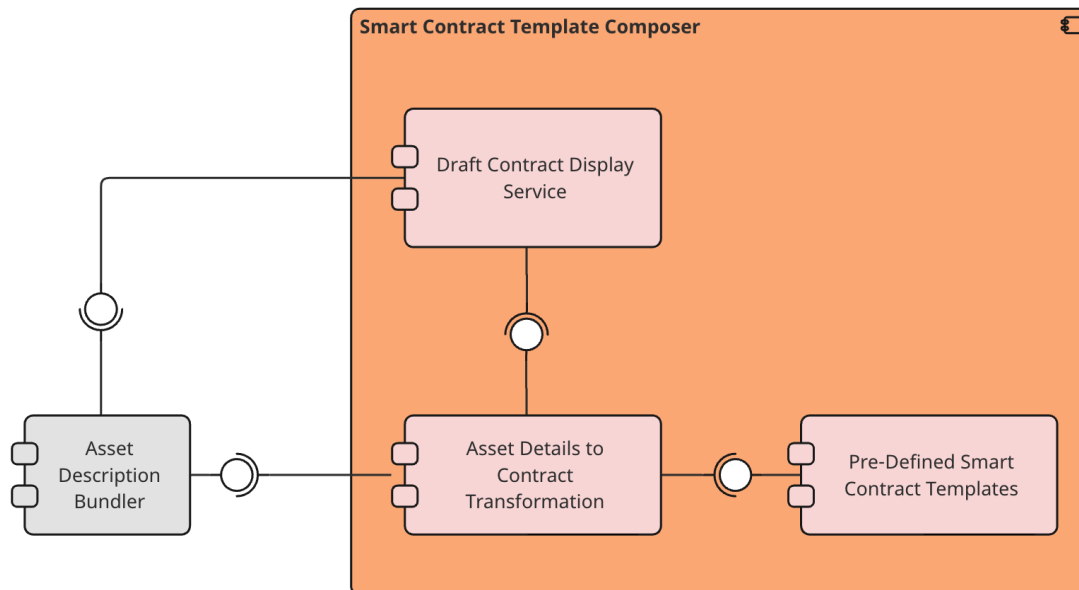


Figure 5-6: Smart Contract Template Composer's Internal Architecture

### 5.3.7 Mock-ups and Screenshots

This is a backend component and does not provide a UI.

## 5.4 SMART CONTRACT EXECUTION ENGINE

### 5.4.1 Component Description

The Smart Contract Execution Engine component, as its name suggests, is responsible for the actual execution of the smart contracts. Smart Contract Execution Engine executes all the stored and valid data trading smart contracts and makes the data artefacts available to the demand side. It has two modes of operation, one as part of the Data Exchange Preparation bundle and its functionality supports the PISTIS Data Factory and one as part of the Data Exchange Governance bundle and its functionality supports the PISTIS Cloud Platform. This component is responsible for the initiation of authorization and observability operations (e.g., triggering the Transaction Auditor or the Smart Contract Checker) prior to any contract execution to check the validity and the status of trading /sharing actions in a secure and proper manner. Figure 5-7 presents the high-level architecture of the Smart Contract Execution Engine including the internal components of the tool.

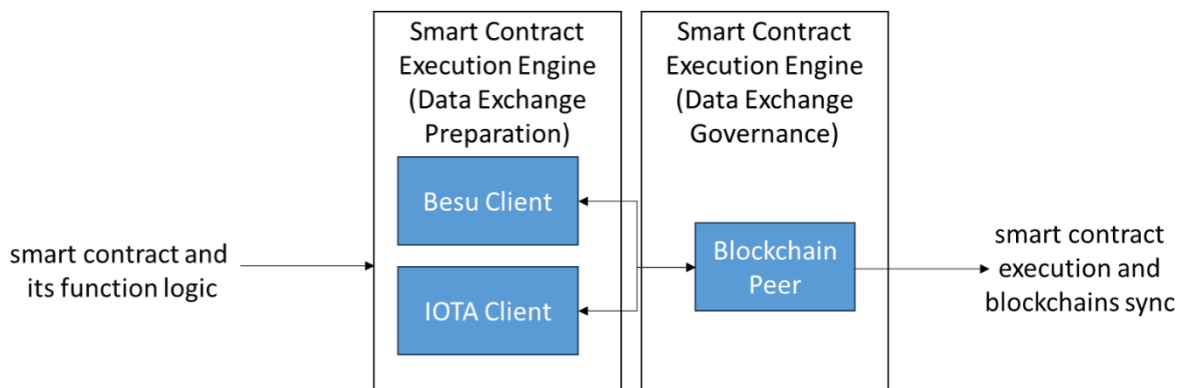


Figure 5-7: Smart Contract Execution Engine high-level Architecture

### 5.4.2 Technology Background

The Smart Contract Execution Engine leverages several advanced technologies to ensure robust, secure, and efficient handling of smart contracts within the PISTIS ecosystem.

