Data Platform for the Connection of Cognitive Ports

Title:	Document Version:
D2.4 Platform Architecture and Specifications	1.2

Project Number:	Project Acronym:	Project Title:
H2020-871493	DataPorts	A Data Platform for the Cognitive Ports of the Future

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type*-Security*:
M24 (December 2021)	M24 (December 2021)	R-PU

*Type: P: Prototype; R: Report; D: Demonstrator; O: Other; ORDP: Open Research Data Pilot; E: Ethics.

**Security Class: PU: Public; PP: Restricted to other program participants (including the Commission); RE: Restricted to a group defined by the consortium (including the Commission); CO: Confidential, only for members of the consortium (including the Commission).

Responsible:	Organization:	Contributing WP:
Amirshayan Ahmadian	UniKoblenz	WP2

Authors (organization):	
Amirshayan Ahmadian (UniKoblenz)	Daniel García (EVR)
Jan Jürjens (UniKoblenz, FHG)	Eliseo Venegas (EVR)
Francisco Valverde (ITI)	Tristan Kley (UDE)
Santiago Cáceres (ITI)	Xhulja Shahini (UDE)
Miguel Bravo (ITI)	Anastasios Nikolakopoulos (ICCS)
Achilleas Marinakis (ICCS)	Inna Skarbovsky (IBM)
Andreu Belsa (UPV)	Konstantinos Votis (CERTH)
Matilde Julian (UPV)	Sofia Terzi (CERTH)
Hector Iturria (PRO)	Vasilis Siopidis (CERTH)
Jose Antonio Clemente (PRO)	Evi Manganopoulou (CERTH)
Andreas Metzger (UDE)	Alexandros Zerzelidis (CERTH)
Fabiana Fournier (IBM)	

Abstract:

This deliverable provides technical and functional specifications of all the components composing the industrial data platform, together with the relationships among them and the final architecture. It also includes the blockchain design guidelines, criteria and solutions envisioned in the scope of T2.4.

Keywords:

Data Platform Architecture, Cognitive Ports Components, Data Flows in Cognitive Ports, Data Platform Functionalities, Blockchain technology in Seaports

Revision History

Revision	Date	Description	Author (organization)
V0.1	22.07.2020	Added outline, updated outline	Shayan Ahmadian
V0.2	20.08.2020	Added the UKL contribution	Shayan Ahmadian
V0.3	26.08.2020	Added the ICCS, ITI, UDE contribution	Shayan Ahmadian and partners
V0.4	28.08.2020	Added the PRO contribution	Shayan Ahmadian and partners
V0.5	31.08.2020	Added EVR contribution	Shayan Ahmadian and partners
V0.6	18.12.2020	Merged, improved the document after second round of establishing the document.	Shayan Ahmadian and partners
V0.7	16.06.2021	Merged, improved the document after the third round of establishing the document	Shayan Ahmadian and Partners
V0.7	13.07.2021	Merged, improved the document, after the contributions on functionalities	Shayan Ahmadian and Partners
V1.0	24.11.2021	Merged contributions, improved the document (Content and format)	Shayan Ahmadian and Partners
V1.1	14.12.2021	Pre-Final version	Santiago Cáceres (ITI)
V1.2	20.12.2021	Modifications based on peer review process	Santiago Cáceres (ITI), Fabiana Fournier (IBM), Eliseo Venegas (EVR)

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement № 871493.

More information available at https://DataPorts-project.eu

Copyright Statement

INDEX

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 DATAPORTS PROJECT OVERVIEW

DataPorts is a project funded by the European Commission as part of the H2020 Big Data Value PPP programme and coordinated by the ITI - Technological Institute of Informatics. DataPorts rely on the participation of 13 partners from five different nationalities. The project involves the design and implementation of a data platform, its deployment in two relevant European seaports connecting to their existing digital infrastructures and addressing specific local constraints. Furthermore, a global use case involving these two ports and other actors and targeting inter-port objectives, and all the actions to foster the adoption of the platform at European level.

Hundreds of different European seaports collaborate with each other, exchanging different digital data from several data sources. However, to achieve efficient collaboration and benefit from AI-based technology, a new integrating environment is needed. To this end, DataPorts project is designing and implementing an Industrial Data Platform.

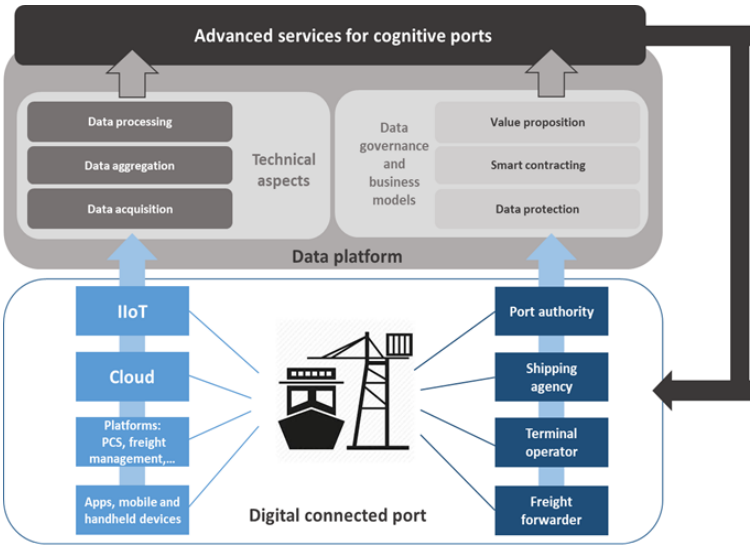

Figure 1 - DataPorts project overview

The DataPorts Platform aim is to connect to the different digital infrastructures currently existing in digital seaports, enabling the interconnection of a wide variety of systems into a tightly integrated ecosystem. In addition, to set the policies for a trusted and reliable data sharing and trading based on data owners' rules and offering a clear value proposition. Finally, to leverage on the data collected to provide advanced Data Analytic services based on which the different actors in the port value chain could develop novel AI and cognitive applications.

DataPorts will allow establish a future Data Space unique for all maritime ports of Europe and contribute to the EC global objective of creating a Common European Data Space.

1.2 DELIVERABLE PURPOSE AND SCOPE

Specifically, the DOA states the following regarding this Deliverable:

This deliverable will provide technical and functional specifications of all the components composing the industrial data platform, together with the relationships among them and the final architecture. It will also include the blockchain design guidelines, criteria and solutions envisioned in the scope of T2.4.

The main purpose of this document is to specify the architecture of the DataPorts' data platform. This document serves as a reference for implementing various components of the platform. Furthermore, the full list of the functionalities of the data platform is listed and the corresponding requirements and components are specified.

1.3 DELIVERABLE CONTEXT

Its relationship to other documents is as follows:

Primary Preceding documents:

- DataPorts' Grant Agreement [1]: Providing the foundation for the actual research and technological content of DataPorts. Importantly, the Description of Action includes a description of the overall project work plan.
- DataPorts' Deliverable 2.1 [2]: This document is used as the source of DataPorts platform's requirement
- DataPorts' Deliverable 5.1 [3]: Integration, software quality assurance and deployment plan

Primary Dependant documents:

- Several documents including all the component description deliverables finalized in the course of WP3 are depending on this document. Furthermore, this document considers the platform requirements introduced in Deliverable D2.1 (Industrial Data Platforms and Seaport Community Requirements and Challenges).

1.4 DOCUMENT STRUCTURE

This deliverable is broken down in the following sections:

- **Section 1** includes the introduction of the document which entails the description of the scope, purpose and the structure of the document.
- **Section 2** describes shortly the methodology used to specify the architecture design. Furthermore, a background on International Data Spaces Association is provided.
- **Section 3** provides a summary of the DataPorts' requirements.
- **Section 4** provides a list of the DataPorts' functionalities.
- **Section 5** describes two scenarios that (conceptually) could be used for data sharing using DataPorts data platform.
- **Section 6** introduces the architectural components of the data platform including all relevant specifications.
- **Section 7** shortly describes the mechanism that will be used to evaluate the architecture and the technical decisions, in the context of the future deliverables and tasks.
- **Section 8** Concludes the document.

Annexes

- **Annex A:** References
- **Annex B:** Acronyms

2 METHODOLOGY AND PRELIMINARIES

In this section we shortly introduce our methodology to design the architecture of the platform. We further provide a background on the International Data Spaces (IDS) Reference Architecture Model (RAM). This reference architecture model introduces a methodology to share data in a trusted environment.

2.1 THE PROCESS OF SPECIFIYING THE DATAPORTS PLATFORM ARCHITECTURE

The architecture design which is introduced in the Description of Action (DoA), is used as a basis to define the architecture design of the DataPorts platform. However, the initial architecture design is permanently modified, refined within several iterations, and discussions to obtain the current architecture design.

Various task leaders have introduced their contributions to the architecture design in several meetings, and workshops organized by the University of Koblenz Landau (the leader of the relevant task to design the architecture and submit this deliverable). The specification of the architecture has been an iterative process.

Several reference architecture models are mentioned in the project proposal [1], prescribing that the resulting DataPorts' architecture must follow these reference models. Our architecture model is designed in way that it follows several concepts of the IDS RAM. Several concepts from the IDS reference architecture model are like components of the DataPorts platform. IDS RAM itself is a standard reference architecture model that follows other popular reference architecture models such as Industrial Internet Reference Architecture (IIRA), and RAMI 3.0 (and some concepts from RAMI 4.0), as well as the IoT Reference Architecture (HLA). Concerning this fact, the design of our platform follows the concepts of the mentioned reference architecture models.

2.2 INTERNATIONAL DATA SPACES REFERENCE ARCHITECTURE MODEL

In a nutshell, the International Data Space (IDS) is a virtual environment (see Figure 2), leveraging existing standards and technologies, as well as governance models well-accepted in the data economy, to facilitate secure and standardized data exchange and data linkage in a trusted ecosystem [4].

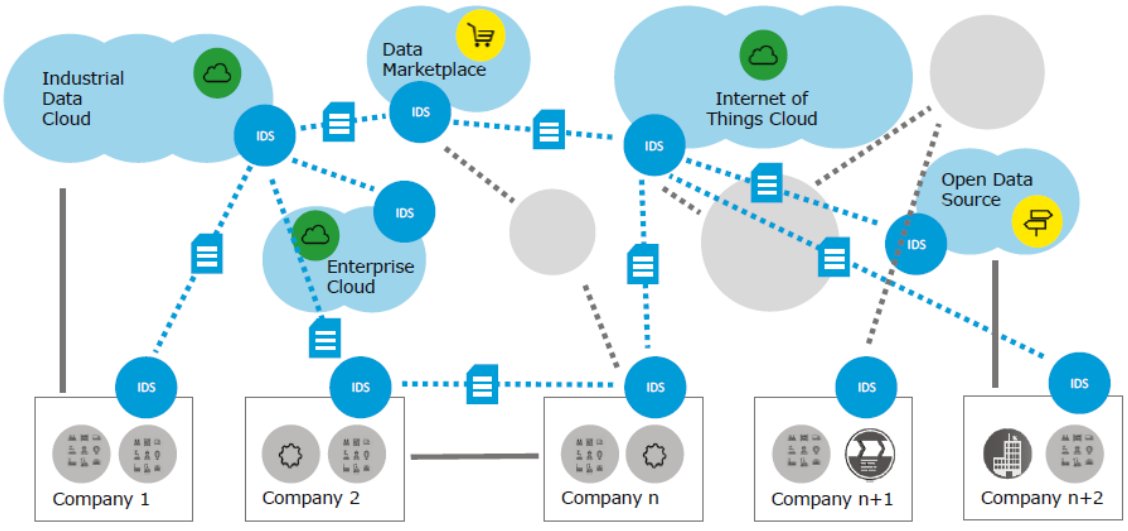

Figure 2 - IDS provides an ecosystem where various cloud platforms are connected [4]

2.2.1 IDS strategic requirements

The main strategic requirements that are aimed by the IDS are:

- **Trust:** Each participant is evaluated and certified before being granted access to the trusted business ecosystem.

- **Security and data sovereignty:** All components of the IDS rely in state-of-the-art security measures.
- **Ecosystem of data:** The architecture design of the IDS does not require a central data storage capability. Instead, it pursues the idea of decentralized data storage, which means that data physically remains with the respective data owner until it is transferred to a trusted party.
- **Standardized interoperability:** The International Data Space Connector, being a central component of the architecture, is implemented in different variants and can be adapted for different purposes.
- **Value adding apps:** The apps enable service on top of the data exchange processes. This includes services for data processing, data format alignment, and data exchange protocols. Furthermore, data analytics services can be provided by remote execution of algorithms.
- **Data market:** The International Data Space enables the creation of novel, data-driven services that make use of data apps. It also fosters new business models for these services by providing clearing mechanisms and billing functions, and by creating domain-specific broker solutions and marketplaces. In addition, the International Data Spaces provides templates and other methodological support for participants to use when specifying usage restriction information and requesting legal information.
- **Re-use of existing technologies:** The IDS initiative aims to use existing technologies (from the open-source domain), and standards (e.g., semantic standards of the W3C).
- **Contribution to standardization:** The IDS initiative supports the idea of standardized architecture stacks.

By proposing an architecture for secure data exchange and trusted data sharing, the International Data Spaces contributes to the design of enterprise architectures in commercial and industrial digitization scenarios. The architecture is designed with the objective to overcome the differences between top-down approaches and bottom-up approaches.

2.2.2 Layers of the reference architecture model

The IDS reference architecture model consists of five layers [4],namely business layer, functional layer, process layer, information layer, and system layer.

2.2.2.1 Business layer

The Business Layer of the Reference Architecture Model defines and categorizes the different roles the participants in the International Data Spaces may assume. Furthermore, it specifies basic patterns of interaction taking place between these roles. It thereby contributes to the development of innovative business models and digital, data-driven services to be used by the participants in the International Data Spaces. While the Business Layer provides an abstract description of the roles in the International Data Spaces, it can be considered a blueprint for the other, more technical layers. The Business Layer can therefore be used to verify the technical architecture of the International Data Spaces. In this sense, the Business Layer specifies the requirements to be addressed by the Functional Layer.

2.2.2.2 Functional layer

The Functional Layer defines – irrespective of existing technologies and applications – the functional requirements of the International Data Spaces, and the features to be implemented (Figure 3).

Figure 3 - Functional architecture of the IDS-RAM [4]

2.2.2.3 Process layer

The Process Layer specifies the interactions taking place between the different components of the International Data Spaces. It thereby provides a dynamic view of the Reference Architecture Model. In the following, three major processes and their sub processes are described:

- On boarding, i.e., what to do to be granted access to the International Data Spaces as a Data Provider or Data User;
- Exchanging data, i.e., searching for a suitable Data Provider and invoking the actual data operation; and
- Publishing and using Data Apps, i.e., interacting with the IDS as an App Provider and user of a Data App, respectively.

2.2.2.4 Information layer

The Information Layer specifies the Information Model, the domain-agnostic, common language of the International Data Spaces. The Information Model is an essential agreement shared by the participants and components of the IDS, facilitating compatibility and interoperability. The primary purpose of this formal model is to enable (semi-)automated exchange of digital resources within a trusted ecosystem of distributed parties, while preserving data sovereignty of Data Owners. The Information Model therefore supports the description, publication and identification of data products and reusable data processing software (both referred to hereinafter as “Digital Resources”, or simply “Resources”). Once the relevant Resources are identified, they can be exchanged and consumed via semantically annotated and easily discoverable services. Apart from those core commodities, the Information Model describes essential constituents of the International Data Spaces, its participants, its infrastructure components, and its processes.

