This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Num. 871754

i3-MARKET

Intelligent, Interoperable, Integrative and deployable open source MARKETplace

Deliverable D4.1

Decentralised Storage Specification and Reference Implementation V1

Deliverable No.	D4.1	**Status**	Final
Title	Decentralised Storage Specification and Reference Implementation V1		
Type	Other	**Dissemination Level**	Public
Work Package No.	WP4	**Work Package Title**	i3-MARKET Backplane for the Integration of Data Platforms and Market Economy
Version	1.1	**Date**	30-June-2021

Authors

Name	Partner	e-mail
Kaarel Hanson	GUARDTIME AS	Kaarel.hanson@guardtime.com
Andres Ojamaa	GUARDTIME AS	andres.ojamaa@guardtime.com
Juan Hernandez Serrano	UPC	j.hernandez@upc.edu
Rafael Genés Durán	UPC	rafael.genes@upc.edu
Jose Luis Muñoz Tapia	UPC	jose.luis.munoz@upc.edu

History

Rev.	Author(s)	Organisation(s)	Date	Comments
V0.1	Kaarel Hanson	GUARDTIME	24/05/2021	ToC Update from internal version
V0.2	Andres Ojamaa	GUARDTIME	30/05/2021	Deployment Plan and i3-MARKET V1 contributions
V0.3	Andres Ojamaa	GUARDTIME	01/06/2021	Operative Specification and Management Operative
V0.5	Andres Ojamaa	GUARDTIME	10/06/2021	End-user operative
V0.6	Andres Ojamaa	ATOS	15/06/2021	Final Contributions
V0.7	Kaarel Hanson	GUARDTIME	16/06/2021	Release version
V0.8	Filia Filippou	Telesto	20/06/2021	Internal Review QR
V0.9	Angel cataron	SIEMENS SRL	25/06/2021	Quality Review
V1.0	Martin Serrano	NUIG	30/06/2021	Final Version Approved for Submission

Rev.	Author(s)	Organisation(s)	Date	Comments
V1.0	Kaarel Hanson	GUARDTIME	25/09/2021	Revision of the first Release, Abstract is updated (page 5) and section 8 (page 36) is included.
	Filia Filippou	Telesto	25/09/2021	Internal Review QR
	Angel cataron	SIEMENS SRL	27/09/2021	Quality Review
V1.1	Martin Serrano	NUIG	27/09/2021	Final Version Approved for Submission

Key data

Keywords	Data Storage, blockchain, decentralised storage, distributed storage
Lead Editor	Name: Kaarel Hanson Partner: Guardtime
Internal Reviewer(s)	Name: Filia Filippou, Partner: TELESTO Name: Angel Cataron, Partner: SIEMENS S.R.L

Statement of originality

List of abbreviations

AB	Advisory Board
AMR	Annual Public Management Report
DESCA	Development of a Simplified Consortium Agreement
DoW	Description of Work
EC	European Commission
EU	European Union
EUPL	European Union Public Licence
FIA	Future Internet Assembly
GA	Grant Agreement
GNU	Generic Public Licence
GCC	GNU Compiler Collection
IERC	European Research Cluster on the Internet of Things
IMR	Interim Management Report
IP	Internet Protocol
IPR	Intellectual Property Rights
ITU-T	International Telecommunications Union
OGC	Open Geospacial Consortium
PMR	Periodic Management Report
PC	Project Co-ordinator
PMO	Project Management Office
PM	Project Manager
PO	Project Officer
PHP	Hyper Text Processor
QA	Quality Assessors
SB	Supervisory Board
SSN	Semantic Sensor Networks
STREP	Specific Targeted Research Project
TL	Task Leader
TMB	Technical Management Board
WP	Work Package
WPL	Work Package Leader
W3C	World Wide Web Consortium

Abstract

Every federated information system requires means to store and share data securely. i3-MARKET network is not an exception; hence, a well-thought solution that is secure, reliable and usable by all entities in the i3-MARKET network, is needed. The aim of data storage is to store common data in a federated network of data marketplaces. The common data shared between participating data marketplace instances may include identity information, shared semantic models, meta-information about data sets and offerings, semantic queries, sample data, smart contract templates and instances, crypto tokens and payments. No single party should fully control the data storage system and there shall be no single point of failure. In order to fulfil the needs of the aforementioned data types, two separate storage solutions are used: the decentralised and the distributed one.

The former supports the management of distributed identities and smart contracts. However, the latter has an important role in data synchronization between different i3-MARKET nodes, and, optionally, storage of data sets on sale. Moreover, the distributed storage supports non-repudiation service and auditable accounting.

The design of the distributed storage has been an iterative process. As the key component using the storage is the Semantic Engine, data synchronisation has been the major topic of discussion between Tasks 4.1 and 4.2. Several solutions were analysed and proposed, resulting in the implementation of a federated query engine index.

Data storage system takes full advantage of available base technologies and builds on top of these in order to satisfy i3-MARKET needs and requirements, with a focus on federated system architecture. The underlying technologies chosen for decentralised and distributed storage are Hyperledger BESU and CockroachDB, respectively.

The first prototype of the index management solution is available and integrated into the i3-MARKET network. Another concept, that is yet to be defined in more detail, is the storage of data that has been put on sale by the data owners. With this feature, the data owners can rely on the data storage capacity to hold the data on sale, without needing their own storage means.

The section 8 named technical contributions of i3-MARKET project includes a summary of the data Storage subsystem providing details on both structured database and ledger storage solutions. There are several contributions that the Data Storage subsystem brings to the i3-MARKET project as described in this section.

Design and analysis work has taken majority of the time, however, there are several features that are in line for the upcoming releases. A solution for storing datasets, together with logical workflows and interfaces, will be focused on in future releases. Moreover, managing the access to this data requires a properly designed access management solution, which in turn relies on reliable and secure key management solution. Finally, usage and monitoring of the storage capacity of the datasets for monetization is in line for completion by Release 2, the earliest.

Table of contents

List of tables

List of figures

1 About This Document

This is the first version of specification of Task 4.1, Decentralised data storage specification.