Node.js serves as the backbone for our context broker. This JavaScript runtime is well-suited for building fast, scalable network applications. It allows our platform to handle numerous simultaneous connections with high throughput, which is essential for the demands of distributed ledger technologies.

The platform's backend functionality is developed in TypeScript, enhancing our development workflow with static typing and advanced JavaScript features, leading to more robust code and maintainability.

Data persistence and management are handled through MongoDB, a NoSQL database known for its high performance and flexibility. This database is integrated using Mongoose, an Object Data Modeling (ODM) library for MongoDB and Node.js, which provides a straightforward schema-based solution to model our application data.

We also utilize Web3.js, a collection of libraries that allow our server to interact with a local or remote Ethereum node using HTTP, IPC, or WebSocket. This includes sending Ether, interacting with smart contracts, and reading block data, which are crucial for the operation of our smart contracts.

Additionally, Docker containers are employed to encapsulate the engine's environment, ensuring that our application can be deployed consistently across any platform. This aligns with our use of Elastic, the default search engine provided by Quorum, which supports data indexing and complex searches within the blockchain network, enhancing the auditability and traceability of transactions.

### 5.4.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	Data Consumer	view each transaction	I can have proof of transaction validity	Alpha	Smart contract execution	Done	PISTIS.OUS.10, PISTIS.OUS.11
US_02	Data Provider	keep each transaction	I can look back on the history of updates and transfers	Alpha	smart contract creation	Done	PISTIS.OUS.10, PISTIS.OUS.11
US_03	Data Provider, Data Consumer,	track all interactions with assets	I can ensure transparency and accountability in asset exchanges	Alpha	Smart contract interaction via Node.js server	Done	PISTIS.OUS.10, PISTIS.OUS.11

	PISTIS Administrator						
US_04	PISTIS Administrator	monitor real-time data usage	make informed decisions on data management and policy	Alpha	Real-time monitoring with Node.js and MongoDB	Done	PISTIS.OUS.10, PISTIS.OUS.11
US_05	Data Provider	register new assets on the platform	expand the marketplace offerings and track new entries	Alpha	Asset registration with Web3 and Solidity Contracts	Done	PISTIS.OUS.10, PISTIS.OUS.11
US_06	Data Consumer	access detailed reports of data usage	have an in-depth understanding of data consumption patterns	Alpha	Detailed report generation via Node.js and MongoDB	Done	PISTIS.OUS.10, PISTIS.OUS.11
US_07	System Integrator	integrate with external systems	enhance interoperability and streamline data flows	Alpha	Integration with external systems	Done	PISTIS.OUS.10, PISTIS.OUS.11

#### 5.4.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	Smart Contract Execution Engine supports smart contract execution	US_01, US_02, US_03	Smart Contract Execution Engine of both organisational boundaries and PISTIS platform sides
FR_02	Smart Contract Execution Engine supports synchronization between monetary and data transactions for both ledgers	US_01, US_02, US_03	Smart Contract Execution Engine of both organisational boundaries and PISTIS platform sides
FR_03	Real-time tracking of transactions for monitoring and verification.	US_01, US_04	Provides transparency and traceability of all transactions on the platform.
FR_04	GDPR compliance validation for all data transactions.	US_05	Confirms that data handling adheres to privacy regulations.
FR_05	Detailed reporting of data usage statistics for analytics.	US_06	Delivers insights on data interaction patterns within the system.

#### 5.4.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
Performance efficiency	NFR1	Smart Contract execution should be performed in efficient way.
Reliability	NFR2	Smart Contract execution should be performed only after performing the necessary checks.
Security	NFR3	Authorisation prior to smart contract execution is necessary.