The Information Model has been specified at three levels of formalization (namely conceptual representation, declarative representation and programmatic representation). Each level corresponds to a digital representation, ranging from this high-level, conceptual document down to the level of operational code. Every representation depicts the complete Information Model in its way. Among the different representations, the Declarative Representation (IDS Ontology) is the only normative specification of the Information Model. As such, it is accompanied by a set of auxiliary resources (e.g., guidance documents, reference examples, validation tools, and editing tools intended to support a competent, appropriate, and consistent usage of the IDS Ontology).

The Declarative Representation (IDS Ontology) provides a normative view of the Information Model of the International Data Spaces. It has been developed along the analysis, findings, and requirements of the

Conceptual Representation. Based on a stack of W3C Semantic Web technology standards and standard modelling vocabularies (DCAT7, ODRL8, etc.), it provides a formal, machine-interpretable specification of concepts envisaged by the Conceptual Representation. Furthermore, it details and formally defines entities of the International Data Spaces in order to be able to share, search for, and reason upon the structured metadata describing these entities. As such, it comprises a complete referential model allowing the derivation of several Programmatic Representations. The IDS Ontology is typically used and instantiated by knowledge engineers, ontology experts, or information architects. It defines a minimal, domain-agnostic "core model" and relies on third-party standard and custom vocabularies in order to express domain specific facts. According to the common practice, existing domain vocabularies and standards are reused where possible, fostering acceptance and interoperability.

2.2.2.5 System layer

On the System Layer, the roles specified on the Business Layer are mapped onto a concrete data and service architecture in order to meet the requirements specified on the Functional Layer, resulting in what can be considered the technical core of the International Data Spaces. From the requirements identified on the Functional Layer, three major technical components result:

- The Broker (Section 2.2.3.1),
- The Connector (Section 2.2.3.2), and
- The App Store (Section 2.2.3.3).

How these components interact with each other is depicted in Figure 4.

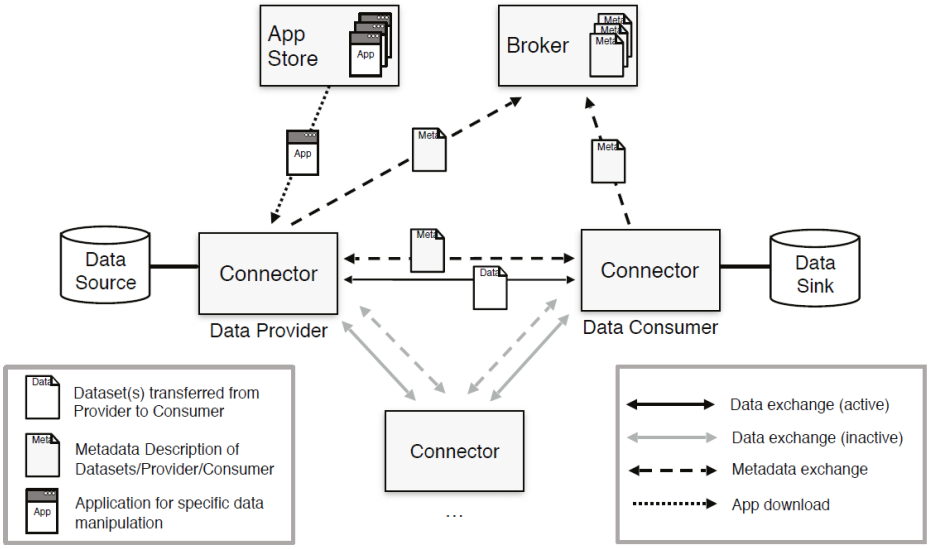

Figure 4 - Interaction of technical components [4]

2.2.3 IDS-RAM major technical components

The two Key components of the IDS reference architecture model are the broker and connector, which in fact realize the decentralized architecture and ensure scalability.

2.2.3.1 IDS Broker

The Broker Service Provider is an intermediary that stores and manages information about the data sources available in the International Data Spaces. As the role of the Broker Service Provider is central but non-exclusive, multiple Broker Service Providers may be around at the same time (e.g., for different application domains).

The activities of the Broker Service Provider mainly focus on receiving and providing metadata. The Broker Service Provider must provide an interface for Data Providers to send their metadata. The metadata should be stored in an internal repository for being queried by Data Consumers in a structured manner. While the core of the metadata model must be specified by the International Data Spaces, a Broker Service Provider may extend the metadata model to manage additional metadata elements.

Using an integrated index service, the Broker manages the data sources available in the International Data Spaces and supports publication and maintenance of associated metadata. Furthermore, the Broker Index Service supports the search for data resources. Both the App Store and the Broker are based on the Connector architecture (which is described in detail in the following paragraphs) in order to support secure and trusted data exchange with these services.

2.2.3.2 The Connector architecture

Standardized data exchange between participants is the fundamental aspect of the International Data Spaces. The IDS Connector is the main technical component for this purpose. The IDS reference architecture model offers the internal structure of a connector. A concrete installation of a connector may differ from this structure, as existing components can be modified, and optional components may be added.

There may be different types of implementations of the Connector, based on different technologies and depending on what specific functionality is required regarding the purpose of the Connector. Two fundamental variants are the Base Connector and the Trusted Connector as they differ in the capabilities regarding security and data sovereignty. Instances of the Trusted Connector enable remote integrity verification, so the integrity of the deployed software stack can be guaranteed before granting access to data. The Trusted Connector guarantees a controlled execution environment for data services and supports the creation of trusted relationships.

Connectors can be further distinguished into External Connectors and Internal Connectors:

- **An External Connector** executes the exchange of data between participants of the International Data Spaces. The International Data Spaces network is constituted by the total of its External Connectors. Each External Connector provides data via the Data Endpoints it exposes. Applying this principle, there is no need for a central instance for data storage. An External Connector is typically operated behind a firewall in a specially secured network segment of a participant (so-called “Demilitarized Zone”, DMZ). From a DMZ, direct access to internal systems is not possible. It should be possible to reach an External Connector using the standard Internet Protocol (IP), and to operate it in any appropriate environment. A participant may operate multiple External Connectors (e.g., to meet load balancing or data partitioning requirements). External Connectors can be operated on-premises or in a cloud environment.
- **An Internal Connector** is typically operated in an internal company network (i.e., a network which is not accessible from outside). Implementations of Internal Connectors and External Connectors may be identical, as only the purpose and configuration differ. The main task of an Internal Connector is to facilitate access to internal data sources in order to provide data to External Connectors.

The Connector Architecture uses application container management technology to ensure an isolated and secure environment for individual data services. A data service matches a system which offers an API to store, access or process data. To ensure privacy of sensitive data, data processing should take place as close to the data source as possible. Any data pre-processing (e.g., filtering, anonymization, or analysis) should be performed by Internal Connectors. Only data intended for being made available to other participants should be made visible through External Connectors.

Data Apps are data services encapsulating data processing and/or data transformation functionality bundled as container images for simple installation by application container management.

Using an integrated index service, the Broker manages the data sources available in the International Data Spaces and supports publication and maintenance of associated metadata. Furthermore, the Broker Index

Service supports the search for data resources. Both the App Store and the Broker are based on the Connector architecture (which is described in detail in the following paragraphs) in order to support secure and trusted data exchange with these services.

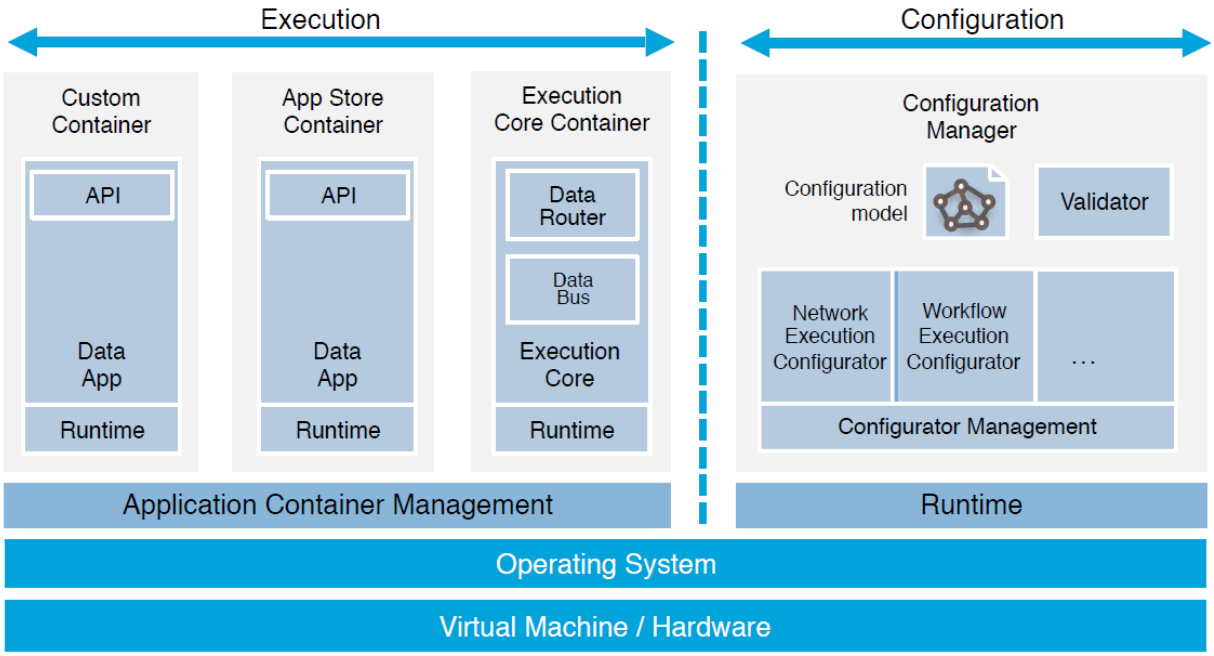

Figure 5 - IDS connector architecture [4]

Figure 5 illustrates the internal structure of the Connector. A concrete installation of a Connector may differ from this structure, as existing components can be modified, and optional components added. The components shown in Figure 5 can be assigned to two phases: Execution and Configuration. The Execution phase of a Connector involves the following components:

- Application Container Management: In most cases, the deployment of an Execution Core Container and selected Data Services is based on application containers. Data Services are isolated from each other by containers in order to prevent unintended interdependencies. Using Application Container Management, extended control of Data Services and containers can be enforced. During development, and in case of systems with limited resources, Application Container Management can be omitted. Difficulties in container deployment can be handled by special Execution Configurators (see below).
- An Execution Core Container provides components for interfacing with Data Services and supporting communication (e.g., Data Router or Data Bus to a Connector).
    - A Data Router handles communication with Data Services to be invoked according to predefined configuration parameters. In this respect, it is responsible of how data is sent (and received) to (and from) the Data Bus from (and to) Data Services. Participants have the option to replace the Data Router component by alternative implementations of various vendors. Differences in configuration can be handled by specialized Execution Configurator plug-ins. If a Connector in a limited or embedded platform consists of a single Data Service or a fixed connection configuration (e.g., on a sensor device), the Data Router can be replaced by a hard-coded software, or the Data Service can be exposed directly. The Data Router invokes relevant components for the enforcement of Usage Policies, e.g., a Policy Enforcement Point, as configured in the connector or specified in the Usage Policy.
    - The Data Bus exchanges data with Data Services and Data Bus components of other Connectors. It may also store data within a Connector. Usually, the Data Bus provides the

method to exchange data between Connectors. Like the Data Router, the Data Bus can be replaced by alternative implementations in order to meet the requirements of the operator. The selection of an appropriate Data Bus may depend on various aspects (e.g., costs, level of support, throughput rate, quality of documentation, or availability of accessories).

- An App Store Container is a certified container downloaded from the App Store, providing a specific Data Service to the Connector.
- A Custom Container provides a self-developed Data Service. Custom containers require no certification.
- A Data Service defines a public API, which is invoked from a Data Router. This API is formally specified in a meta-description that is imported into the configuration model. The tasks to be executed by Data Services may vary. Data Services can be implemented in any programming language and target different runtime environments. Existing components can be reused to simplify migration from other integration platforms.
- The Runtime of a Data Service depends on the selected technology and programming language. The Runtime together with the Data Service constitutes the main part of a container. Different containers may use different runtimes. What runtimes are available depends only on the base operating system of the host computer. From the runtimes available, a service architect may select the one deemed most suitable.

The Configuration phase of a Connector involves the following components:

- The Configuration Manager constitutes the administrative part of a Connector. Its main task is the management and validation of the Configuration Model, followed by deployment of the Connector. Deployment is delegated to a collection of Execution Configurators by the Configurator Management.
- The Configuration Model is an extendable domain model for describing the configuration of a Connector. It consists of technology-independent, inter-connected configuration aspects.
- Configurator Management loads and manages an exchangeable set of Execution Configurators. When a Connector is deployed, the Configurator Management delegates each task to a special Execution Configurator.
- Execution Configurators are exchangeable plug-ins which execute or translate single aspects of the Configuration Model to a specific technology. The procedure of executing a configuration depends on the technology used. Common examples would be the generation of configuration files or the usage of a configuration API. Using different Execution Configurators, it is possible to adopt new or alternative technologies and integrate them into a Connector. Therefore, every technology (operating system, application container management, etc.) gets its own Execution Configurator.
- The Validator checks if the Configuration Model complies with self-defined rules and with general rules specified by the International Data Spaces, respectively. Violation of rules can be treated as warnings or errors. If such warnings or errors occur, deployment may fail or be rejected.

As the Configuration phase and the Execution phase are separated from each other, it is possible to develop, and later on operate, these components independently of each other. Different Connector implementations may use various kinds of communication and encryption technologies, depending on the requirements given.

The International Data Spaces Connector, being a central component of the architecture, is implemented in different variants and can be acquired from different vendors. Nevertheless, each Connector is able to communicate with any other Connector (or other technical component) in the ecosystem of the International Data Space.

- **Operation:** Participants should be able to run the Connector software in their own IT environment. Alternatively, they can run a Connector on mobile or embedded devices. The operator of the Connector must be able to define the data workflow inside the Connector. Users of the Connector must be identifiable and manageable. Passwords and key storage must be protected. Every action,

data access, data transmission, incident, etc. should be logged. Using this logging data, it should be possible to draw up statistical evaluations on data usage etc. Notifications about incidents should be sent automatically.

- **Data exchange**: The Connector must receive data from an enterprise backend system, either through a push-mechanism or a pull-mechanism. The data can be provided via an interface or pushed directly to other participants. To do so, each Connector must be uniquely identifiable. Other Connectors can subscribe to data sources or pull data from these sources. Data can be written into the backend system of other participants.

2.2.3.3 IDS app service

In addition to the broker and connector as two main components of the IDS RAM, the reference model offers an app store which provides data apps. These are applications that can be deployed inside a connector. Data apps facilitate data processing workflows.

Three types of data apps can be distinguished:

- Self-developed data apps, which are used by the data provider's own connector (usually requiring no certification from a certification body).
- Third-party data apps, which are retrieved from the app store (and which may require certification).
- Data apps provided by the connector of the data consumer, which allow the data provider to use certain functions before data is exchanged.

The IDS App Store is a secure platform for distributing Data Apps; features different search options (e.g., by functional or non-functional properties, pricing model, certification status, community ratings, etc.). An IDS App Store consists of a registry for available Data Apps in this App Store. Therefore, an App Store supports operations for Data App registration, publication, maintenance, and query, as well as operations for the provisioning of a Data App to a connector. These basic operations can be complemented by additional services, e.g., billing or support activities.