1.1 Deliverable context

Project item	Relationship
Objectives	TO.6 – Decentralised and distributed storage solutions provided by Task 4.1 are the base technologies supporting the implementation and operation of a trusted, interoperable and decentralised infrastructure.
Exploitable results	Marketplace Backplane The deliverable contributes to marketplace backplane by providing the underlying storage mechanisms for sharing data between marketplaces, managing identities, storing SLSs, verifiable claims, etc.
Work plan	The current deliverable is an output of T4.1 (WP4).
Milestones	The current deliverable is linked to milestones MS2 and MS3.
Deliverables	The deliverable has direct relationships with the following deliverables: - D2.2 - D2.3 - D3.1 - D3.3 - D3.5 - D4.3 It is foreseen that an updated version of current deliverable, D4.11, shall be released at MS6
Risks	Current deliverable mitigates the following risks from the list of Critical Implementation risks: 1, 2, 5, 8, 11, 14, 15

Table 1. Deliverable Context

1.2 Introduction

The current document presents the specification of the data storage component of i3-MARKET.

Section 2 provides the objective, architectural view and security aspects. Moreover, objectives are broken down into separate releases, R1 and R2.

Section 3 describes the features and use cases that are implemented and supported by the component.

Section 4 lists all requirements that have been engineered for the current component.

Section 5 presents the sequence diagrams of the features provided by the component, to give a better overview of the interactions between the different components in i3-MARKET.

In Section 6, the interfaces of implemented solutions are shown. The descriptions of the interfaces are provided as extracts from Swagger.

Section 7 gives a brief overview of the technologies used for the implementation of data storage component.

Section 8 concludes the work done and discusses about the work still do be done, including feature implementation and improvements.

2 System Specification

2.1 Objectives

The objective of Task 4.1 is to provide the data storage system for the i3-MARKET framework to store common data in a federated network of data marketplaces. The common data shared between participating data marketplace instances may include identity information, shared semantic models, meta-information about data sets and offerings, semantic queries, sample data, smart contract templates and instances, crypto tokens and payments. No single party should fully control the data storage system and there shall be no single point of failure.

The high-level capabilities that the data storage aims to provide are:

1. Decentralised Storage
2. Distributed Storage

The Decentralised storage shall provide highest available security guarantees in a federated network. The Decentralised storage subsystem will be built on a secure Byzantine fault tolerant consensus based distributed ledger. Due to high security requirements, the performance and storage space of such a system may be relatively limited compared to conventional databases.

The Distributed storage shall provide a database-like subsystem that is scalable, runs on a set of distributed nodes, has a rich query interface (SQL) and can handle large amounts of data.

Additionally, the data storage system provides an API to interact with the sub-components. In case of distributed storage, a custom API is implemented in order to provide ease of access to the features supported by the storage, however, the i3-MARKET shall rely on the API of the decentralised storage provided out-of-box.

2.1.1 Task Objectives R1

The objectives of T4.1 in R1 include both, decentralised and distributed storage. However, as the storage layer is dependent on the components that require storage features, the objectives are completed incrementally in coherence with the building blocks.

In the first release, the technologies for both capabilities were decided. The decision was made according to the technology needs of the relying building blocks of i3-MARKET. Initially, the aim was to deploy and configure the base technologies on the i3-MARKET network infrastructure for the first release, but the task was postponed to the second release due to unavailability of the infrastructure.

In terms of the features and functionality provided by the data storage, the decentralised storage supports the management of DID documents.

In R1, no specific implementation was done in the context of the data storage system, however, it was foreseen that the storage system implements functionality in subsequent versions. More specifically, it was foreseen to involve the data storage system in the process of data synchronization between the distributed storage and local triple stores of semantic engines.

Two alternative approaches for data synchronization have been proposed, depending on the overall system architecture:

- Data storage layer hosts a global triple store in addition to the local triple stores in each i3-MARKET instance.
- Data storage layer consists of a distributed SQL database that distributes updates to local triple stores deployed in each marketplace instance.

2.1.2 Task Objectives R2

During the validation of the solutions proposed for data synchronisation between different i3-MARKET instances, a decision was taken to design a more optimal solution. The initial proposed concept foresaw that the data stored in each Semantic Engine would be duplicated on the distributed storage. Such approach initially included data backup without additional effort, but in the end, it was decided that mirroring data is not needed. Instead, the distributed storage shall provide features to support federated data discovery queries.

For this purpose, the distributed storage stores an index that is consulted by the Semantic Engine in case a federated query is produced. The index consists of data categories mapped to the endpoint location addresses of the corresponding Semantic Engines. Among the aims of R2 is also to finalise the implementation of the index and the interfaces required for managing the index. Data storage subsystem exposes at minimum two endpoints for index management. One of the endpoints is used to update the index upon reception of new offerings by the SEED and the second one to query the index itself. The index is necessary for the creation of a federated query that allows data discovery in every marketplace connected to the i3-MARKET network.

Moreover, it is intended to provide data set storage features for data owners and providers, who do not have the means or capacity to store the data that is on sale on any of the marketplaces connected to the i3-MARKET network. In order to manage access to such data stored in the distributed storage, a suitable solution is needed. For this purpose, data storage shall interface with the smart contract manager, handling access and permissions, in order to avoid illegal or undesired access to the data.

2.2 Context

Figure 1 shows the data storage system in the i3-MARKET context. All components that need to persist some global state or use global data for operation will interact with the data storage system. The data storage system needs do interface with the Semantic engine system and the Trust, security and privacy system for access management. However, the interaction between the Data storage system and the Semantic engine system is not yet finalized and is subject to change during the course of implementation.