## 5.4.6 Component's Main Elements and Internal Architecture

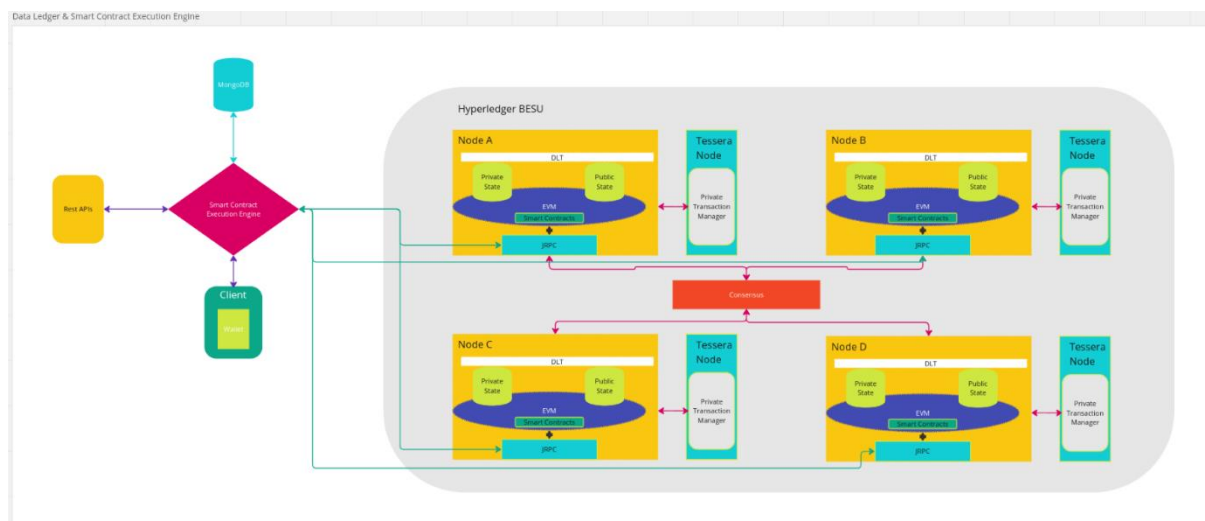


Figure 5-8: Smart Contract Execution Engine's Internal Architecture

For more information for the Smart Contract Execution Engine's internal architecture, please refer to section 4.1.6, as this component is closely connected with the PISTIS Data Ledger.

## 5.4.7 Mock-ups and Screenshots

As this is a backend component, no UI is provided.

## 5.5 DATA FACTORY USER WALLET

### 5.5.1 Component Description

The Wallet is designed to manage decentralized identities by leveraging decentralized identifiers (DIDs) and verifiable credentials (VCs). It allows users to securely and privately manage their identity data, storing credentials and selectively disclosing information as needed. Through verifiable presentations, users can show ownership of their credentials to verifiers without exposing more sensitive information. This approach is built on cryptographic operations that enhance privacy, security, and user control, adhering to the principles of Self-Sovereign Identity (SSI).

### 5.5.2 Technology Background

The Wallet is structured around two primary modules: the DAA-Bridge library and the DAA Core. The DAA-Bridge library is implemented in Java and facilitates the integration of DAA technology into any Holder's device. This library enables local Wallets to handle stored verifiable credentials (VCs) and self-issued verifiable presentations (VPs) with a high level of assurance (LoA). On the other hand, the DAA Core is developed in C and incorporates TPM technology alongside cryptographic libraries such as OpenSSL and IBMTSS. This core module is responsible for performing the advanced cryptographic operations necessary to secure and manage identity data within the Wallet. Together, these modules ensure robust security and high operational standards for managing digital identities.

### 5.5.3 Component Backlog

This section provides the full set of features that belong to the backlog of the component.

ID #	Use Case			Backlog Priority	Acceptance Criteria	Status	WP1 User Stories
	As a <Role>	I want to <Action>,	so that <Reason>				
US_01	As a user	I want a Digital Wallet that provides me with control over my identifying credentials, as well as capabilities for secure storage, access, and management of all my credentials	So that I can be sure that my Credentials, Keys and Certificates are securely stored and managed.	Beta	Certificates Storage and Management	In progress	PISTIS.OUS.02
US_02	As a user	I want my Digital Wallet to connect and get VCs from any certified Issuer - not only be able to store and manage a specific type of VCs.	So that it's easier to receive and store new credentials, I would like my Digital Wallet to seamlessly integrate with certified VC issuers, such as eIDAS, that issue Verifiable Credentials.	Beta	VC Issuance and Storage	In progress	PISTIS.OUS.02
US_03	As a user	I want to know what is the minimum information that a Factory Administrator requires from me	So that I can reveal only the required information, without needing to disclose any additional information.	Beta	Selective Disclosure	In progress	PISTIS.OUS.02
US_04	As a user	I want to have strong guarantees that no one can "clone" my credentials	So that I can be sure that no malicious party may forge my identity or use the cloned credentials on a different device.	Beta	User-Binding	In progress	PISTIS.OUS.02
US_05	As a user	I want to have the necessary guarantees that no one can forge verifiable presentations impersonating me	So that I can engage in online transactions and interactions with a high level of trustworthiness.	Beta	High level of trustworthiness	In progress	PISTIS.OUS.02

### 5.5.4 Functional Requirements

This section provides the functional requirements of the component.