3 SUMMARY OF REQUIREMENTS FOR THE DATAPORTS PLATFORM

Requirement analysis was a critical factor while designing the architecture of the DataPorts Platform, as it defined the features and characteristics that the system must provide. For this reason, since the beginning of the project's lifecycle, a strict template was created in order to elicit both technical and business requirements of DataPorts at WP level. Initially, the fields of this template included:

- **ID** to distinguish the requirements
- **Type** of the requirement (i.e., functional or non-functional)
- **Category** of the requirement (e.g., Interoperability, Functionality, Security, Privacy)
- **Source** of the requirement, (e.g., if it is derived from an internal technical analysis, or if it is an end-user story etc.)
- **Priority** of the requirement, following the MoSCoW methodology (MUST, SHOULD, COULD, WON'T)
- **Description** of the requirement
- **Rationale** behind the requirement

The initial set of requirements served as input to the architecture specification. However, as the definition of the architecture was being evolved and the conceptual design of the components and their functionalities took place, the requirements descriptions were also being enhanced. To this end, the following fields were added to the template:

- The responsible **Task** to which the requirement belongs
- **Test Case / Acceptance criteria**, to be used to evaluate the fulfilment of the requirement

The goal of these two fields is to improve the management of the requirements and thus to ensure their completion by the end of the project.

Overall, 3 rounds of revisions were performed (M6, M11, M14), resulting in the final list of 92 requirements in total. Most of them (71%) must be completed (MUST), for the project to be considered successful. In addition, there is a clear balance between functional (52%) and non-functional (48%) requirements. However, this balance differentiates at WP level, according to the scope of each one WP. For example, the majority of WP3 requirements are functional, as this WP is devoted to the development of the core functional components of the DataPorts Platform. On the other hand, since WP4 deals with the security and privacy aspects of the platform, most of the requirements there are non-functional.

Regarding the categories (Figure 6) in which the requirements belong, the most frequent are interoperability, functionality, security, and privacy. Indeed, DataPorts aims at defining ontologies, mechanisms and enablers to provide semantic interoperability with data platforms. Furthermore, one of the project's objectives is to offer to the involved stakeholders a trusted and secure environment to share their data, based on the Blockchain technology along with many supported functionalities. Consequently, the most frequent categories were anticipated.

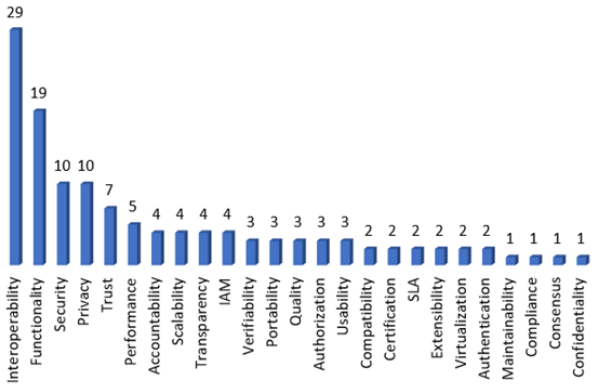

Figure 6 – Requirements per category

Concerning those categories, the following tables present indicative examples of some high-level requirements:

ID	3.1
Type	Functional
Category	Interoperability
Source	Internal Technical Analysts
Priority	MUST
Description	The DataPorts Platform must homogenize the input data so that it can be retrieved in the same format and data model regardless of the data source
Rationale	Due to the big amount of data sources, DataPorts must transform the data to a homogenized format and data model. This process takes care of the structural and content translation to unify its format
Responsible Task(s)	T3.1
Additional Comments	
Test Case / Acceptance Criteria	Check that the data stored in the context broker from the data access agents meets the structure and requirements of the defined data model

Table 1 - A functional requirement of MUST priority that belongs to Work Package 3

ID	3.26
Type	Functional
Category	Functionality
Source	End-Users
Priority	MUST
Description	As an end-user, I want the platform to provide cognitive services specific to ports requirements, so that I could improve my decision-making processes and/or KPIs
Rationale	From the ambition section of the GA (1.4), but also an outcome from the pilot's use case descriptions
Responsible Task(s)	T3.4
Additional Comments	
Test Case / Acceptance Criteria	**Given** a cognitive service to predict the future value of a ports KPI or event, **when** I request the expected value for a specific time period (i.e., the next hour), **then** the service must report visually the prediction and the accuracy

Table 2 - A functional requirement expressed as an end-user story

ID	2.2
Type	Non-Functional
Category	Security
Source	Internal Technical Analysts
Priority	MUST
Description	The DataPorts Platform must provide a secure interface framework for data exchange between itself and the potential data sources
Rationale	Any platform's component to fulfil minimum security and privacy constraints
Responsible Task(s)	T2.5
Additional Comments	Consider security standards (NIST, CSA etc.)
Test Case / Acceptance Criteria	The DataPorts Platform (according to the defined scenarios) share/exchange data in two secure/privacy-protected approaches: off-chain data through IDS connectors (peer-2-peer), and on-chain data were verifiable and immutable data is stored in Blockchain

Table 3 - A non-functional requirement of MUST priority that belongs to Work Package 2

ID	4.2
Type	Non-Functional
Category	Privacy
Source	Internal Technical Analysts
Priority	MUST
Description	As a participating Blockchain organization, I would like to share data in a privacy preserving manner with selected organizations within the business network
Rationale	In multiple business scenarios, organizations would like the ability to share data and/or business logic with subset of their business network, or alternatively share part of the data for public access within this network and hold some data private
Responsible Task(s)	T4.3
Additional Comments	
Test Case / Acceptance Criteria	Use Blockchain networks for each pilot with stakeholder organizations to share data in the network. Use channels between network participants who need to share data in a manner invisible to other participants

Table 4 - A non-functional requirement of MUST priority that belongs to Work Package 4

More details about the requirements elicitation methodology, and the requirements analysis as well as the complete list of the requirements tables can be found in the Deliverable D2.1 [2].

4 DATAPORTS PLATFORM'S SYSTEM FUNCTIONALITIES

In this section, the system functionalities of the DataPorts platform are introduced. For a better organization of the functionalities an excel worksheet is used. The provided table in the worksheet "functionalities" provide a set of functionalities and features of the DataPorts platform. The excel worksheet is maintained in the official repository of the DataPorts project.

In the first round of creating this worksheet, the functionalities and features are extracted directly from the DataPorts proposal [1].

Following this round, the given worksheet has been iteratively refined concerning the identified requirements, and a tight cooperation of the project partners. The partners of the DataPorts project provided their contribution in several rounds, resulting the current tables documented in this section. The worksheet is created and maintained by the University of Koblenz Landau (UKL)[1], which is the leader of the Task 2.5.

In the worksheet, for each functionality, several aspects are identified.

- **ID**: A unique ID which will be used to refer to the functionality during the project and in different documents. The IDs are defined concerning the corresponding work package.
- **Reference in the project proposal (Scope, WP, Task, and Objective)**: Concerning the description of action, this column identifies who is responsible for fulfilling the functionality.
- **Captured requirement**: After specifying the requirements in the first round, the requirements are mapped to the functionalities. For the requirements that could not be mapped to any appropriate existing functionality, proper functionalities were defined.
- **DataPorts' architectural component**: The architectural component that provides the functionality. This column will be iteratively filled out and refined.
- **Concrete existing component/tool in the market**: Already exiting component or tool that supports/satisfies the functionality.
- **Relation/reference to the IDS reference architecture model**
- **Comment**: Any relevant comment, information or description.
- **Added by/reference**: Just for internal tracking issues. The reference part can be included in the deliverable as well.

In this section, due to space limitations, only the technical system functionalities included in the Work Packages 2, 3, and 4 are demonstrated. Moreover, only a few aspects are depicted in the corresponding tables.

ID	Functionality	Reference in the proposal
F-2.1	The DataPorts platform provides various platform governance capabilities and interoperability among different platforms	Task 2.3, Task 3.1, Task 3.2, Task 6.4
F-2.2	The DataPorts platform sets a data driven ecosystem ready for a comprehensive exploitation of data, and virtual data repositories	Task 3.3
F-2.3	The DataPorts platform introduces a novel, decentralized architecture, data, and events can be recorded on a blockchain for transparency and credibility	Task 2.5
F-2.4	The DataPorts platform implements all the authentication and authorization mechanisms to allow data sharing and trading in a secure and reliable way	WP 4

[1] https://www.uni-koblenz-landau.de/de/koblenz/fb4/ist/rgse/agse_home

ID	Functionality	Reference in the proposal
F-2.5	The DataPorts platform will be aligned with IDS reference model, offering data owners the option to describe connectors where type and conditions of data will be clearly stated and offered to data consumers	Task 2.5
F-2.6	The DataPorts platform provides Orion Context Broker and Blockchain component, registering the description of the data	Task 2.4
F-2.7	The DataPorts platform enables the data owners to exchange data	Task 2.3, Task 4.2, T4.3, T3.5
F-2.8	The DataPorts platform enables data sovereignty	Task 2.5

Table 5 – DataPorts functionalities concerning WP2

ID	Functionality	Reference in the proposal
F-3.1	The DataPorts platform enables connections with external sources of data supported by data agent's manager	Task 3.1
F-3.2	The platform will enable a connection among current IT systems in ports environment, allowing them to share data and knowledge	Task 3.1, Task 3.2
F-3.3	The DataPorts platform provides data sanitization algorithm to guarantee data integrity	Task 3.3
F-3.4	The DataPorts platform offers machine learning models	Task 3.4
F-3.5	The DataPorts platform enables the federation of data varying in syntax and semantics	Task 3.1, Task 3.2
F-3.6	The DataPorts platform provides efficient and effective techniques for data wrapping to represent the underlying mechanism to support a selective, release, storage, and analytics on data	Task 3.1
F-3.7	The DataPorts platform provides semantic stream processing	Task 3.4
F-3.8	The DataPorts platform provides semantic data compression	Task 3.1, Task3.2, Task 3.3
F-3.9	The DataPorts platform provides declarative, distributed data aggregation	Task 3.3
F-3.10	DataPorts platform provides an innovative user interface to guide the user in specifying privacy and data access policies	Task 3.1, Task 3.5
F-3.11	The DataPorts platform provides the data owners data driven analytic services	Task 3.4
F-3.12	The DataPorts platform provides the consumers and end users new AI and cognitive applications	Task 3.4 (Overlaps with F3.5)
F-3.13	The DataPorts platform provides tools to help decision processes	Task 3.4
F-3.14	The DataPorts platform provides smart API for cognitive services	Task 3.2, Task 3.3, Task 3.4
F-3.15	The DataPorts platform processes streams of records and publish and subscribe to streams of data	Task 3.2
F-3.16	The DataPorts Platform provides a framework for semantic interoperability from several sources.	Task 3.2
F-3.17	The DataPorts Platforms offers an ontology to guarantee semantic interoperability.	Task 3.1, 3.2
F-3.18	The DataPorts Platform provides REST-style interaction with Linked Data.	Task 3.2

ID	Functionality	Reference in the proposal
F-3.19	The DataPorts Platform provides a data source metadata registry	WP 4
F-3.20	The DataPorts Platform provides data from a federated database on demand	Task 3.1, 3.2
F-3.21	The DataPorts Platform provides data from a federated database through publish and subscribe model	Task 3.1, 3.2

Table 6 – DataPorts functionalities concerning WP3

ID	Functionality	Reference in the proposal
F-4.1	Provide a platform to ensure data sharing among the actors operating in diverse supply chains per the defined data governance rules that respect the competitive advantage of all (who access what).	WP4
F-4.2	The DataPorts platform provides services to ensure security and protection of shared data	WP4
F-4.3	The DataPorts platform ensures the needed anonymization or de-identification mechanisms while preserving the individual features required for effective big data analytics	Task 4.4
F-4.4	The DataPorts platform provides clear rules on how data will be accessed	Task 4.2, Task 4.3
F-4.5	The DataPorts platform provides flexibility of policies on data distribution	Task 4.2
F-4.6	The DataPorts platform provides end to end secure environment	WP4
F-4.7	The DataPorts platform ensures full compliance with GDPR	Task 4.1, Task 4.3, Task 4.4
F-4.8	The DataPorts platform enables efficient processing over protected data while preventing (or limiting) access to actual data content by other parties	WP 4

Table 7 – DataPorts functionalities concerning WP4

5 DATAPORTS' ARCHITECTURE

5.1 DATA PLATFORM'S ARCHITECTURE

DataPorts will improve the transition of European seaports from connected and digital to smart and cognitive, by providing a secure environment for the aggregation and integration of data coming from the several data sources and owned by different stakeholders. This will provide the whole port community with real value in order to improve their processes, offer new services and devise new AI-based and data-driven business models. DataPorts will create an even more trusted, reliable and efficient way of conducting businesses in Europe and to reinforce the European Single Market.

To this end, in the context of the DataPorts project, an industrial data platform (cognitive ports data platform) will be designed that:

- Connects to different digital infrastructures currently existing in digital seaports, enabling the interconnection of a wide variety of systems into a tightly integrated ecosystem,
- Sets the policies for a trusted and reliable data sharing and trading based on data owners' rules and offering a clear value proposition, and
- Leverages on the data collected to provide advanced data analytics services based on which the different actors in the port value chain cloud develop novel AI and cognitive applications.

In this section, two figures are provided, introducing the architecture of the DataPorts data platform from two perspectives:

- A high-level perspective describing only the main components of the platform including the main functionalities (Figure 7)
- A detailed perspective where the sub-components and their corresponding interactions are depicted (Figure 8).

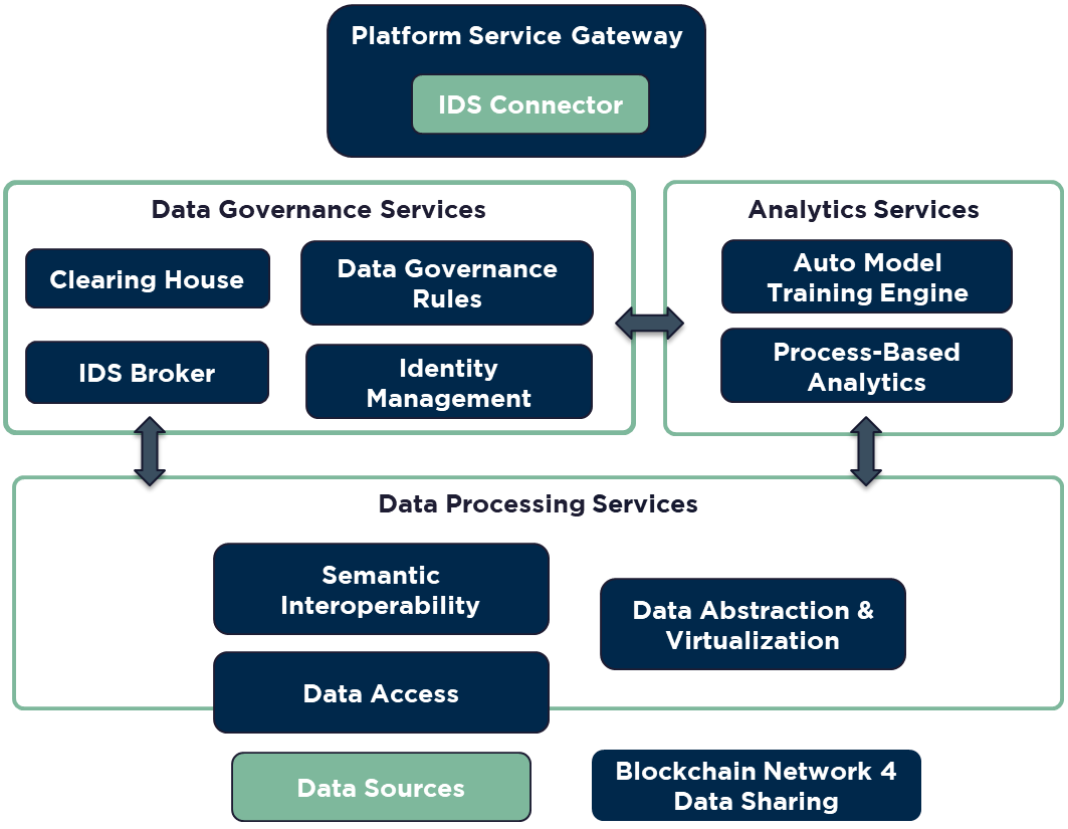

Figure 7 - The high-level architecture of the platform

The methodology that is used to develop this architecture and the corresponding revolution from the initial architecture are introduced in Section 2.1. The current architecture is the result of several weekly meetings with all project partners. The current document and the containing architecture are finalized after four iterative rounds of documentation and review. Therefore, the document and the architecture reflect the perspective and the opinion of all project partners. In different project plenaries and in the context, several workshops the architecture is discussed, and improved.