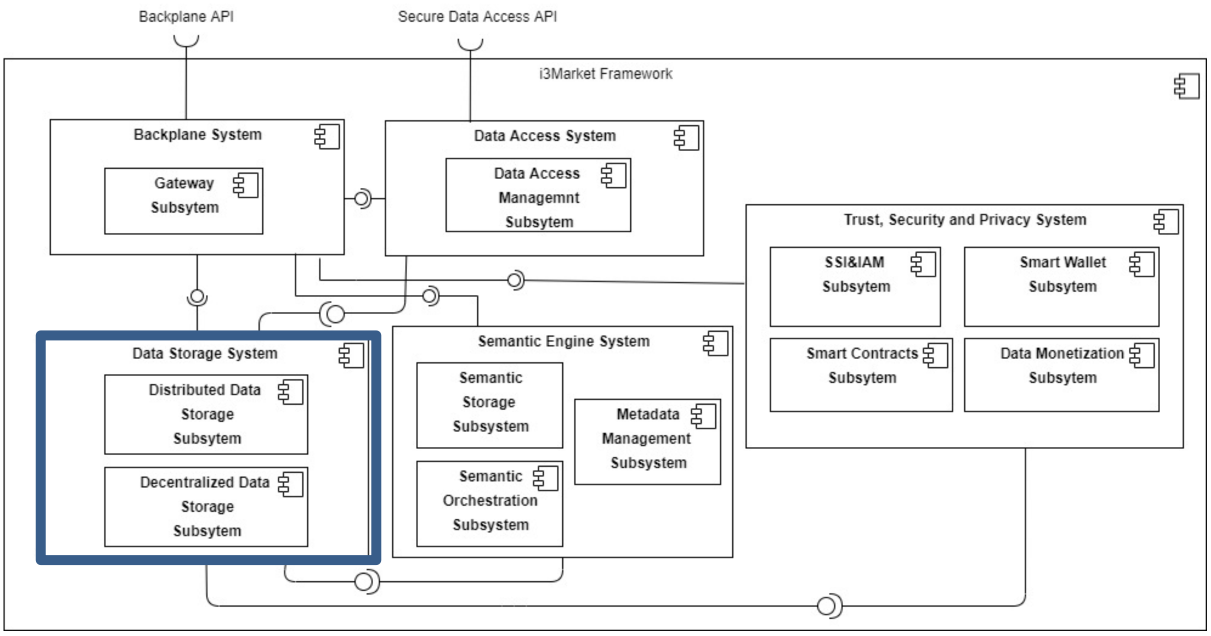

Backplane API
Secure Data Access API
i3Market Framework
Backplane System
Gateway Subsytem
Data Access System
Data Access Managemnt Subsytem
Trust, Security and Privacy System
SSI&IAM Subsytem
Smart Wallet Subsytem
Smart Contracts Subsytem
Data Monetization Subsytem
Data Storage System
Distributed Data Storage Subsytem
Decentralized Data Storage Subsytem
Semantic Engine System
Semantic Storage Subsystem
Metadata Management Subsystem
Semantic Orchestration Subsystem

Figure 1. Data Storage System in i3-MARKET context

2.3 Building block big picture

The Storage system consists of two main subsystems for implementing the decentralised storage and distributed storage features, respectively. The subsystems are, at least in the initial architecture, relatively independent of other systems and also with each other.

The diagram of Decentralised storage subsystem is shown in Figure 2. The Decentralised storage subsystem is implemented as a blockchain-based distributed ledger network. The software implementation is Hyperledger Besu in a permissioned setup using IBFT 2.0 consensus. Hyperledger Besu uses internally an embedded RocksDB instance for storing linked blocks (the journal of transaction) and world state (the ledger). Hyperledger Besu can instantiate and execute smart contracts for supporting the use cases of i3-MARKET framework.

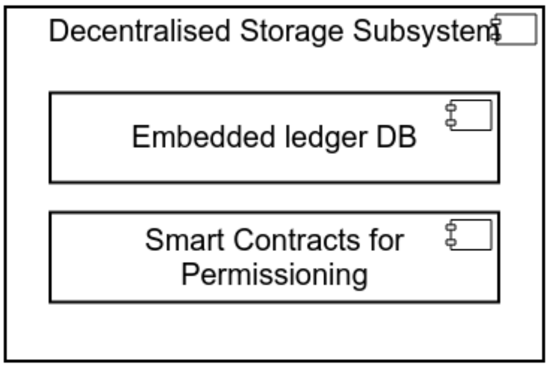

Decentralised Storage Subsystem
Embedded ledger DB
Smart Contracts for Permissioning

Figure 2. Decentralised Storage Subsystem

The components depending on the decentralised storage subsystem will use Hyperledger Besu's native JSON-RPC-based interface. A separate interface layer for accessing (or limiting access to) decentralized storage is not planned, as the nodes of the decentralised storage will already validate all transactions submitted to the ledger.

The diagram of Distributed storage subsystem is shown in Figure 3. The subsystem consists of a distributed cluster of database nodes and an optional interface layer (not implemented for R1). The database provides an SQL interface to other i3-MARKET framework components. The software implementation database is CockroachDB that can be accessed via PostgreSQL-compatible wire protocol for which a large number of client libraries exist in different languages and platforms. Only secure access to the database will be enabled, hence all clients need to use private keys and valid certificates to access the database.

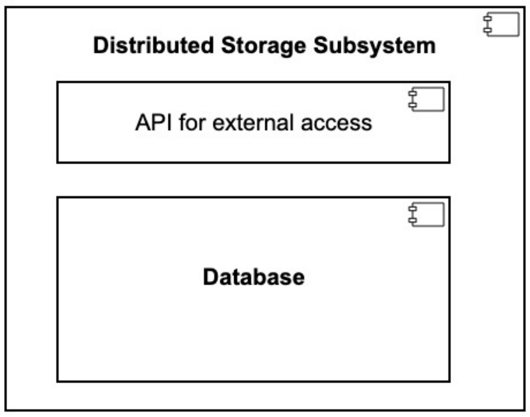


Figure 3. Distributed Storage Subsystem

2.4 Security

2.4.1 Authentication and Authorization

The distributed storage component is an internal component with no external access. That is to say that it will have connections only with other trusted services within the i3-MARKET backplane. Even though this simplifies the necessary measures in terms of authentication and authorization, it is still needed to secure machine-to-machine connections between the i3-MARKET services, since they can be deployed on shared infrastructure.

Although the approach may be reconsidered in the near future, the current solution relies on providing the distributed storage behind a TLS server endpoint and requiring TLS client certificates for the different connecting services. The setup will guarantee end-to-end security between the distributed storage service and any of its client services.

The governance of the certificates has followed up to now the *keep it simple* approach. The Distributed Storage system will be in charge of issuing the servers' and clients' certificates. For release 2, the definition of the governance rules for issuing certificates within the i3-MARKET federation, will be considered.

2.4.2 Service Availability

The storage subsystem is a critical component of the i3-MARKET network contributing to the proper functioning of the platform. Hence, appropriate measures in the form of design, choice of technologies and deployment, have to be applied. Fortunately, the two main subsystems used in the storage solution already have strong built-in availability features that will be summarised below.

2.4.2.1 Distributed Storage

The distributed storage solution is based on a CockroachDB server cluster consisting of four nodes. All data is replicated to at least three nodes before a transaction is considered committed. Therefore, data will be available even in a catastrophic event when half of the cluster is destroyed.

In the current setup, the database cluster can continue with normal transaction processing when three nodes out of the four are available. This feature guarantees the availability of the cluster in case of, for example, regular maintenance and upgrades of server software. CockroachDB supports the addition of new nodes as needed to support the load the component is required to process.

The federated search engine index service uses the CockroachDB server cluster as its storage backend. For availability, multiple independent instances of the index can be deployed. The system is designed to have horizontal scalability with no shared state between the instances.