ID	Description	Related Use Cases	Comments
FR_01	Storage and Management of Verifiable Credentials and Certificates	US_01, US_02	
FR_02	The wallet is secured	US_04, US_05	
FR_03	The wallet gives the control of the data to the user.	US_03	

### 5.5.5 Non-Functional Requirements

The following table presents the non-functional requirements of the component.

Requirement Sub-category	Id	Description (Detailed description of the requirement)
Functional Suitability	NFR1	Ensures completeness, correctness, and appropriateness of wallet functions.
Performance efficiency	NFR2	Wallet operates quickly and resource-efficiently under varying conditions.
Compatibility	NFR3	Functions correctly across different devices, operating systems, and interfaces.
Usability	NFR4	Easy to understand, learn, and operate by all users.
Reliability	NFR5	Consistently performs intended functions without failure.
Security	NFR6	Protects against unauthorized access and ensures data integrity.
Portability	NFR7	Easily adaptable to different environments without significant changes

### 5.5.6 Component's Main Elements and Internal Architecture

The diagram illustrates the high-level architecture of the Data Factory User Wallet component. At the outset, keys and requisite credentials are acquired from each organization and directed to the Wallet. Central to the operation is the DAA Library, acting like a Node.js server, managing the flow of data. Within this structure, the DAA Core is tasked with the intensive cryptographic processing. This core functionality facilitates the production of Verifiable Presentations, which can be selectively disclosed, providing essential privacy controls and data integrity.

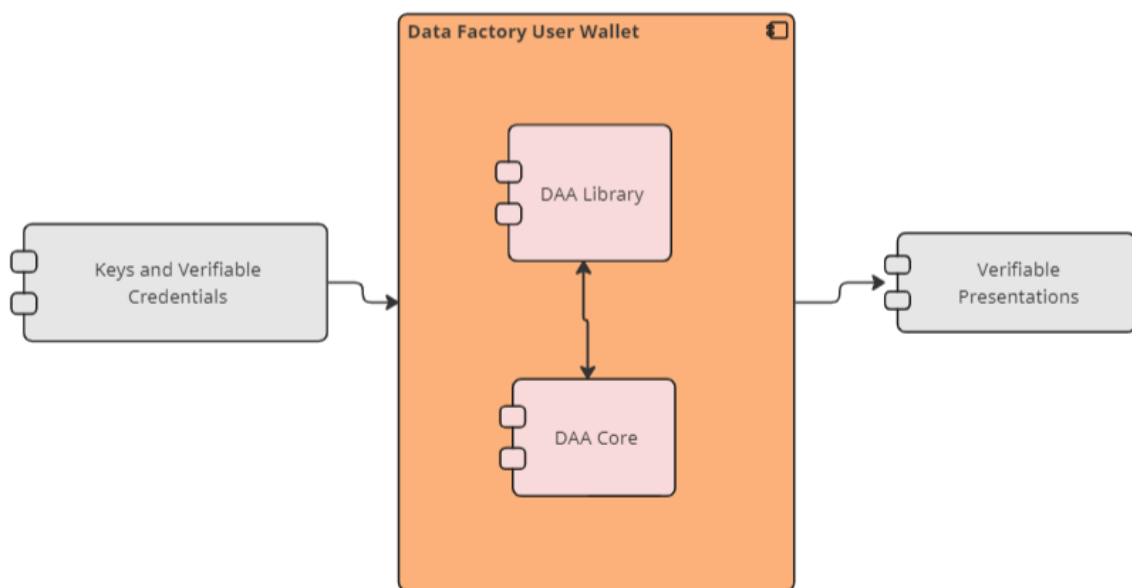


Figure 5-9: Data Factory User Wallet's Internal Architecture

### 5.5.7 Mock-ups and Screenshots

There is no UI provided by this component, as this is a backend component.



## 6 CONCLUSIONS

The deliverable D3.2 has documented the design and development of the alpha release of the different components that comprise the PISTIS Market Exchange Platform. The functionalities of each component have been described and the technologies that have been exploited for the development of the corresponding alpha releases have been presented. The user stories for each component have been specified and through them the functional and non-functional requirements of the different components have been defined. Moreover, the internal architecture of each component has been provided, showcasing the interconnections among the subcomponents of each component. For the components that have a user interface, respective screenshots depicting the components' functionalities have been also presented.

The design and development of the alpha releases of the components of the PISTIS Market Exchange Platform will drive the integration activities of the project in the next period towards realizing the delivery of the alpha version of the overall PISTIS Platform that will be described in D4.2. The components of the PISTIS Market Exchange Platform will continue to evolve with updates and new features that will be encapsulated gradually in the following beta and final releases of the components and will be documented in the corresponding D3.3 and D3.4 deliverables.