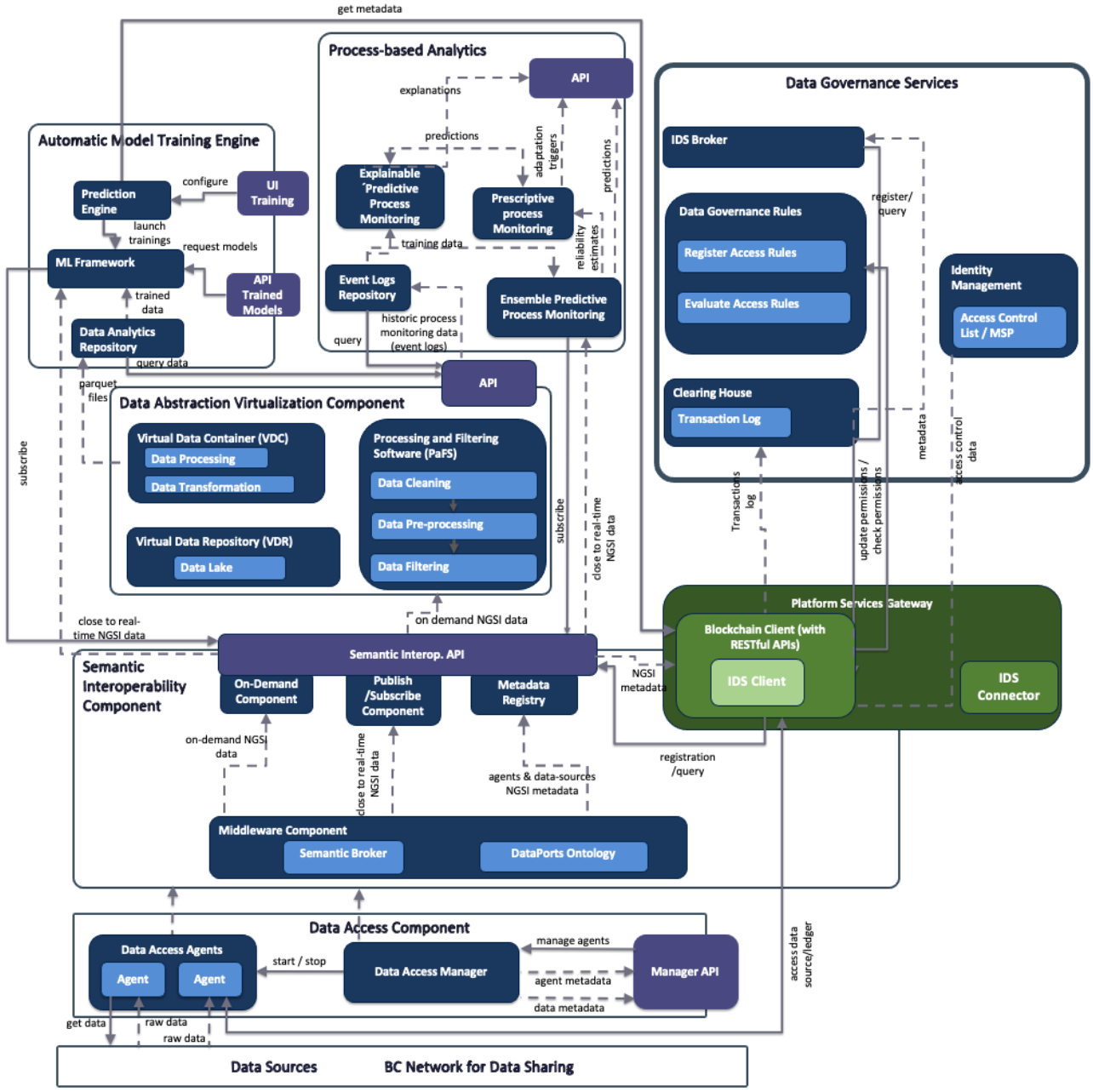

Figure 8 - The detailed architecture of the platform

Section 6 gives complete details on the components, including their contribution to the platform, requirements covered, detailed descriptions, interactions between them, and subcomponents and tools.

5.2 APPROACHES FOR DATA EXCHANGE AND DATA SHARING IN DATAPORTS PLATFORM

In the context of the DataPorts platform, two main approaches for data sharing and data exchange have been taken into consideration, on-chain and off-chain storage.

5.2.1 Data is stored off-chain

In this approach we adopted concepts from the International Data Space (IDS) reference architecture model. In a nutshell, we consider the peer-to-peer data exchange between data owners and data consumers through IDS connectors. To ensure the security and privacy aspects, blockchain manages the consents of access to the data. Smart contracts decide if a particular access to data is allowed considering the invoker's credentials and the specification of access rules for the data.

Following IDS architecture principles, the data remains at the data owner's premises and is exchanged using peer-to-peer communication mechanisms, while the data access rights to the data are declared and verified using blockchain (BC) data governance smart contracts. Data owners remain in full control of their data and might change at any time the terms of the data usage. The BC Data Governance component is leveraged to act as IDS broker and clearing house components in an IDS compliant architecture. BC data governance solution also provides functionality for data providers to define access rules for their datasets on the one hand, and to monitor the actual access to the data-by-data consumers, on the other hand. In this approach, the datasets are held at the owner's premises (stored off-chain), and only the data access rules, governance policies, and data access requests are stored on-chain on the BC ledger.

5.2.2 Data is stored on-chain (blockchain for shared data)

Already at the first stages of the project it was evident that some use cases of the ports should benefit from applying BC technology and, therefore, might require applying BC in a shared-data manner. In this approach, the data is shared in a transparent and verifiable manner among the port's ecosystem participants, and not held at the premises of a single business entity (as in the first approach). In those cases, the data is stored on the ledger (data on-chain) as opposed to data stored off-chain as in the case of the governance policies approach. The transactions occurring in the process are recorded in an append-only fashion on the ledger and become available to all participants in the BC network. For instance, cold chains alarms of a container alerting upon temperature violations can be recorded, allowing all participants of the BC network to be aware of the event and act upon it. BC assures that this information is true and reliable serving as a single source of truth and providing transparency and a non-repudiation process.

The general flow of data exchange in the platform is demonstrated in Figure 9 and described henceforth from a data provider and data consumer perspectives. The figure is intended for illustrative purposes only as it is a simplified visual description of the actual invocation of components within business flows for data access. All the communication between the users and Dataports components is done via the platform service gateway, which in turn interacts with internal platform components via their exposed APIs. Needless to say, that all communication is done in a secure manner, allowing only authenticated and authorised users to invoke appropriate component's APIs (using platform-wide identity management solution for certification and authorization of access).

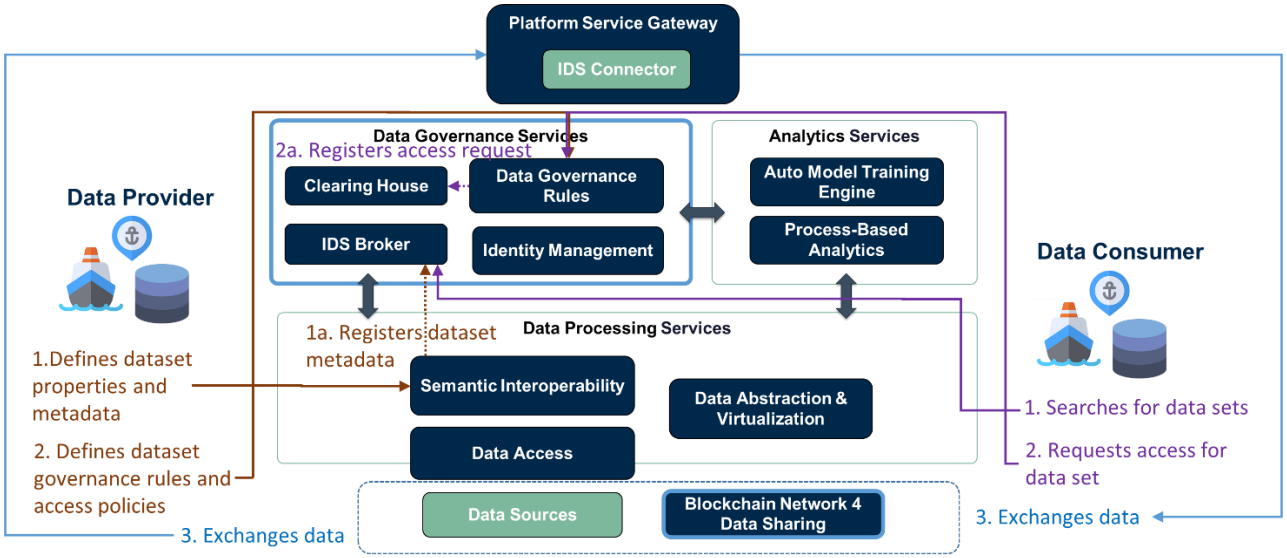

Figure 9 - Data exchange flows in Dataports platform

The data provider, owning the data source to be exchanged with data consumers on the platform, registers all the dataset pertaining information and metadata using the data access and semantic interoperability layer. Important to note, that a shared data BC can also be considered as a data source in a similar way (Step 1). Then, the semantic interoperability layer communicates with the data governance component, registering the data set metadata in the Broker. The Broker, according to IDS RAM, acts as a catalogue for platform's available data sources (Step 1a). Following, the data provider defines governance policies and access rules for the data source: who, when, and under which conditions someone can access the specified data source (Step 2).

The data consumer searches for an available data source in the Broker (Step 1). When a suitable data source is found, the data consumer can request access for this data source (Step 2). The data governance component verifies the governance policies and access rules pertaining to this data source and verifies if the specific data consumer can access the data source based on the owners' defined policies. The access request is also registered in the clearing house which acts as a transparent, immutable and verifiable audit log for all access requests and actual data exchanges (Step 2a). If the access request for the data set is allowed, the data exchange between data provider and data consumer is realized by secure communication mechanism based on IDS RAM specifications. While the actual data exchange is done outside the BC, the transaction is logged in the clearing house for auditability purposes.

After access is granted to the data consumer to the data set per the governance rules stored in BC, the actual exchange of data would be carried out in a secure manner following IDS specifications (Step 3)

Further information related to data governance and shared data BC networks architecture and implementation can be found in deliverables D2.3, D3.4, and D4.2[2].

Next sections of this deliverable will dive deeper into the platform components functionalities and their interactions.

5.3 ADRESSING SECURITY IN THE DATAPORTS PLATFORM

During the development of the DataPorts architecture, several mechanisms for securing the system have been defined. The purpose of these mechanisms is to protect the system and the participants from external attacks and security incidents.

[2] https://dataports-project.eu/deliverables/

Security incidents are the unplanned events or near-events that may have undesired effects that can result in data breaches, business continuity incidents or information security incidents. The costs of these unplanned events may lead to huge economic losses, laws infringements and reputational damage.

In order to protect the DataPorts architecture and avoid potential security incidents, it is necessary to consider relevant standards, regulations and good practises:

- ISO 27001[3]: This is one of the most relevant standards in cybersecurity. It specifies how Information Security Management System must be defined and implemented. Every information management system that needs to be secure should consider this standard.
- NIS Directive[4]: Is the first piece of EU-wide cybersecurity legislation. This directive describes all the cybersecurity requirements that operators of essential services from critical operators, like the transport one, must comply with.
- GDPR[5]: This is the European personal data protection regulation. Any data that can identify a person (like telephone number, photo or e-mail) is considered personal data. Every information system that manages personal data must consider this regulation and adapt to it.
- IDS Architecture model[6]: Is the architecture model propose by the IDSA. The mission is to define a secure and sovereign system of data sharing in which all participants can realize the full value of their data.

For the definition of the DataPorts architecture, new technologies have been considered to improve the protection of data managed by the architecture. This is the case of blockchain technology, chosen as core of the Data Governance Services. From a security point of view, blockchain improves the architecture by:

- Improving the reliability of the system through the decentralization, which avoids potential fraud or loss of governance in the governing authority.
- Ensuring the confidentiality and integrity of data through the continuous encryption of data.
- Ensuring traceability of actions by maintaining a log of transactions in the blockchain solution.
- Backing up data through decentralization, because each node has a copy of the data.
- Being a necessary solution for implementing the secure data exchange in the IDS architecture model.

The identity of participants is also addressed though blockchain, ensuring the pseudonymity of participants by encryption. The deployment of a traditional centralized identity management tool could be relevant for complementing the decentralized identity management solution addressed in the blockchain networks, and therefore, will be discussed below.

Through blockchain it is ensured that only trusted participants can interact with the platform. This capability, together with the traceability of actions, minimize the potential attacks caused by malicious participants.

Finally, for reviewing and improving the security of the architecture, several cybersecurity consulting services has been carried out. A Risk Assessment is carried out for analysing and addressing all the potential cybersecurity risks. An anonymization procedure is defined for secure confidential data before being injected in the platform. Lastly, a static and a dynamic test will analyse the potential vulnerabilities of the platform.

[3] https://www.iso.org/standard/54534.html

[4] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:32016L1148

[5] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32016R0679

[6] https://internationaldataspaces.org/wp-content/uploads/IDS-Reference-Architecture-Model-3.0-2019.pdf

5.3.1 IDS Architecture Model

The IDS architecture model enforces the security in the platform by requiring that several security features are implemented:

- Security in communications: All communications must be secured by implementation a point-to-point encryption and an end-to-end authorization. This is implemented by using HTTPS protocol in communications.
- Identity management: The platform must address the access control related decisions that are based on reliable identities and properties of participants. Therefore, an identity management solution that controls the identification, authentication and authorization of participants must be implemented. The implementation of this solution is presented below.
- Trust management: To establish trust across the entire ecosystem, cryptographic methods must be in place. The objective is to allow each entity to authenticate against other Participants by defining secret keys for allocating every entity. In order to do so, a Public Key Interface (PKI) must be defined. In this architecture, the Hyperledger Fabric CA is defined as Certification Authority (CA), issuing certificates based on PKI (Public Key Infrastructure) to organizations and their users.
- PKIs and MSPs work together: a PKI provides a list of identities, and an MSP indicates which identities belong to the members of a particular organization that participates in the network.
- Trusted platform: To ensure reliability in the platform, it must specify the minimal requirements for Participants, to establish trustworthy relationships for participants and to enable trustworthy execution of Data Apps, guaranteeing system integrity and strong isolation of components. Separation of components and identity reliability are the key controls for ensuring this security feature.
- Data access control: In information security, access control defines the restriction of data access. Authorisation controls are key for grating access to the necessary data, but also to keep tracking on which participant access to each data. For this purpose, the Data Access Component has been defined.
- Data usage control: Is an extension of the data access controls feature. It specifies and enforces the restrictions for controlling data, defining what may be done with the data, and what not. The rules for data sharing are defined in the smart contracts of the Data Governance component.
- Data provenance tracking: This security property tracks when, how and by whom data was modified, and which other data influenced the process of creating new data items. This kind of traceability is similar to the data protection requirements for data controllers. This feature is implemented by the Clearing House component, which is implemented using the inherent functionality of the Blockchain (transaction logs).

5.3.2 Blockchain Security Features

The purpose of the blockchain technology in this project is bifunctional: it implements the Data Governance and the Data Sharing components. These components are an important element of the architecture, and therefore, their relevance within the platform is significant.