2.4.2.2 Decentralised Storage

The decentralised storage used in the platform is a Hyperledger BESU network which uses the IBFT 2.0 (Proof of Authority) consensus protocol. In this network, there are 4 validator nodes based on the genesis configuration stored in the corporative Nexus. In this configuration, there are 3 accounts to be used by the i3-MARKET federation.

In this scenario, different components like the auditable accounting, are capable to deploy and manage smart contracts and transactions over those accounts.

3 Use Cases / Features

The distributed storage sub-component of data storage provides functionality to manage an index used for data discovery. The federated query engine index supports federated queries, a concept implemented by the Semantic Engine. Additionally, distributed storage will provide means to data owners and providers without storage capacity to store datasets on sale in the i3-MARKET network. Moreover, the distributed storage plays a vital role in supporting non-repudiation service and auditable accounting.

3.1 Federated Query Engine Index management

Distributed storage implements two main use cases – managing the index and querying the index – in order to provide the required functionality to the Semantic Engine for accessing the content of the index.

The index is a collection of data categories together with the endpoint location addresses of the corresponding Semantic Engines. One Semantic Engine is not limited to storing offerings belonging to one category, but to several of them. Hence, the index contains one to many relationships, linking a specific Semantic Engine to a set of data categories.

Every new marketplace joining the i3-MARKET network, will connect to the distributed storage through Semantic Engine. If the marketplace has been around for a while, the marketplace has most probably stored offerings metadata. This metadata should also be stored in the SEED to participate in federated queries. Therefore, such marketplace would have to populate the index by inserting category information to the distributed storage.

3.1.1 Manage

In order to provide the most recent and accurate information to the Semantic Engines in the i3-MARKET network, the index must be kept up to date at all times. Therefore, functions – insert, update and delete - to maintain the index are required. All these activities are limited to registered i3-MARKET nodes only and the authentication uses self-signed certificates.

3.1.1.1 Insert

Before an i3-MARKET instance receives any data offering registrations, the Semantic Engine has no reason to insert any content into the index. Although it is possible to insert an empty entry containing the endpoint address and an empty category list to the index, it is recommended to keep the index clean of unnecessary information. After receiving the first data offering, the Semantic Engine inserts the first entry to the index, revealing to other i3-MARKET instances the category of offerings stored in that specific Semantic Engine.

3.1.1.2 Update

Over the course of the market lifecycle, data offerings of different data categories are stored in a single marketplace. Upon the registration of a data offering belonging to category that is not yet present in the Semantic Engine, the Semantic Engine updates the index with relevant information (data category, endpoint address, etc.) by inserting a new entry to the database.

3.1.1.3 Delete

The final management activity of the index lifecycle allows the removal of entries from the index. It is the responsibility of the Semantic Engine to keep the index up to date, therefore, redundant and outdated information is removed from the index. In the event of closing down of an i3-MARKET marketplace instance, either temporarily for maintenance or indefinitely, the Semantic Engine has to remove unavailable content from the index. Moreover, this function should be accessible by a system administrator to remove relevant entries from the index, in case of a sudden shut down of a marketplace/i3-MARKET node.

3.1.2 Query

In case a Semantic Engine needs to perform a federated query among all other instances in the i3-MARKET network, the index shall provide input to the federated query. The Semantic Engine firstly queries the index with relevant parameters (data category, description, etc.) and the distributed storage shall return information from the index indicating which i3-MARKET instance contains the data that the SEED is looking for.

3.2 Data Set Storage

It is estimated that the majority of data owners and providers have the capacity, managed by themselves or by a marketplace, to store the data that has been put on sale on the marketplace. There is, however, a segment that does not possess such means, therefore, i3-MARKET aims to come to the rescue. As the distributed storage is a scalable and reliable storage solution, service is made available to the owners and providers of the data to store the data in i3-MARKET instead. Therefore, additional features are needed in order to:

1) store the datasets,
2) monitor the usage of capacity for monetization, and
3) access and download the data.

3.2.1 Store data

The owner or provider of the data can choose to store the data for sale on the connected i3-MARKET instance. It is up to the data owner or provider to decide when the data is uploaded to storage and made available to the consumers. The aim is to make the process as smooth and automated as possible, without the need of interventions by the owner/provider in the data purchase process.

3.2.2 Monitor capacity usage

In order to monetize the use of capacity for storing data sets, means for monitoring such capacity usage are required. The monitor collects information about the capacity (size) and time (length) during the use of storage services. This data is in turn used to calculate the cost.

3.2.3 Access data

The access to data sets stored in the distributed storage is restricted and access is granted to consumers only after the purchase has been completed. The privilege to access the data is managed by smart contracts, which connect to the storage and provide said privileges once the conditions (payment has been processed, etc.) have been met.

3.3 i3-MARKET backplane use cases

It is expected that several services of the i3-MARKET Backplane use the Distributed Storage for reliably storing metadata regarding the provided service. Although the interaction is not planned for Release 1 of the project, at least the following services will have direct or indirect access to the distributed storage: the non-repudiation service and the Auditable Accounting service.

3.3.1 Non-repudiation service

The distributed storage will be used by the Non-Repudiation protocol for reliably storing proofs of the data exchanges between i3-MARKET consumers and providers. The non-repudiation protocol is developed in T3.3, and a first release has been presented in deliverable D3.5 "Crypto token and data monetisation specification and reference implementation report V1". The proofs that make possible a reliable billing of the service are expected to be stored by consumers and providers in the near future. The provision of either direct access to the distributed storage or indirect access through the Auditable Accounting system is still to be decided, but in any case, only the involved stakeholders in every data exchange should gain access to the non-repudiation proofs.

3.3.2 Auditable accounting

The Auditable Accounting component is responsible for logging relevant information that occurs in the ecosystem for data marketplaces, and in this way, it enhances the trust in the platform. Our solution must ensure the enforcement of the DSA terms, agreed upon by all involved parties, by recording them in an auditable, transparent, and immutable way. Smart Contracts are the key part of the proposed solution for auditable accounting. The auditable accounting component is an abstraction layer to access the Smart Contracts and to allow the integration with the rest of the platform. The auditable accounting component is a service which includes an API to automate the process of logging and auditing interactions between components and record the registries in the Blockchain.

To allow external parties to check that logs have been properly registered in the blockchain, interested parties need to obtain certain data from the distributed ledger as well as some off-chain data provided by the Auditable Accounting module via an API. This off-chain data are essentially Merkle proofs for each individual record (more details about this module including its implementation and architecture are provided in the deliverable D3.3).

In this context, it is important that the off-chain data are provided with high availability. For this reason, the Auditable Accounting module uses the distributed storage component. In this way, high availability and data replication is provided to the relevant off-chain data required to store the registries and verify auditable logs.