The blockchain proposed in this architecture is implemented following Hyperledger Fabric. This implementation adds some security features to the platform due to the way the blockchain is defined:

- Nodes can be easily created, started, stopped, reconfigured, and deleted, unlike a traditional blockchain node. This is a good practice for avoiding incidents from spreading between nodes.
- The nodes are provided by the organizations belonging to the consortium, what ensures the trustability of the nodes, and therefore, the trustability in the platform.
- Data privacy is achieved in two ways. On the one hand, through the channels themselves. Each channel has its own ledger, and only the organizations participating in the channel can access its content. In addition, private transactions can be created within the same channel, so that only a

subset of the channel's organizations can see it. As an alternative, customers can encrypt their data before making the transaction and then establish a mechanism to share the key with the recipient.

5.3.3 Identity Management

The identity management is a core component of the platform, necessary according to the IDS architecture model and to the NIS Directive security requirements. Identity management is addressed by two elements, the centralized identity manager implemented using Keycloack, and the decentralized solution implemented in by the blockchain module, responsible of the authorisation between components.

As Hyperledger Fabric is a private and authorized network, only identified and identifiable participants can make transactions and view them. This is a first approach of how access to the network is being limited only to “potentially trusted” participants.

To manage identities Fabric includes an MSP component (Membership Service Provider), an abstract component of the system that provides credentials to clients and their peers. MSP is responsible for abstracting all the cryptographic mechanisms and protocols that manage the issuance and validation of certificates and the authentication of users.

Each local MSP includes a keystore with a private key for signing transactions and a signcert with a public x.509 certificate. In addition, an MSP can allow the identification of a list of identities that have been revoked.

An MSP is not only able to list who is a network participant or member of a channel, it can also identify specific roles within the organization that represents the MSP and establish the basis for defining access privileges in the context of a network and channel.

Hyperledger Fabric CA is the default Certification Authority (CA), issuing certificates based on PKI (Public Key Infrastructure) to organizations and their users. The CA issues two different certificates: a root certificate (rootCert) for each organization and an enrolment certificate (ECert) for each authorized user.

PKIs and MSPs work together: a PKI provides a list of identities, and an MSP indicates which identities belong to the members of a particular organization that participates in the network. By default, the Fabric CA server and client store private keys in a PEM-encoded file. All Hyperledger Fabric CA servers in a cluster share the same database for keeping track of identities and certificates.

Specifically, X.509 certificates are used in client application transaction proposals and smart contract transaction responses to digitally sign transactions. Subsequently the network nodes who host copies of the ledger verify that transaction signatures are valid before accepting transactions onto the ledger.

In order to ensure proper integration with the rest of components in DataPorts, user identity must be compatible across all parts of the network. For this purpose, to define and implement a traditional and centralized identity management module is being discussed. Keycloack is an identity and access management solution which allows single sign-on, to enable two factor authentication and to be integrated into LDAPs. Keycloack consist of a server and an application adapter that must be incorporated into the different modules of the architecture. Keycloack is a potential open-source solution that is being analysed in order to implement the centralized identity management module.

The interplay between the centralized identity management module and the decentralized one is discussed below.

Hyperledger Fabric provides certificates for the blockchain components of the platform, and for them to be able to be used among different platform components, a one-time, auto-verifiable token system is proposed. These tokens are created when identification is required and can be validated inside the chaincode at the moment of creation. The platform-wide auto-verifiable token is read and validated by the Fabric CA, who then provides a blockchain general certificate to be used for blockchain related operations, whose identity is linked to that of the user for which the token was minted.

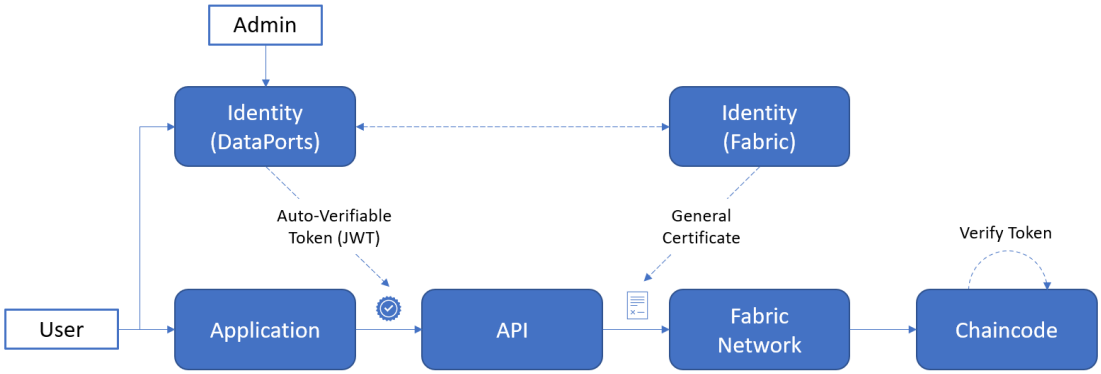

Figure 10 – Identity Management Approach

5.3.4 Cybersecurity consulting services

During the implementation of the platform, several cybersecurity activities are being carried out. The aim of this activities is to review the security status of the platform and to make suggestions on which security features may be improved, enhancing the security level of the hole platform.

The first activity carried out is a Risk Assessment. The aim of the Risk Assessment is to review the architecture and to identify and assess potential security risks. Once the risks are assessed, countermeasures to face those risks are proposed. Also, the standards, regulations and good practices identified above had been considered, for proposing security measures in order to follow and comply with them.

The following activity is a penetration testing of the platform. A penetration test is a dynamic test, which is a testing of the platform when the application is running to detect vulnerabilities in the functionality. An external attack is simulated by the testers, with the aim of discovering how potential attackers can penetrate or damage the platform.

The next activity is a security code review. This is also known as static testing, and it consist in a test of the code. The aim is to identify potential vulnerabilities in the code and dependencies.

The object of these two testing activities is to identify vulnerabilities in the platform and to study how they can be exploited, thus they can be patched, and external attacks can be avoided.

Lastly, an anonymization procedure is defined for confidential data that want to be shared in the platform. This procedure defines how data must be anonymized before being shared.

6 DATAPORTS' ARCHITECTURAL COMPONENTS

In this section various components of the data platform are introduced separately. Moreover, potential interactions between the components are described.

6.1 DATA ACCESS COMPONENT

Data Access Component (DAC) is the entry point to DataPorts. It is the component responsible for accessing the various data sources connected to the DataPorts platform. Among these data sets are the following:

- Internal Data. Data from different sources (different actors in the port ecosystem). For example, data from the Port Community System, IoT devices, or also third-party applications running in the port.
- External Federated Data. Data from other organizations that have also integrated DataPorts in their ecosystem and provide data to the platform. Data coming from other platforms or ecosystems with a common data model.
- Open Data. Public Data sources. For example, a third-party API.

Figure 11 depicts the different dataset types and examples of each one.

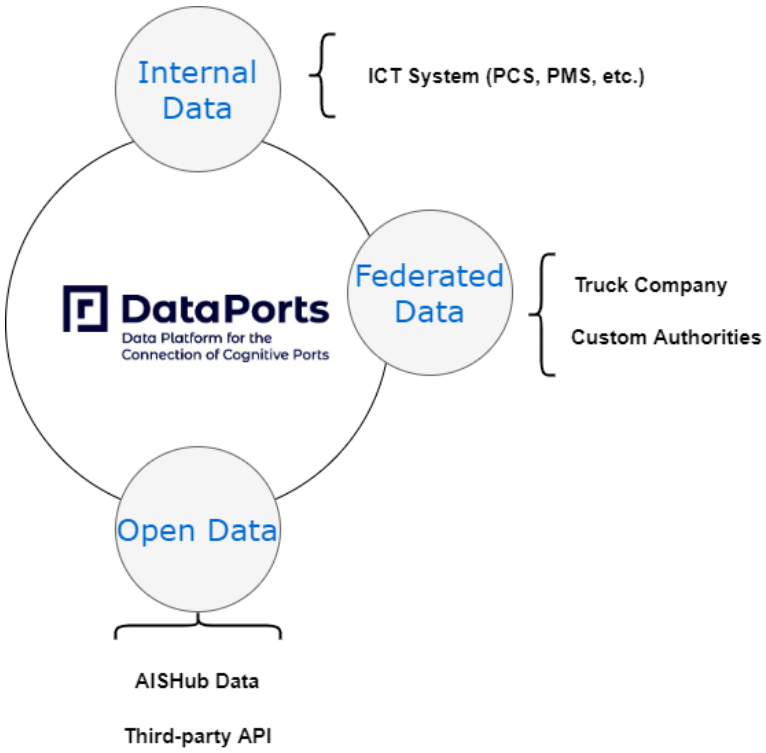

Figure 11 - Different dataset types accessible through DataPorts

6.1.1 Position in the Architecture

The Data Access Component is the lowest component of DataPorts platform. It accesses and connects to the data and feeds the Semantic Interoperability component. It is the only component of the DataPorts platform capable to interact with the data sources and provide access to their raw data.

6.1.2 Contribution to the Platform and the Platform's Objectives

The Data Access Component provides the mechanisms to build connections and get data from many kinds of sources based in the agent or connector development and include them as available data sources for the rest of the platform. For that purpose, there is a wizard that helps organizations to connect their data through the provided agent templates or also to create new ones thanks to the component's SDK.

The component also lets organizations manage these connections through the application's user interface, so they can monitor the status and events happening in real time.

For example, for each of the scenarios defined in the use cases, there is described where the data is, the current source. In order to be able to get this data and send it to DataPorts, the Data Access Component will implement an agent for each of the sources, specifying all that is needed to extract the information from the data base and make the data available for the rest of the platform in a comprehensive format.

The Data Access Component works together with the Semantic Interoperability Component to make possible that other components of the platform use the data without knowing the format or technology in which it was stored on each of the sources. The Data Access component will connect and get the data from the sources and then transform it following the Semantic Interoperability defined models and rules, so when the result is stored, it is available for other components through the Semantic Interoperability API.

6.1.3 Addressed Requirements

Table 8 shows the addressed requirements:

ID	Description	Category
R3.1	DataPorts platform must homogenize the input data so that it can be retrieved in the same format and data model regardless of the data source.	Interoperability
R3.2	The Data Access component has to be able to make use of the interface of each data source connected to the platform in order to retrieve the data in its original format. Supported data interchange formats should be at least: OWL, RDF, XML and JSON. The Data Access component should also support the security mechanisms implemented in each data source.	Interoperability
R3.3	Each data source connected to DataPorts infrastructure should provide version management for its API in order to allow a proper identification of their API and data formats.	Interoperability
R3.4	Each data source that should be connected to the DataPorts infrastructure should provide documentation about its API (such as a swagger definition) so that the connector can be implemented.	Interoperability
R3.6	The DataPorts platform must be able to obtain data from platform/sensors/sources with and without explicitly defined ontology in order to support as many potential data sources as possible.	Interoperability
R3.8	The Data Access component must provide the proper connectors to communicate with the available data sources.	Interoperability
R3.9	The Data Access component should support the most common communication protocols in order to be able to communicate with as many potential data sources as possible.	Interoperability
R3.23	The components of the DataPorts platform could be virtualized in order to ease its deployment and portability.	Virtualization, Scalability
R3.24	The DataPorts platform could offer mechanisms for the automatic deployment, maintenance and scaling of the developed components.	Virtualization, Scalability
R3.31	As a data scientist I want the platform to deal with the original data sources heterogeneity, real-time (streaming) or persistent data, relational or non-relational databases (NO-SQL), so that I can use the data without taking into account the underlying storage system	Interoperability

ID	Description	Category
R3.32	As a data provider I want to share data with the DataPorts platform so it is available for the platform users with the access rights.	Functionality, Interoperability
R3.33	As a data consumer I want to subscribe for changes in a property of an entity so I would be noticed when the property is updated.	Functionality, Interoperability
R3.34	As a data consumer I want to be able to cancel a current subscription so I stop receiving data modifications.	Functionality, Interoperability
R3.35	As a data consumer I want to subscribe for changes in a data entity so I would be noticed when any of its values is updated.	Functionality, Interoperability
R3.36	As a data consumer I want to ask for all the information of a data entity hosted by other DataPorts instances so I can gather all available data.	Functionality, Interoperability
R3.46	Integration of legacy sources. DataPorts platform must be integrated with all the relevant port platforms	Compatibility

Table 8 - Addressed requirements for Data Access Component

6.1.4 Description of the Component

The Data Access Component is the responsible for gathering, transforming and publishing data from different data sources to the platform. It supports many kinds of sources, like file parsing, data base queries or API calls. The component is extensible and so is not limited to accessing the various data sources that are shared on the platform in this project.

For that purpose, it provides different functionalities that can be split in two sections: data processing and component management.

First, the component has the capabilities for getting data from data sources of many different types thanks to the implementation of the agents, that will be developed for accessing each of the source kinds with a customizable periodicity, processing and transforming the information received into data models supported by the platform. Once the data is understandable by the platform, the agents will interact with other components publishing, sharing or sending the result of previous steps.

Second, the tools and mechanisms to manage the processes execution and status. The component provides an API and user interface for the management of the processes. Agents can be created, deleted, started and stopped, or also executed depending on the agent's kind. Logs, properties and status can be checked through the functionalities of the component. The Data Access Component also provides the tools for helping the development of new agents and the subsequent integration in the component.

Table 9 shows the list with the functionalities implemented by the Data Access Component:

ID	Description	Captured by requirement
F-3.1	The DataPorts platform enables connections with external sources of data supported by data agent's manager	R3.2 R3.6 R3.8 R3.9 R3.46
F-3.5	The DataPorts platform enables the federation of data varying in syntax and semantics	R3.1 R3.6, R3.46
F-3.6	The DataPorts platform provides efficient and effective techniques for data wrapping to represent the underlying mechanism to support a selective, release, storage, and analytics on data	R3.1, R3.2, R3.3, R3.4, R3.9, R3.31
F-3.10	DataPorts platform provides an innovative user interface to guide the user in specifying privacy and data access policies	R3.46

ID	Description	Captured by requirement
F-3.17	The DataPorts Platforms offers an ontology to guarantee semantic interoperability.	R3.1
F-3.20	The DataPorts Platform provides data from a federated database on demand	R3.32, R.3.36
F-3.21	The DataPorts Platform provides data from a federated database through publish and subscribe model	R3.33, R3.34, R3.35

Table 9 - Functionalities implemented by the Data Access Component

6.1.5 Interactions of the Component

There are two kinds of agents: publish/subscribe and on-demand. The first ones run once or periodically depending on their configuration. The on-demand agents are executed when an external call is performed. The DAC exposes an API for the management of the agents and for calling the on-demand execution of them. The information that is generated as a result of the execution of the agents can be published on a context broker, that is shared with the Semantic Interoperability component or sent to a call back endpoint. The execution of the agents and the subscriptions are managed from the Semantic Interoperability component, but the results can be returned both to that component or to a given target. So, the DAC interacts mainly with one component in the DataPorts platform which is the Semantic Interoperability Component, other components or systems as the receivers of the on-demand responses and with all the different data sources outside the platform as shown in the Figure 12 below:

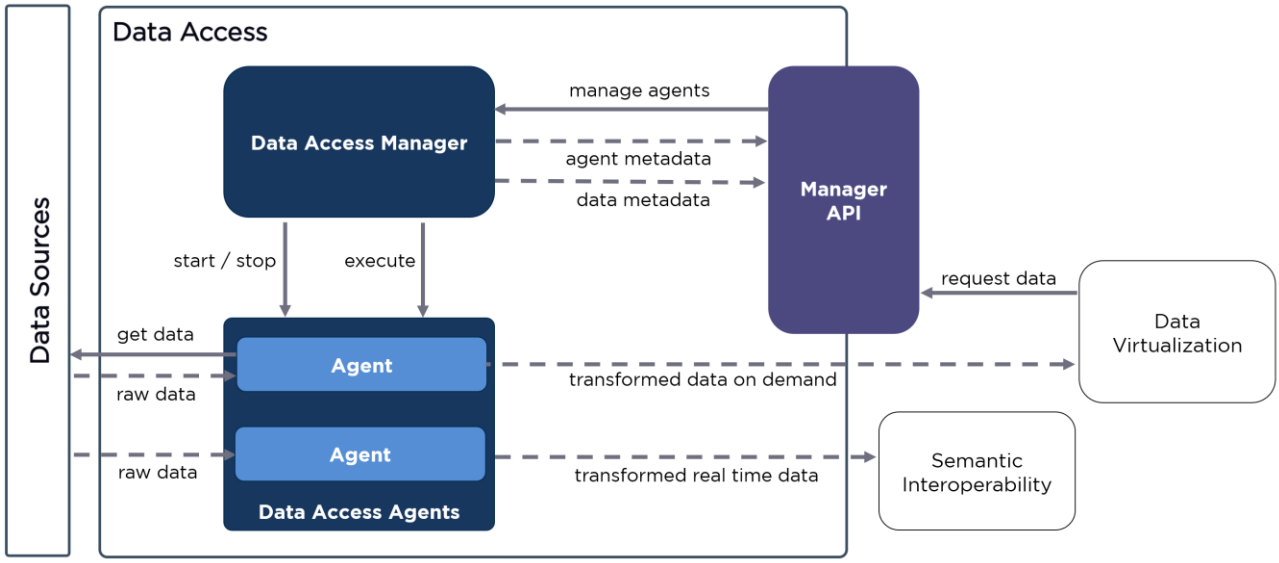

Figure 12 - Interactions of DAC

6.1.6 Subcomponents / Tools

Figure 13 depicts the subcomponents involved in the DAC.

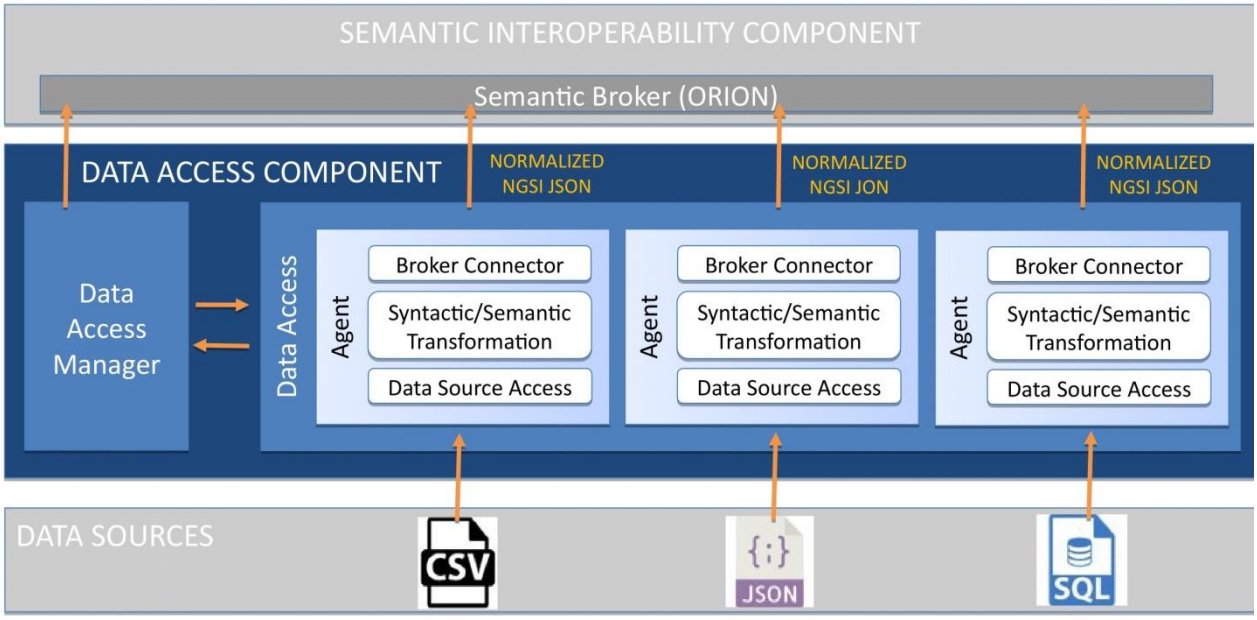

Figure 13 - Subcomponents of the DAC

The DAC component is composed of two subcomponents:

- **Data Access**. It is a set of agents that will contain as many as different data sources need to be accessed. These agents will be responsible for:
    - Access to the Data Source. Get the data from the external systems.
    - Syntactic / Semantic transformation. Transform the data to the platform ontology through the Data Models supported by DataPorts.
    - Publish data as normalized NGSI JSON into the Orion Context Broker in the Semantic Interoperability Component.

    The chosen option to build these agents is by using pygnsi which is a Python framework to write NGSI entities to Fiware Orion (hosted on GitHub at: https://github.com/pixel-ports/pyngsi). Agents are integrated in the component as docker containers, so they will be integrated in the component following the defined, and provided, methodology and tools.

- **Data Access Manager (DAM)**. It is the subcomponent responsible for managing all the tasks related to the Data Access Component. So, it has two different parts, the API and the UI.

    The API provides the functionalities of the component and the way to execute them. These functionalities are focused on the agent management, including the creation:

    - Create agent template
    - Import new agent

    And once the agents are integrated in the component:

    - Instantiate agent
    - Start agent
    - Stop agent
    - Execute agent
    - Delete agent
    - Check agent logs
    - Check agent properties

    The user interface supports the execution and management of the functionalities:

- **UI to create instances (agents)** of the different images available in the platform. Figure 14 depicts an example of UI for this functionality. Creating the instance, the agent will start.

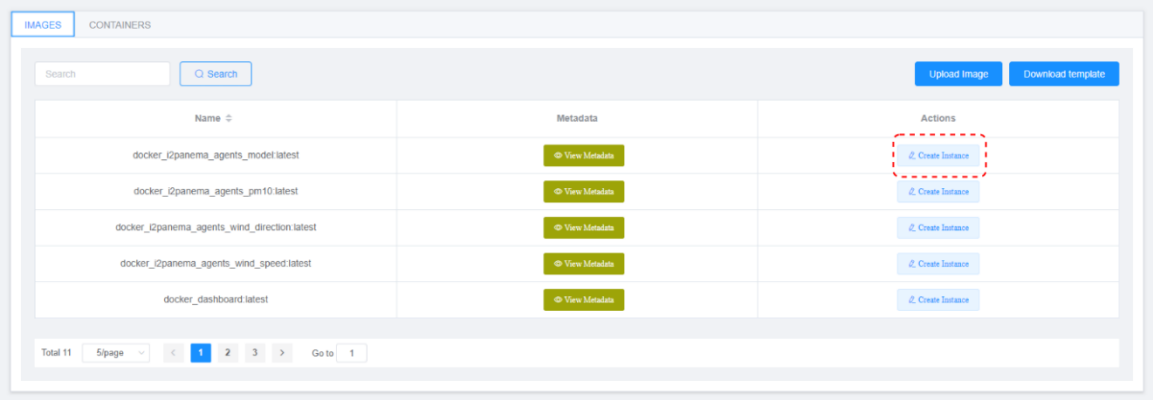

Figure 14 - Example of UI that creates instances / agents

- **UI to manage the agents** available in the platform. The actions allowed over the agents will be for example: stopping the container, deleting the agent or playing again the container after a pause. Figure 15 depicts this functionality.

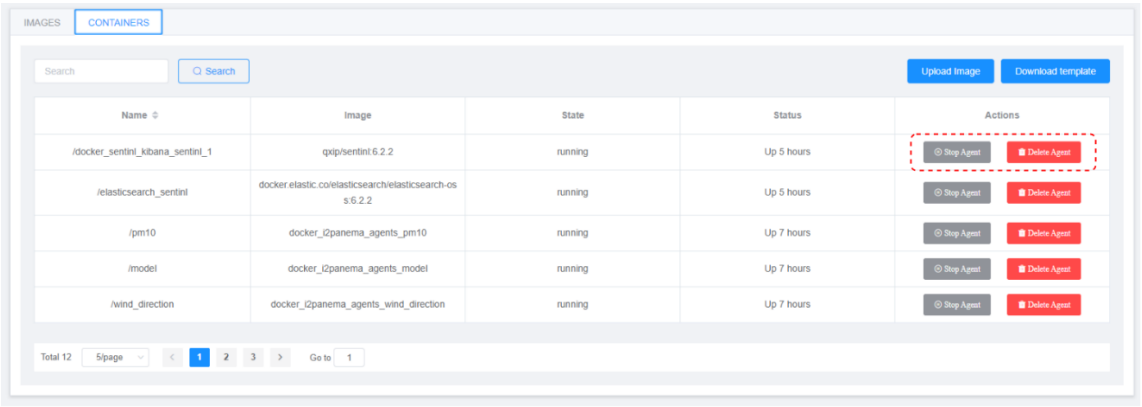

Figure 15 - Example of UI that manages the different actions over the agents

- **UI for registry / uploading agents** to the platform. It will be possible to upload the image of the agent to the platform through the UI. Figure 16 depicts the button for that functionality.

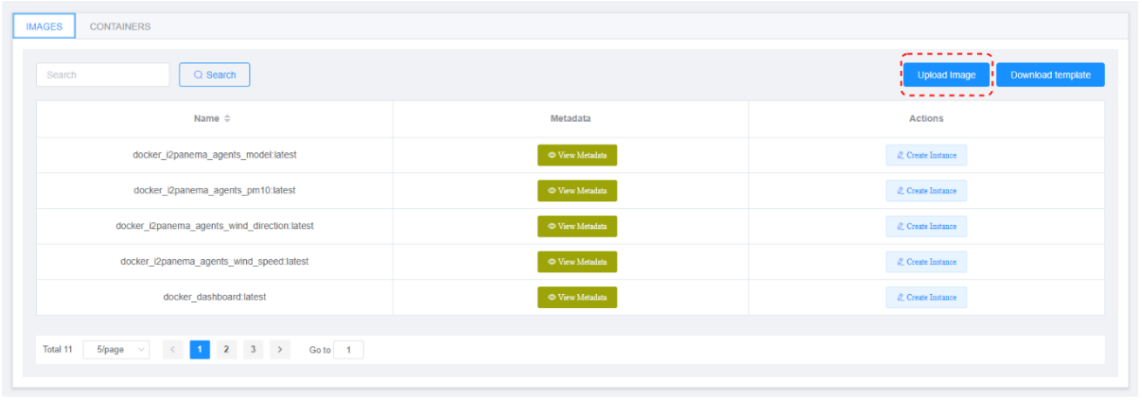

Figure 16 - Example of UI that registry images in the platform

- **Wizard / Template for creating agents**. With the aim to help in the development of the agents a wizard will be included in the UI. This wizard will have different steps:
    1. Select data source. Choose among the different options available.

2. Type of agent. Publish/subscription or on-demand.
3. Select Data Model. It will be possible to choose the Data Model for the agent.
4. Configuration. In this step the user will complete the parameters needed for the agent.

Figure 17 depicts the wizard and the different steps for this functionality.

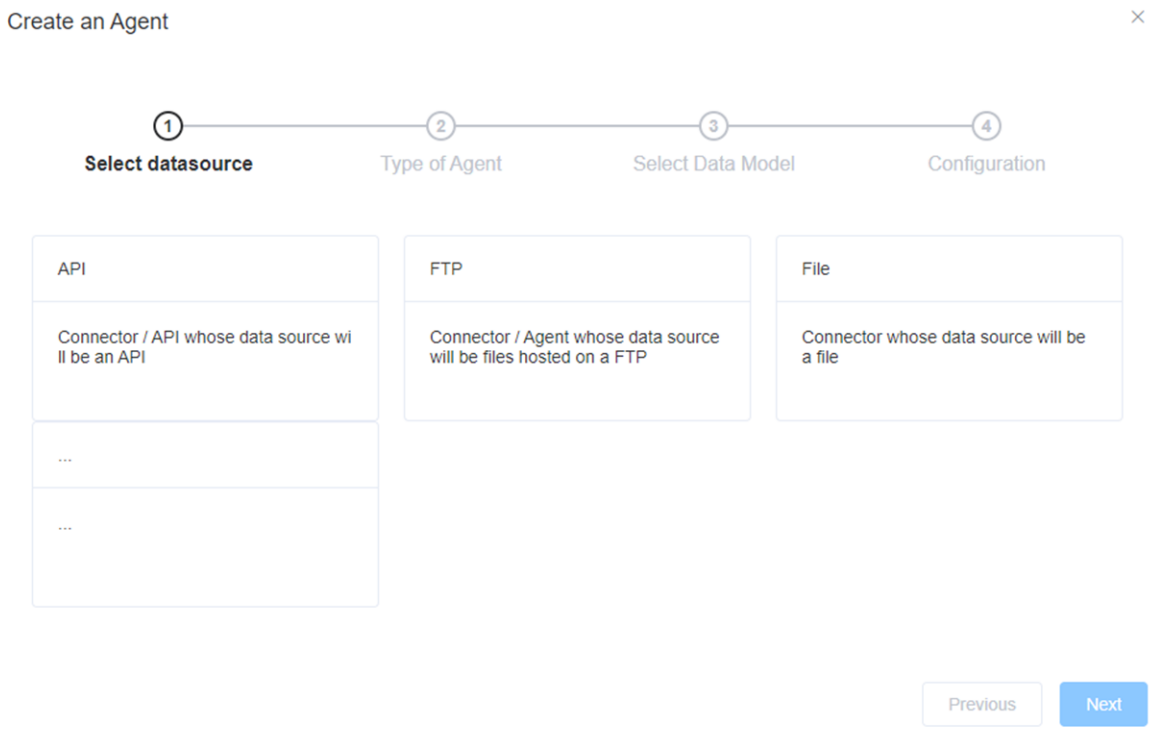

Figure 17 - Example of Wizard for creating agents for different data sources

6.2 SEMANTIC INTEROPERABILITY COMPONENT

The DataPorts platform will guarantee semantic interoperability in order to provide a unified virtualized view of the data for its use by the different Data Consumers and the Data Analytic services. DataPorts Project is developing a Semantic Component to describe ports data together with mappings to standard vocabularies in order to simplify the reuse of data applications for analytics and forecasting. This semantic component will codify domain knowledge of the domain experts, and thus can be reused and exploited by the data experts directly, thereby empowering building cognitive port applications.

The Semantic Interoperability Component exposes a unified API to access the data from the different data sources connected to the DataPorts platform, providing both real-time and batch historical data to the data consumers. In collaboration with the Data Access Component, it provides a data shared semantics solution for the interoperability of diverse data sources using a common ontology, which is based on elements from existing ontologies. In order to ensure that the data consumer can understand the structure and meaning of the data, output data follow the common NGSI format and the DataPorts ontology, regardless of the data source that generates it.

Regarding the metadata, the Semantic Interoperability Component obtains information about the different data sources from the Data Access Manager and stores it in a Metadata Registry to make it available for the other subcomponents of this component. In addition, the metadata is sent to the IDS broker to provide information to the data consumers about the available data sources.

6.2.1 Position in the Architecture

The Semantic Interoperability Component input data is gathered from the Data Access component, so this component is just on top of the second one. Any component that makes use of the data from the data sources, such as the Data Abstraction and Virtualization Component (DAV), the Automatic Model Training Engine or the Process-based Analytics, will use the Semantic Interoperability Component data as input. Hence, the Semantic Interoperability Component is positioned below the DAV and the Data Analytics components.

In addition, it is below a Blockchain client which is connected to the Semantic Interoperability Component in order to facilitate the communication with the Blockchain-Data Governance Component and the IDS Broker.