4 Technical Requirements

For Data Storage, the following high-level capabilities have been defined:

1. Decentralised Data Storage

Table 2. High-level Capabilities of Decentralised Data Storage

Name	Description	Labels
Embedded Ledger Database	Embedded Ledger Database is shared between operator nodes and keeps a shared state that is guaranteed to be the same at each honest node and is updated according to agreed rules. Children: 1. BFT Consensus 2. Data Security 3. Scalability of Decentralised Storage 4. DID Document status 5. Consent Status Management	V1 Epic
Smart Contracts for Permissioning	Smart contracts are programs that are instantiated from smart contract templates and stored in the distributed ledger along with their state. Children: 1. Smart contracts 2. Smart Contracts Template Storage	V1 Epic

The following table presents the user stories of the decentralised data storage capability.

Table 3. User Stories of Decentralised Data Storage

Name	Description	Due Date	Labels
Smart contracts	As a user I want to instantiate and invoke smart contracts to use the functionality of the system. ——— Siblings: 1. BFT Consensus 2. Data Security 3. Scalability of Decentralised Storage 4. Smart Contracts Template Storage 5. DID Document status 6. Consent Status Management Parents: 1. Smart Contracts for Permissioning	None	V1 User Story
Smart contract template storage	As a developer I want to store smart contract templates for instantiation by users so that I can extend the functionalities provided by the platform. ——— Parents: 1. Smart Contracts for Permissioning Siblings: 1. BFT Consensus 2. Data Security 3. Scalability of Decentralised Storage 4. Smart contracts 5. DID Document status	None	V1 User Story

DID Document Status	As a user I want to register and update the status of my DID document so that I can manage my identity. ______________________ Parents: 1. Embedded Ledger Database Siblings: 1. BFT Consensus 2. Data Security 3. Scalability of Decentralised Storage 4. Smart contracts 5. Consent Status Management 6. Smart Contracts Template Storage	None	User Story V1
Consensus status management	As a user I want to register and update my consent status so that I can control the use of my data. ______________________ Parents: 1. Embedded Ledger Database Siblings: 1. BFT Consensus 2. Data Security 3. Scalability of Decentralised Storage 4. Smart contracts 5. DID Document status	None	
BFT Consensus	As a Data Marketplace I want the Decentralized Data Storage to use Byzantine Fault Tolerant consensus so that I can be sure a malicious party cannot compromise the data. ______________________ Parents: 1. Embedded Ledger Database Siblings: 1. Smart contracts 2. DID Document status 3. Consent Status Management	None	User Story V1

	4. Smart Contracts Template Storage 5. Data Security 6. Scalability of Decentralised Storage		
Data Security	As a stakeholder I want to have guarantees about ledger data integrity, availability, confidentiality so that I can rely on the services provided by the system. Parents: 1. Embedded Ledger Database 2. Smart Contracts for Permissioning Siblings: 1. BFT Consensus 2. Smart contracts 3. DID Document status 4. Consent Status Management 5. Smart Contracts Template Storage 6. Scalability of Decentralised Storage	None	
Scalability of decentralized storage	As an operator of ledger databases, I want to be able to scale the storage to meet space and transaction rate demands in order to be able to run the ledger. Parents: 1. Embedded Ledger Database Siblings: 1. BFT Consensus 2. Data Security 3. Smart contracts 4. DID Document status 5. Consent Status Management 6. Smart Contracts Template Storage	None	

2. Distributed Data Storage

Table 4. High-level Capabilities of Distributed Data Storage

Name	Description	Labels
Main Database with Consensus, Sharding and Permissioning	The main database supporting consensus, sharding and permissioning. --- Children: 1. Semantic Data Availability 2. Semantic Data Updates 3. Data Offering Registration 4. Verifiable Claim Storage 5. Offer Query Registration 6. Metadata Update 7. SLA Template Management 8. Scalability of Distributed Data Storage 9. Data Storage: Semantic Data Storage 10. Metadata Storage	V1 Epic
Synchronization	The distributed storage database must support data synchronization between nodes. --- Children: 1. Semantic Data Availability 2. Semantic Data Updates 3. Data Offering Registration 4. Verifiable Claim Storage 5. Offer Query Registration 6. Metadata Update 7. SLA Template Management 8. Scalability of Distributed Data Storage	V1 Epic
Semantic Database	Semantic database is a distributed database supporting the storage of semantic data and processing semantic queries. --- Children: 1. Semantic Data Availability 2. Semantic Data Updates 3. Data Offering Registration 4. Verifiable Claim Storage 5. Offer Query Registration 6. Metadata Update	

	7. SLA Template Management 8. Data Storage: Semantic Data Storage 9. Metadata Storage	
API for External Access	The API for External Access provides an interface for using the distributed storage to access and store metadata, verifiable claims, semantic data, semantic queries etc. Children: 1. Semantic Data Availability 2. Semantic Data Updates 3. Data Offering Registration 4. Verifiable Claim Storage 5. Offer Query Registration 6. Metadata Update 7. SLA Template Management 8. Data Storage: Semantic Data Storage 9. Metadata Storage	

The following table lists the user stories of the distributed data storage capability.

Table 5. User Stories of the Distributed Data Storage

Name	Description	Due Date	Labels
Semantic data availability	As a Data Marketplace I want to access semantic data so that I can process semantic queries. Parents: 1. Semantic Database 2. API For External Access 3. Synchronization 4. Main Database with Consensus, Sharding, Permissioning	None	V1 User Story

Semantic data updates	As a Data Marketplace I want to access semantic data so that I can process semantic queries. Parents: 1. Semantic Database 2. API For External Access 3. Synchronization 4. Main Database with Consensus, Sharding, Permissioning	None	
Data Offering registration	As a Data Provider I want to register my data offering at a Data Marketplace so that I can sell my data discovered via the offering. Parents: 1. Main Database with Consensus, Sharding, Permissioning 2. API For External Access 3. Semantic Database 4. Synchronization	None	User Story V1
Verifiable claim storage	As a user I want to use the distributed storage to store verifiable claims (including consents). Parents: 1. API For External Access 2. Semantic Database 3. Synchronization 4. Main Database with Consensus, Sharding, Permissioning	None	User Story V1
Offer query registration	As a data consumer I want to register offer query so that the system can find the data I need. Parents: 1. API For External Access 2. Semantic Database 3. Synchronization 4. Main Database with Consensus, Sharding, Permissioning	None	

Metadata update	As a data provider I want to update the metadata of my data so that the metadata is up-to-date. --- Parents: 1. API For External Access 2. Semantic Database 3. Synchronization 4. Main Database with Consensus, Sharding, Permissioning	None	
SLA template management	As a stakeholder I want to store SLA templates so that other users could fetch the templates. --- Parents: 1. API For External Access 2. Semantic Database 3. Synchronization 4. Main Database with Consensus, Sharding, Permissioning	None	User Story V1
Scalability of distributed data storage	As a distributed data storage node operator I want to use sharding so that I can scale the storage. --- Parents: 1. Synchronization 2. Main Database with Consensus, Sharding, Permissioning		User Story V1
Metadata storage	As a data provider, I want to store metadata of the dataset. --- Parents: 1. Main Database with Consensus, Sharding, Permissioning 2. Semantic Database 3. API For External Access		User Story V1

4.1 Technical Requirements - R1

The following table list all the epics from the backlog of Release1.