6.2.2 Contribution to the Platform and the Platform's Objectives

The Semantic Interoperability Component provides semantic interoperability by offering a common API to access the data from the data sources and defining a common meaning and format for the data. This component works together with the Data Access Component to enable a common way to access the data from the connected data sources, which allows the upper components of the platform to make use of the data without having to deal with a variety of interfaces, data formats and semantics. In addition, the Semantic Interoperability Component stores locally the metadata from the data sources connected to the instance of the DataPorts platform and sends it to the IDS broker to make it available for the other components of the platform as well as other possible client applications and services following the rules defined in the Data Governance Component. The upper components will make use of the metadata to list the available data sources and select their input data.

The Data Abstraction and Virtualization component makes use of the Semantic Interoperability component API to obtain que historical data from the data sources. The historical data is provided following the DataPorts data model and its conformity to the data model is validated before sending the data.

The Automatic Model Training Engine and the Process-Based Analytics obtain the current values of the data from the Semantic Interoperability component and use them as input for the models once they have been trained with the historical data. To obtain the input data for the models, these components can request the last values of the data to the Semantic Interoperability component or subscribe to the data of interest.

In addition to the upper components of the DataPorts platform, other applications and services (either existing or new) can make use of the API provided by the Semantic Interoperability component to obtain data in the common DataPorts format and semantics.

Finally, the Semantic Interoperability component can also be used by other organizations to access the data through IDS connectors if the proper permissions have been granted.

Regarding the DataPorts Data Model, it defines a common meaning for the data managed by the DataPorts platform and will facilitate the creation of new agents to connect new data sources, as well as applications and services able to make use of the data obtained from the platform.

6.2.3 Addressed Requirements

Table 10 shows the requirements that are addressed by the Semantic Interoperability Component:

ID	Description	Category
R3.5	The DataPorts platform must provide an ontology that describes the concepts from the data models of all the connected data sources so that all the input data can be represented following a common ontology.	Interoperability

ID	Description	Category
R3.7	The DataPorts platform must provide publish/subscribe operations to internal components in order to provide access to business real-time data from the connected sources	Interoperability
R3.10	The Semantic Interoperability Layer must support semantic modelling in order to provide an unambiguous meaning of the data.	Interoperability
R3.23	The components of the DataPorts platform could be virtualized in order to ease its deployment and portability.	Virtualization, Scalability
R3.24	The DataPorts platform could offer mechanisms for the automatic deployment, maintenance and scaling of the developed components.	Virtualization, Scalability
R3.32	As a data provider I want to share data with the DataPorts platform so it is available for the platform users with the access rights.	Functionality, Interoperability
R3.33	As a data consumer, I want to get the list of the available data sources and all the methods provided by the platform to subscribe or request data on demand.	Functionality, Interoperability
R3.34	As a data consumer, I want to subscribe to an available subscription provided by the DataPorts Platform.	Functionality, Interoperability
R3.35	As a data consumer, I want to be able to cancel a current subscription, so that I stop receiving data modifications.	Functionality, Interoperability
R3.36	As a data consumer, I want to request data on demand from the data sources using the methods provided by the DataPorts Platform.	Functionality, Interoperability
R3.37	The DataPorts Platform could reduce and process the data on the ports' side before they reach the Data Abstraction and Virtualization component's repository (Virtual Data Repository – VDR) and become available to distributed resources.	Performance

Table 10 - Addressed requirements of Semantic Interoperability component

6.2.4 Description of the Component

The Semantic Interoperability Component provides a repository with the DataPorts Data Model and DataPorts Ontology. It offers an ontology definition using OWL[7] and a Data Model description using JSON schema and JSON-LD context documents. The aim is to integrate the following ontologies and data models: Fiware Data Models[8], IDS Information Model[9], UN/CEFACT[10] model and SAREF[11] ontology.

In addition, the Semantic Interoperability Component acts as a middleware layer between the data sources and the Data Abstraction and Virtualization component. It receives the information from the Data Access Component in a common data model and interacts with this component through a Semantic Broker. The functional blocks that compose this component are shown in Figure 18.

[7] https://www.w3.org/TR/owl-guide/

[8] https://www.fiware.org/developers/data-models/

[9] https://github.com/International-Data-Spaces-Association/InformationModel

[10] https://umm-dev.org/about-umm/

[11] https://saref.etsi.org/

The component performs the following actions:

- Manages and stores locally the information regarding the connected data sources. Data sources (and their corresponding agents in the Data Access Component) are registered locally in the Metadata Registry. This way, the metadata is available for all the subcomponents of this component.
- The Metadata Registry offers the description of the data sources available for the internal components of the DataPorts Platform.
- Each local instance of the Semantic Interoperability Component registers in the IDS broker the data sources that it manages in order to make the description of the data sources available for the data consumers from other organizations.
- Provides an interface for publishing and subscribing streams of data.
- Provides data on demand. This includes both historical data and last values.
- Provides REST API interaction with Linked Data.
- Validates that the data sent to the data consumers is in the correct format.
- Distributes the data to the components of the upper layers, such as the DAV component, the Automatic Model TrainingEngine and the Process-based Analytics.

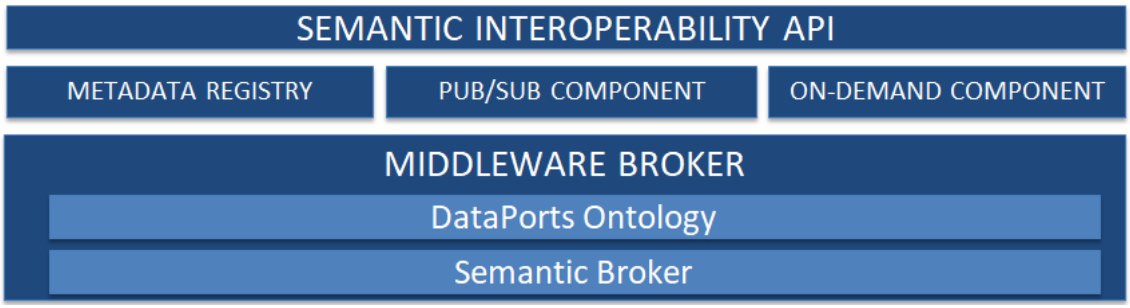

Figure 18 - Semantic Interoperability Component

The main methods that are going to be provided by the Semantic Interoperability API are the following:

- Related with the Metadata Registry
  - Create/delete/update data source
  - Obtain data source information
  - List data sources/ Query by data source identification / Query by metadata
  - Obtain agents description
  - Retrieve data source attributes
  - List data types
- Related with the Publish-Subscribe Component
  - Create/delete/update subscription
  - List subscriptions
  - Retrieve subscription information
- Related with the On-demand Component
  - Get historical data
  - Get last values

Table 11 shows the list of functionalities that are implemented by the Semantic Interoperability Component:

ID	Description	Captured by requirement
F-3.5	The DataPorts platform enables the federation of data varying in syntax and semantics	R3.5, R3.10, R3.33, R3.34, R3.36

ID	Description	Captured by requirement
F-3.7	The DataPorts platform provides semantic stream processing	R3.25, R3.37
F-3.15	The DataPorts platform processes streams of records and publish and subscribe to streams of data	R3.7
F-3.16	The DataPorts Platform provides a framework for semantic interoperability from several sources	R3.1, R3.10
F-3.17	The DataPorts Platforms offers an ontology to guarantee semantic interoperability	R3.10
F-3.18	The DataPorts Platform provides REST-style interaction with Linked Data	R3.5, R3.10
F-3.19	The DataPorts Platform provides a data source metadata registry	R3.33
F-3.20	The DataPorts Platform provides data from a federated database on demand	R3.32, R.3.36
F-3.21	The DataPorts Platform provides data from a federated database through publish and subscribe model	R3.33, R3.34, R3.35

Table 11 - Functionalities implemented by Semantic Interoperability Component

6.2.5 Interactions of the Component

As can be seen in Figure 19, the Semantic Interoperability Component works with the data sources registered in the Metadata Registry and the data provided by the Data Access Component. Then, data is distributed to the Data Abstraction and Virtualization Component, the Automatic Model Training Engine and the Process-based Analytics. Communication between components is performed through NGSI messages normalized and translated to the supported ontology. More concretely, the Automatic Model Training Engine and the Process-based Analytics will mainly make use of the subscriptions to obtain the input data for the methods and the Data Abstraction and Virtualization is expected to retrieve data on demand through the Semantic Interoperability component API.

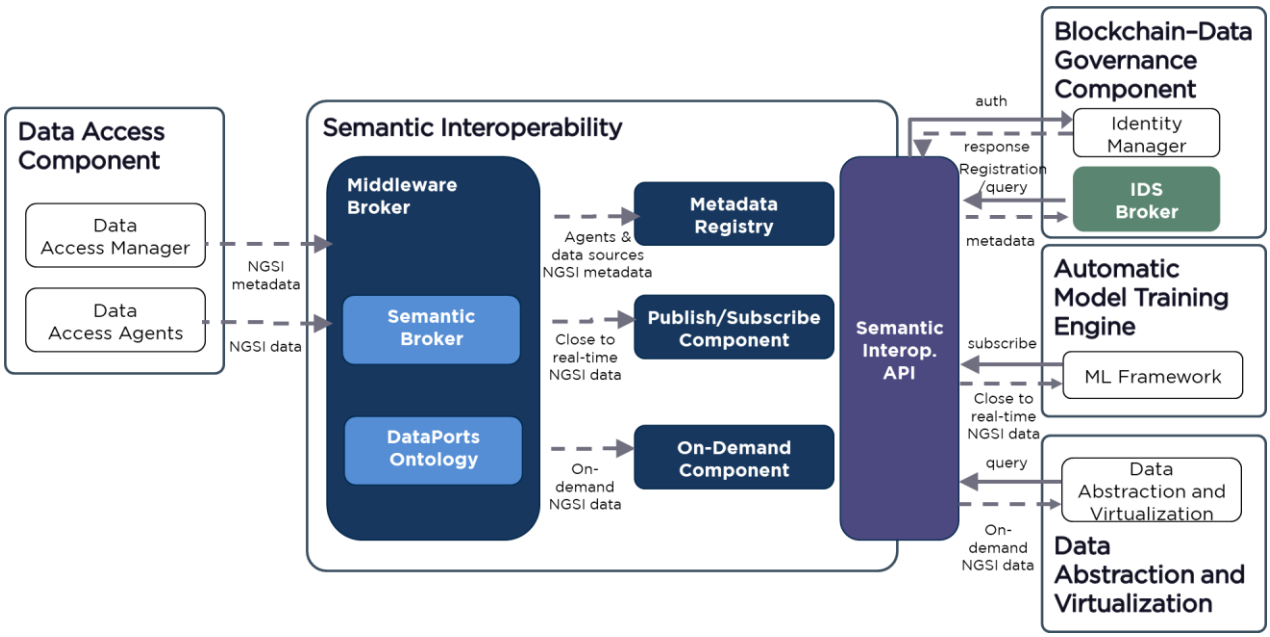

Figure 19 - Interactions of Semantic Interoperability Component

Regarding the metadata, the Semantic Interoperability Component obtains information about the different data sources from the Data Access Component and stores it in the Metadata Registry to make it available for the other subcomponents of this layer. The metadata indicates the data types provided by the data source, the available attributes, the type of agents that are associated with the data source (publish/subscribe or on-demand) and whether the data is stored on Blockchain or not. Then, the metadata is sent to the IDS Broker to provide information to the data consumers from other organizations about the available data sources.

Finally, the Semantic Interoperability Component API interacts with the Data Governance components to enforce the data access policies, in order to ensure proper data protection in the exchanges with the data consumers, as well as with the Identity Manager to enable authentication.

The communication with the Blockchain-Data Governance Component and the IDS Broker is done through a Blockchain client and IDS connector, which is connected to the Semantic Interoperability API.

6.2.6 Subcomponents / Tools

The Semantic Interoperability Component is composed of the following subcomponents (Figure 18):

- Middleware Broker:
  - Semantic Broker: Broker to receive information and interact with the Data Access component
  - DataPorts Ontology: repository with the Ontology and the Data Model description.
- Metadata Registry: Retrieves the information about the available data sources and semantics. Includes information about agents' features.
- Publish/Subscribe Component: Manages the subscriptions with the data sources and enables access to real-time data, interacting with the agents implemented as publish/subscribe.
- On-demand Component: Manages the access to data on demand (for example, historical batch data queries), interacting with the agents implemented as on-demand.
- Semantic Interoperability API: Common API to access to the Semantic Interoperability Component functionalities. It is implemented as an API Gateway in order to facilitate the integration with other components like Blockchain - Data Governance Component.

The main subcomponent of this component is the Middleware Broker, which is composed by the Semantic Broker and a repository for the ontology. The Semantic Broker receives the data from the agents and sends it to the other components that act as data consumers, while the ontology defines the semantics of the data to ensure that all the data consumers will be able to understand its meaning.

Regarding the implementation, the Orion Context Broker[12] provides the functions of the Semantic Broker, Metadata Registry and a Publish/Subscribe component interface. Orion receives the data from the agents in the Data Access Component in NGSI format, keeps a registry of the connected data sources and sends notifications to the data consumers in the upper components. Orion Context Broker was developed as a part of the FIWARE platform[13]. Orion is an implementation of the Publish / Subscribe Context Broker GE, providing an NGSI interface[14]. Orion allows the user to manage the whole lifecycle of context information including updates, queries, registrations and subscriptions.

Since the Orion Context Broker only stores the latest value of each entity, a custom on-demand module has been included. This module provides a query API to retrieve historical data and communicates with the Orion Context Broker to retrieve information about the data sources and data model.

[12] https://fiware-orion.readthedocs.io/en/master/

[13] https://www.fiware.org/

[14] https://fiware.github.io/specifications/ngsiv2/stable/

To ensure that the data sent to the data consumers follows the common data model, a data model validation module will be included.

Finally, a unified API has been included in this component. This API exposes the functions of the Orion Context Broker and the on-demand module (which have been described in Section 6.2.4) and will be integrated with the Data Governance component.

Additionally, some extra tools may be used with the Semantic Interoperability Component to facilitate its management and the integration with Blockchain. If necessary, the inclusion of extra subcomponents to enable interoperability with other platforms will be considered, although the preferred way of interaction in this case is the use of the Data Access Component to connect data sources to the Semantic Interoperability Component and the use of the Semantic Interoperability Component API to retrieve data.

6.3 DATA ABSTRACTION AND VIRTUALIZATION COMPONENT

The Data Abstraction and Virtualization (DAV) component aims at properly preparing the data inputs from various sources inside the generic DataPorts architecture, keeping metadata from all the feeds/streams and finally exporting the "cleaned"/processed datasets through exposed RESTful APIs, thus making them available to any potential client. Persistent data (that is, already stored and existing data) streams shall be the basic source of load for the DAV component, which is thoroughly described in the following subsections.

6.3.1 Position in the Architecture

For the DAV component to function properly, it must be positioned after any possible data input / provider / stream. This means that the component must be placed after the Data Access & Semantic Interoperability components, as well as any possible data provider that "lives" outside the two components. However, since DAV provides the processed data through exposed APIs, by its nature it must be positioned before the Automatic Models Training Engine, as well as any other potential data recipient. In conclusion, DAV is an intermediary between the main data providers and consumers (playing the role of a data provider-consumer itself) of DataPorts architecture, as it gains persistent data input (mainly from the Semantic Interoperability component) and proceeds to forward these data streams to any interested recipient (mainly the Automatic Models Training Engine).

6.3.2 Contribution to the Platform and the Platform's Objectives

The Data Abstraction & Virtualization component plays an integral role within the DataPorts Platform. Since it acts as an intermediary between data providers and recipients, DAV is responsible for properly preparing and maintaining data for as long as necessary. DAV is the main entity when it comes to data cleaning, storing and distributing inside the DataPorts Platform. Therefore, its existence in the platform seems imperative. Without it, incoming data streams would enter the platform without being properly cleaned and filtered. Moreover, they would be lost in case the recipient did not respond immediately in the incoming data "call". Furthermore, no additional processing would take place in these data streams. DAV's VDC subcomponent forwards the requested data, after processing specific request parameters. It then forwards them in a desired format (such as Apache Parquet).