Table 6. Epics of Release 1

Name	Description	Due Date	Labels
AutomotivePilot Share Data	Use case grouping the different actions needed for a Data Owner to share generated data from a Data Provider. --- Children: 1. Work with data/metadata under CIDM specification 2. Push Data 3. Accept Data Requests 4. Automotive-Pilot Work with data/metadata under CIDM specification 5. Automotive-Pilot Get Metadata 6. Automotive-Pilot Accept Data Requests	None	
Data Storage: Embedded Ledger Database	Embedded Ledger Database is shared between operator nodes and keeps a shared state that is guaranteed to be the same at each honest node and is updated according to agreed rules. --- Children: 1. BFT Consensus 2. Data Security 3. Scalability of Decentralised Storage 4. DID Document status 5. Consent Status Management		Epic V1
Data Storage: Main Database with Consensus, Sharding, Permissioning	The main database supporting consensus, sharding and permissioning. --- Children: 1. Semantic Data Availability 2. Semantic Data Updates 3. Data Offering Registration 4. Verifiable Claim Storage 5. Offer Query Registration 6. Metadata Update 7. SLA Template Management		Epic V1

	8. Scalability of Distributed Data Storage		
Data Storage: API for External Access	The API for External Access provides an interface for using the distributed storage to access and store metadata, verifiable claims, semantic data, semantic queries etc. Children: 1. Semantic Data Availability 2. Semantic Data Updates 3. Data Offering Registration 4. Verifiable Claim Storage 5. Offer Query Registration 6. Metadata Update 7. SLA Template Management		Epic V1

The following table list all the features from the backlog of Release1.

Table 7. Features of Release 1

Name	Description	Due Date	Labels
AutomotivePilot Push Data	Describe and link with related user stories using connection button. It should be attached a UML diagram. Children: 1. Push data and share across i3-MARKET 2. Automotive-Pilot Push metadata and share across i3-MARKET Parents: 1. Share Data 2. Automotive-Pilot Share Data	None	
Data Storage: DID Document	As a user I want to register and update the status of my DID document so that I can manage my identity. Parents: 1. Embedded Ledger Database		V1 User Story

	Siblings: 1. BFT Consensus 2. Data Security 3. Scalability of Decentralised Storage 4. Smart contracts 5. Consent Status Management 6. Smart Contracts Template Storage		
Data Storage: Data Offering Registration	As a Data Provider I want to register my data offering at a Data Marketplace so that I can sell my data discovered via the offering. ——— Parents: 1. Main Database with Consensus, Sharding, Permissioning 2. API For External Access 3. Semantic Database 4. Synchronization		V1 User Story
Data Storage: Data Security	As a stakeholder I want to have guarantees about ledger data integrity, availability, confidentiality so that I can rely on the services provided by the system. ——— Parents: 1. Embedded Ledger Database 2. Smart Contracts for Permissioning Siblings: 1. BFT Consensus 2. Smart contracts 3. DID Document status 4. Consent Status Management 5. Smart Contracts Template Storage 6. Scalability of Decentralised Storage		V1 User Story
Data Storage: BFT Consensus	As a Data Marketplace I want the Decentralized Data Storage to use Byzantine Fault Tolerant consensus so that I can be sure a malicious party cannot compromise the data. ——— Parents: 1. Embedded Ledger Database		

	Siblings: 1. Smart contracts 2. DID Document status 3. Consent Status Management 4. Smart Contracts Template Storage 5. Data Security 6. Scalability of Decentralised Storage		
Data Storage: Verifiable Claim Storage	As a user I want to use the distributed storage to store verifiable claims (including consents). Parents: 1. API For External Access 2. Semantic Database 3. Synchronization 4. Main Database with Consensus, Sharding, Permissioning		V1 User Story
Data Storage: Metadata Storage	As a data provider, I want to store metadata of the dataset. Parents: 1. Main Database with Consensus, Sharding, Permissioning 2. Semantic Database 3. API For External Access		V1 User Story

The following table list all the tasks from the backlog of Release1.

Table 8. Tasks of Release 1

Name	Description	Due Date	Labels
Identify options to store Verifiable Claims		None	TASK R1

AutomotivePilot Push metadata	As a Data Provider I want to push metadata to i3-MARKET storage so that I can let other marketplaces know the data I am willing to share. Parents: 1. Push Data 2. Automotive-Pilot Get Metadata		V1 Data Provider User Story
Finalise Distributed data storage technology decision	In order to proceed with data storage design, consensus must be achieved in choosing the most suitable solution for distributed storage. In R1, a single technology that suits most needs shall be chosen. In later version, it shall be decided whether the chosen technology suits all needs and if there is a need to switch or add another solution to the storage layer.		TASK R1
Data Storage layer architecture	Produce an architectural view of the data storage layer, once the data flows, requirements and relevant technologies are clear.		TASK R1
Deploy Blockchain	Deploy and orchestrate BESU network in the allocated Virtual Machines.		TASK R1
Deploy Distributed Data storage	Deploy and maintain distributed storage in the allocated Virtual Machines.		TASK R1
Finalise blockchain technology decision	Perform a high-level evaluation on blockchain technologies that provide the required functionality for building blocks.		TASK R1
Analyse distributed storage solutions	An analysis and evaluation must be carried out in order to choose the suitable solution to store metadata, offerings, queries etc.		
Define APIs	Define the APIs for - SEEDs - SSI & IAM - Smart Contracts - Data Monetisation - Auditable Accounting - Cloud Wallet		TASK R1

5 Sequence Diagrams

5.1 Federated Query Engine Index Management

The sequence diagram in Figure 4 shows the interaction of the distributed storage on the SEED regarding the federated query engine index management. Each function – insert, update, delete and query – has been depicted on a single sequence diagram, as there is no relevant complexity to be shown for each interaction.

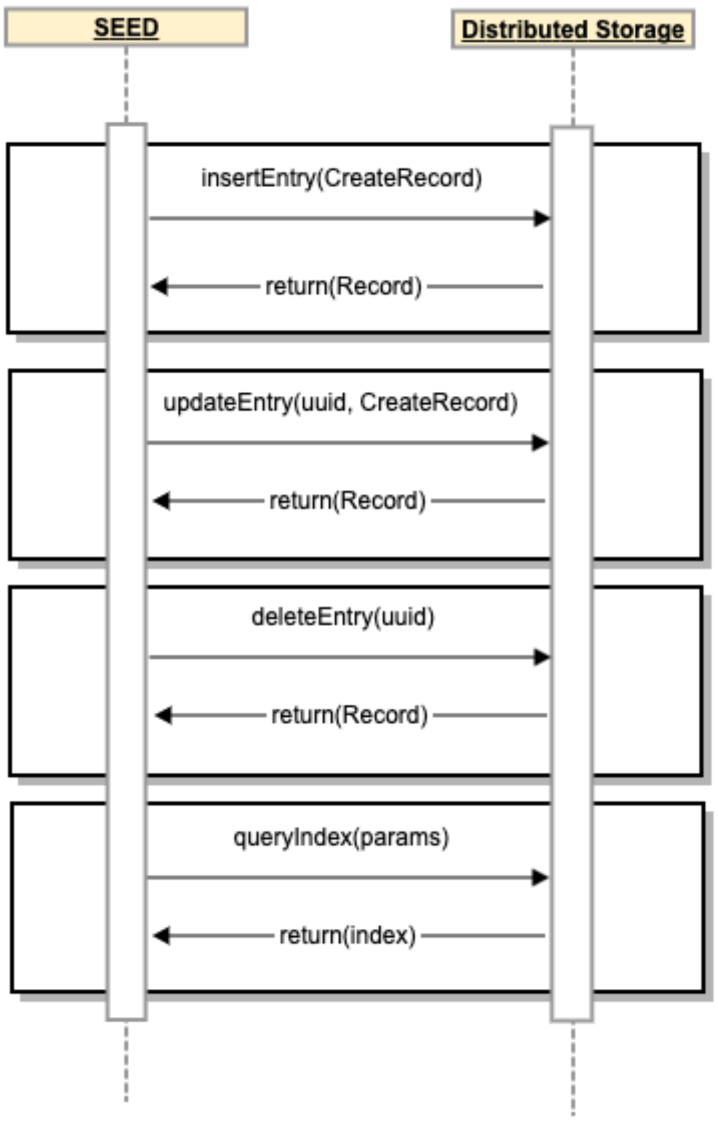
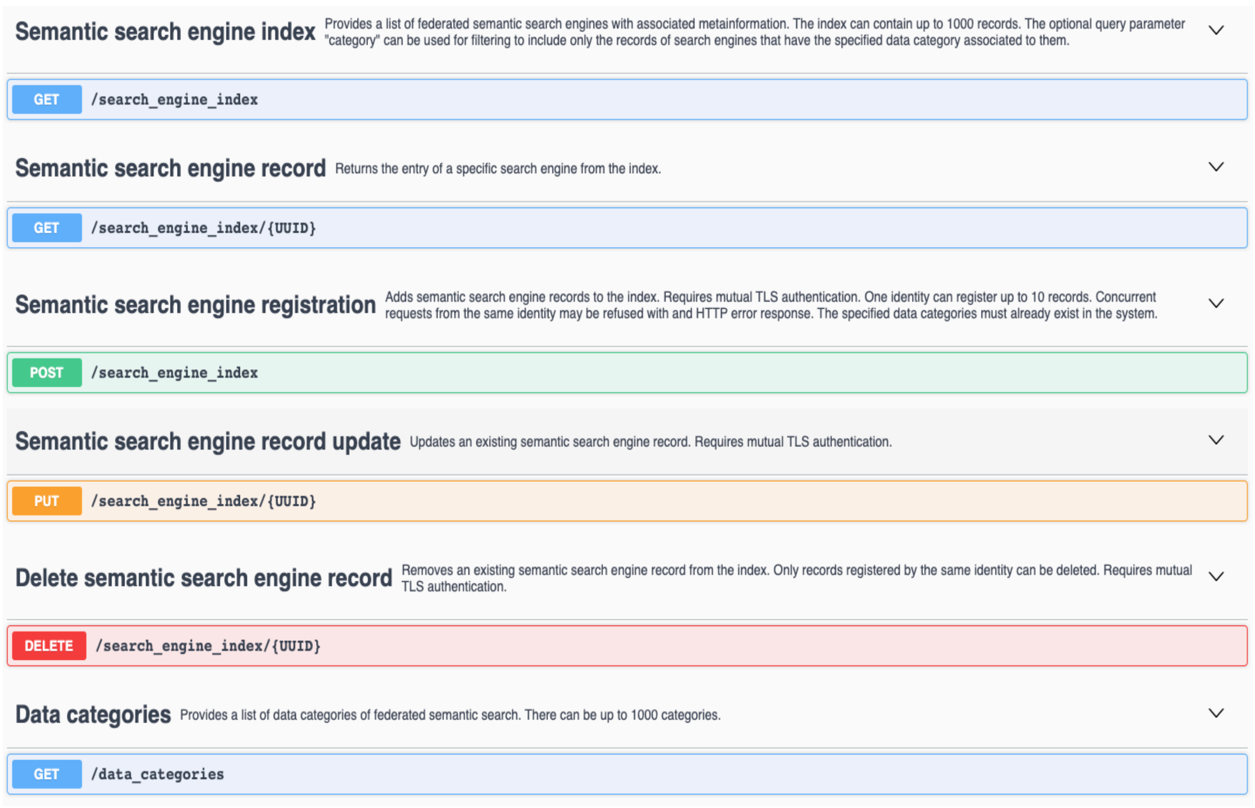


Figure 4. Federated Query Engine Index Management

6 Interfaces Description

The description of the interfaces that have been implemented for the distributed storage system can be seen below, the services implemented for the management of the federated query engine index are shown. The extract is taken from the Swagger UI served by the distributed storage service. An up-to-date online version of the interface description is provided by the distributed storage system and can be accessed at the nodes where the service is deployed[1]. The interface description is also available in OpenAPI 3 form to facilitate integration and the generation of client libraries.

The API provided by the decentralised storage comes out-of-box with the solution. The service can be accessed via JSON-RPC protocol offered by Hyperledger BESU. Please refer to *BESU Documentation*[2] for the details of the API provided by Hyperledger BESU and client libraries..



Semantic search engine index Provides a list of federated semantic search engines with associated metainformation. The index can contain up to 1000 records. The optional query parameter "category" can be used for filtering to include only the records of search engines that have the specified data category associated to them.

GET `/search_engine_index`

Semantic search engine record Returns the entry of a specific search engine from the index.

GET `/search_engine_index/{UUID}`

Semantic search engine registration Adds semantic search engine records to the index. Requires mutual TLS authentication. One identity can register up to 10 records. Concurrent requests from the same identity may be refused with and HTTP error response. The specified data categories must already exist in the system.

POST `/search_engine_index`

Semantic search engine record update Updates an existing semantic search engine record. Requires mutual TLS authentication.

PUT `/search_engine_index/{UUID}`

Delete semantic search engine record Removes an existing semantic search engine record from the index. Only records registered by the same identity can be deleted. Requires mutual TLS authentication.

DELETE `/search_engine_index/{UUID}`

Data categories Provides a list of data categories of federated semantic search. There can be up to 1000 categories.

GET `/data_categories`

Figure 5. Distributed Storage API

1 Documentation available at i3-MARKET node 1: http://95.211.3.244:7500/docs/

2 Hyperledger BESU documentation, https://besu.hyperledger.org/en/stable/

7 Selected Technologies

The partners within Task 4.1 have taken the following decisions for the choice of selected technologies in order to satisfy the high-level capabilities of T4.1 in Release 1:

- Hyperledger BESU to satisfy decentralised storage
- CockroachDB[3] to satisfy distributed storage

Hyperledger BESU is an open-source Ethereum client. The decision to select Hyperledger BESU to satisfy the needs for decentralised storage has been made based on the following assumptions:

- Self-sovereign identity and access management (T3.1) has decided to base the reference implementation on Veramo, which specifically requires Ethereum blockchain.
- Auditable accounting and data monetisation rely on Ethereum mainnet.

CockroachDB is a relational database management system. CockroachDB has been chosen as the distributed storage due to previous experience of the technology by partners in T4.1. Moreover, it is highly scalable, designed to deliver fast access and resilient to network outages. The only shortcoming of the chosen technology is the lack of features guaranteeing data integrity in case of the presence of malicious (e.g., because of honest mistakes or sophisticated external attacks) users. It has been decided that in R1 it is not an issue, and a workaround shall be designed later. A potential workaround for protecting data integrity in CockroachDB is to anchor authorized data hashes in the decentralised storage that is resistant to Byzantine faults.

Although Hyperledger BESU has been identified as the most suitable solution in R1, re-evaluation of the technology needs shall be conducted during the project and the final version could be based on a different distributed ledger implementation. The same applies to the distributed storage solution, as currently, not all technical needs of building blocks are clear and a consensus regarding the overall architecture needs to be achieved by all related parties.

The distributed search engine index is implemented as a Haskell Servant[4] application and it uses the CockroachDB database cluster as the storage backend. For client authentication, mutual TLS authentication is used.

3 CockroachDB, https://www.cockroachlabs.com/product/

4 servant – A Type-Level Web DSL, https://docs.servant.dev/en/stable/

8 Technical contributions of i3-MARKET project

Data Storage subsystem has a vital role in i3-MARKET network, providing both structured database and ledger storage solutions, hence supporting several use cases. There are several contributions that the Data Storage subsystem brings to the i3-MARKET project.

The main technical contribution of i3-MARKET project for decentralised and distributed storage is combining existing state-of-the-art storage technologies in a novel way and enriching these components with custom application-specific logic and interfaces to support the diverse goals of the i3-MARKET framework that an off-the-shelf solution would not be able to cover. This can demonstrate how technologies with different technical characteristics can be used synergistically in a distributed system to leverage the relative strengths of each storage component in order to achieve new qualities in the whole system.

In particular, the decentralised storage component Hyperledger Besu already provides storage with strong security guarantees (integrity, availability), but due to its architecture, it is relatively inefficient and expensive for storing large or mutable data sets. Also, the query processing capabilities of Hyperledger Besu are quite limited compared to a relational database engine. As the i3-MARKET framework contains elements of a conventional information system where such features are desirable, it is reasonable to integrate a database management system that provides efficient storage for large and mutable data sets as well as rich query processing facilities. An SQL database management system comes with mature software development tools and libraries while also being already familiar to software developers. However, even the state-of-the-art SQL database management systems, such as CockroachDB, that support redundancy and distributed deployments, lack some of the integrity guarantees of blockchain-based distributed ledgers. The integrity of these databases relies on the node operators being trusted. In case of an i3-MARKET like setting where participants may have conflicting financial interests and there is a possibility of collusion to achieve gains, it is important to minimise the amount of trusted components ("trusted component" means that users must rely on the component without being able to actually verify the correct operation of the component). By combining these technologies, the whole system can exploit a storage system with large capacity, powerful query processing and good integrity properties without relying on trusted components.

The auditable accounting component is a good example how the storage system can be utilized – small pieces of data (hashes) are anchored in the distributed ledger and larger proofs are stored in CockroachDB. Also, the storage supports the coordination between different instances of semantic search engines where all parties can verify that the database cluster nodes operated correctly and, for example, did not hide some records from some parties.

With the novel storage infrastructure in place, we expect future use cases to utilize more of the potential and advanced features that the storage subsystem provides, hence minimising the development effort of new use cases. This may also include integrating additional storage technologies (i.e., IPFS, NoSQL, graph database) to the existing framework.

9 Conclusion / Future Work

The architectural view of the data storage system has been defined and the core technologies to support the features of decentralised and distributed storage have been chosen and are Hyperledger BESU and CockroachDB, respectively.

Considerable effort was put on the solution to support data synchronization between the marketplaces connected to the i3-MARKET network. Several options were presented, however, in the end the decision landed on federated query engine index management solution. The distributed storage shall maintain an index that is consulted by each i3-MARKET node in order to build federated queries across all marketplaces. The functionality of the federated query index will likely be expanded based on emerging needs and innovative features built into the i3-MARKET platform.

The next step is to design a solution along with the logical workflows and interfaces, for storing data sets on sale to the distributed storage. This solution requires a common understanding between involved components and respective engineering partners.

Moreover, the capacity usage requires means for monitoring, in order to monetize the service.

The authentication and security of machine-to-machine connections is based on public infrastructure. Proper management processes for the i3-MARKET PKI have to be designed and implemented.