By its nature as an internal component, DAV is implemented in such a way that it works in the "background", meaning that no user interface is required for the interaction with it. The integration with data providers and recipients takes place through strong and consistent API connections. In addition, DAV is a generic component, meaning that it can apply its services in different kinds, types and formats of datasets. It can also perform complex tasks upon requests, in order to extract useful analytics or metadata from the stored datasets, or simply provide the recipient with information from different datasets combined. Again, its ability to forward the requested data to a desired format can be modified, enabling many kinds of data formats (such as Microsoft Excel, CSV, JSON, Apache Parquet, TXT, etc.). DAV's generic nature makes it capable of performing as a standalone component in any use case scenario that involves the movement of (big) data. As an intermediary (as already stated), its goal is to maintain the project's high data quality standards under

any circumstances. Therefore, DAV is a solid performer across multiple environments, both in and out of DataPorts.

Data Abstraction & Virtualization's key role within the DataPorts architecture is also evident by the specifications of some of DataPorts' use cases. More specifically, in a use case defined by Valencia Port (named "Port Authority Data Sharing and Analytics Services ") [3], a component is needed in order to achieve data aggregation for the incoming datasets. DAV's Virtual Data Container (VDC) subcomponent is more than capable of performing data aggregation to datasets with numerical attributes. As already stated, VDC can handle even more complex tasks than aggregation techniques on datasets. Therefore, more metadata could be extracted for each incoming data stream. In any case, Valencia Port's need for data aggregation services is covered by DAV.

Apart from that, the same use case by Valencia Port highlights the need to provide analytical services to other companies operating in the port (or other members of the platform interested). DAV's VDC subcomponent can extract basic analytics from the stored datasets in order to enrich the information collected by other companies within the port network. However, in the context of DataPorts, the Data Analytics and AI Services components are responsible for generating advanced analytics from DataPorts' datasets, which could be used by the companies. Nevertheless, for the Automatic Models Training Engine to function properly, quality-of-data of the datasets must be assured. This is where DAV's services work, as the missing link between the Data Analytics and AI Services components and the Valencia Port's need for analytics. In conclusion, not only DAV can generate analytics from the existing datasets, but its cleaning & filtering tasks achieve high quality of these datasets, before forwarding them to the analytics components.

6.3.3 Addressed Requirements

Table 12 depicts the list of requirements that are addressed by the Data Abstraction and Virtualization component:

ID	Description	Category
R3.18	The Data Abstraction and Virtualization component must have an open API in order for big vendors as well as new providers to be able to publish their services and components	Interoperability
R3.19	The Data Abstraction and Virtualization component must be data-source independent	Interoperability
R3.20	The data model specification should follow the semi-structured format	Interoperability
R3.21	The notation language must be able to be parsed by multiple different programming languages	Compatibility
R3.22	The notation language should be human readable, easy to script and to understand	Interoperability
R3.41	The DataPorts Platform should have (efficient) provisions for checking data quality (e.g., to detect concept drift, missing data, inconsistent data etc.)	Quality
R3.42	The DataPorts Platform should deliver cleansed, integrated etc. data to the analytics services	Quality
R3.44	Data correlation. The data from different sources should be correlated (virtual object) or in the same process	Interoperability
R3.46	Data quality. Before correlate data from different sources, quality should be checked to guarantee the correct interpretation	Quality

Table 12 - Addressed requirements of Data Abstraction and Virtualization component

6.3.4 Description of the Component

The Data Abstraction and Virtualization component can be viewed as a wrapper of smaller sub-components / tools that, combined, achieve the predefined goal. We divide the component in three main parts: The first

part consists of the initial processing of the incoming datasets, named “Processing and Filtering Software” (PaFS). The second part plays the role of the repository for all the existing datasets that have made their way in DAV, named “Virtual Data Repository” (VDR). It can be seen as a data lake, containing all the pre-processed datasets. The third part is the key “connector” between DAV and any potential data recipient, named “Virtual Data Container” (VDC). It further processes the data, based on the recipient’s request. Moreover, upon the dataset request by a recipient, it transforms the data into the desired format, whilst making them available to him/her. The complete architecture is presented in Figure 20. Let us further analyse these sub-components:

First and foremost, we begin with the initial DAV’s sub-component, the Processing and Filtering Software (PaFS). Any incoming dataset passes through PaFS first, in order then to be stored in the Virtual Data Repository (which is described later on). Once again, it is worth noting that the main source of data shall be the Semantic Interoperability components of T3.2, and more precisely its On-Demand sub-component. As soon as a complete data stream has entered DAV, it becomes subject of cleaning, pre-processing and filtering. PaFS makes sure that the dataset does not contain any missing / dirty values (or zero ones), errors, wrong value formats, or any other attribute that could lead to problems in the future. In a few words, PaFS properly prepares the dataset for the VDR, where it will be stored, waiting for retrieval (which shall take place through VDC). Furthermore, some basic metadata will be drawn, regarding the dataset and its contents. The metadata will be generated in cases when the incoming datasets do not already contain them (or some of them). It is safe to assume that PaFS plays a vital role within DAV, since it prepares the data for any future use. All future recipients will lie in PaFS to have the datasets cleaned, filtered and ready.

Moving on to the second part of DAV, it is the Virtual Data Repository, shortly referred to as VDR. As mentioned before, VDR is the sub-component storing all the datasets that pass-through DAV. After PaFS has processed each data stream, it then enters the VDR. We could present VDR as a data lake, containing multiple data from various sources. In addition, it is worth repeating that DAV shall deal only with persistent (that is, already existing) datasets, which means that VDR stores only cleaned and filtered historical data on-demand. Any recipient who requests data, will actually draw instances (or ‘Data Ponds’) of VDR’s data lake, which are made available through VDC. This flow is based on an already proposed Information Life Cycle Management Framework [5].

The last sub-component of DAV is the Virtual Data Container (VDC). VDC is an abstraction layer between the data consumers (e.g., data-intensive applications) and the VDR. It serves the data according to the application requirements and constraints in terms of quality and format and hides the complexity of the underlying infrastructure (VDR), which cannot be accessed directly from external components, but only from DAV modules that run inside a Kubernetes cluster. More specifically, any instance of the VDC achieves two critical tasks: i) it further processes the datasets, which leads to analytics and metadata extraction. These metadata vary from basic ones such as means, sums, variances and standard deviations, to more complex tasks, such as the combination of data from different datasets or the extraction of deeper analytics, and ii) it transforms the data in a format, suitable for the data recipient. For example, Automatic Models Training Engine needs to obtain the data in Parquet format. Consequently, -in this case- the data transformation task focuses on the conversion to Parquet files, from the existing formats, which shall be in JSON / JSON-LD. The output of the VDC is a data pond, meaning a processed subset of the VDR’s data lake, which focuses on a particular subject, fitted to the purpose of the application and without redundancy.

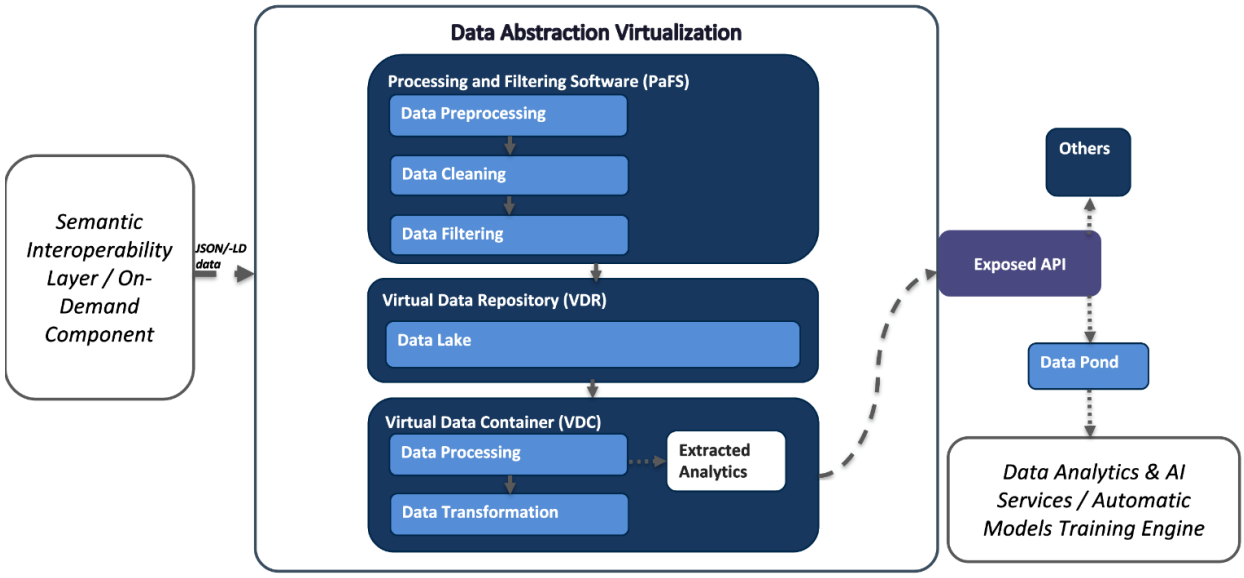

Figure 20 - Data Abstraction & Virtualization component's architecture

Table 13 depicts the list of functionalities that are partially or totally implemented inside the Data Abstraction & Virtualization component:

ID	Description	Captured by requirement
F-2.2	The DataPorts Platform sets a data driven ecosystem ready for a comprehensive exploitation of data, and virtual data repositories	R3.18, R3.19, R3.20, R3.21, R3.22
F-3.3	The DataPorts Platform provides data sanitization algorithm to guarantee data integrity	R3.41, R3.42, R3.44, R3.46
F-3.9	The DataPorts Platform provides declarative, distributed data aggregation	R3.41, R3.42, R3.44, R3.46
F-3.14	The DataPorts Platform provides smart API for cognitive services	R3.18

Table 13 - Functionalities implemented by Data Abstraction and Virtualization component

6.3.5 Interactions of the Component

As mentioned before, the Data Abstraction & Virtualization component's interactions consist mainly of the links between itself and the Semantic Interoperability component (that is, the On-Demand component, DAV's main data source/input), as well as any other data recipient, as seen in Figure 21. Incoming datasets are structured in JSON/JSON-LD format, for them to be properly received by DAV's data retrieval tools. The main data recipient of DAV is the Data Analytics & AI Services' Automatic Models Training Engine. According to the DAV's architecture, shown in the flow above (but also in Figure 21), other possible data recipients have access to the processed data through an exposed RESTful API, collecting them as data ponds. Concerning the Automatic Models Training Engine, the data ponds' selected format is Parquet. Once again, it is safe to conclude that DAV plays the role of an intermediary within the DataPorts architecture, since it gains persistent data input from the Semantic Interoperability component but serves as a source itself (for the processed data streams) to other recipients, mainly the Data Analytics & AI Services components. Moreover, as an intermediary, DAV shall also generate basic metadata (sums, means, variances etc.) and useful analytics from any dataset that finds its way through it.

The position of DAV in the DataPorts architecture, as well as its interaction with the components is shown in the figure below:

Figure 21 - Interactions of Data Abstraction and Virtualization component

It is also worth noting that DAV's sub-components communicate and interact internally (through the infrastructure in which they are implemented), which means that no external access is allowed. Only the Virtual Data Container (VDC) is available for external communication.

6.3.6 Subcomponents / Tools

As already mentioned, and described, DAV consists of three sub-components, which "live" inside a Kubernetes cluster.

The Processing and Filtering Software (PaFS) is DAV's first sub-component and is implemented using a big data appropriate framework, co-operating with Kubernetes. It consists of:

- Data Pre-processing module: it prepares the incoming dataset for the following Cleaning and Filtering modules. Here, the incoming dataset is "identified" (based on its type) and transformed from its original format. According to its type identification, different cleaning and filtering techniques might apply to it. Any potential metadata retrieved along with the dataset, are also evaluated here.
- Data Cleaning module: it parses the incoming dataset and detects wrong/zero/null/non-existing values. It also detects false format values. In a few words, this module makes sure that all the inconsistencies of the dataset are detected
- Data Filtering module: it fills the null / non-existing values with means from the results generated above. Different scenarios exist for the cases of categorical null / non-existing values or occurrences with too many nulls (where means extraction reduces the dataset's quality). This way the dataset does not bleed from dirty values. As soon as the dataset exits from the Data Filtering module, it is ready to be stored in the DAV's data lake, meaning the VDR.

The Virtual Data Repository (VDR) is the second sub-component. It is the DAV's data lake. All the incoming, pre-processed and cleaned datasets (from PaFS) are stored there. Regarding the database framework that is used, after considering multiple alternatives, MongoDB was selected. There have been tweaks and modifications in the default MongoDB configuration, for VDR to be an efficient and load-manageable data lake. Any recipient that wishes to obtain the processed and cleaned datasets stored in the VDR, will have to do so through the VDC.

The Virtual Data Container (VDC) is DAV's third and final sub-component, implemented using - again – a big data appropriate framework under Kubernetes' umbrella. VDC contains the following modules:

- The Data Processing module: it is implemented using widely adopted processing methods. Based on the request made by the data recipient, the Data Processing module enables communication with VDR, in order to obtain the data needed. Computational tasks upon the data retrieved from querying VDR might take place, always depending on the client's request.
- The Data Transformation module: it transforms a dataset to the desired format (e.g., Parquet) and prepares it to be forwarded as a response to the client's request.

As already mentioned, since DAV is considered an internal component of the DataPorts Platform, there is no need for a GUI to interact with. Developers shall interact with the component via an exposed RESTful API.

6.4 DATA GOVERNANCE COMPONENT

Data Governance is the mechanism of data, monitoring its complete lifecycle. It increases consistency and confidence of the data registered, improving data security, and minimizing the risk of not complying with relevant regulation.

The Data Governance component has two main functionalities that consist of sharing data between the participants in a decentralized way through a Blockchain network and storing evidence of the transactions carried out in the system, providing it with immutability and being a source of auditability of the operations carried out between the data providers and data consumers.

On the other hand, it establishes sharing rules through smart contracts for each data set. Smart contracts will allow or deny the consumption of data sets based on compliance with the privacy policies defined in smart contracts. Actions carried out by the participants are stored in the Blockchain after the smart contract evaluation is successful, obtaining an auditable and immutable record of the data sets provided and consumed by each organization to allow for environment of trusted data exchange. The smart contracts will enable, revoke, or deny the consumption of data sets based on compliance with the privacy policies defined in the smart contracts. The Data governance component should also include special access policies such as requiring special permission from more than one party to access the data.

The inclusion of a Blockchain network in DataPorts for data governance purposes aims to provide a comprehensive framework in which data ownership and data distribution policies become a relevant part of the entire DataPorts platform. It should provide various governance capabilities and interoperability among different platforms, establishing the rules for data consumption in DataPorts. To this effect, end users should be able to define access policies for their datasets in a streamlined way without needing programming experience.

For more information about this topic, refer to Section 6 in Deliverable D2.3 [6].

6.4.1 Position in the Architecture

The Blockchain network and the smart contracts that evaluate the access rules and grant permission to access the data are part of the Data Governance component, which has its point of access from the connectors of the data owners and data consumers through API REST that makes use of the Hyperledger Fabric SDK for network access.

Within the platform architecture, the Data Access Component and the Semantic Interoperability Component work in conjunction with the Data Governance Component, providing the data and semantic rules to be incorporated into the smart contracts and the metadata written into the ledger itself. The Data Governance component aims to register, evaluate, and enforce access rules for the datasets registered in DataPorts and, to this purpose, communication between the components must be achieved. In this querying and registration process that takes place in the platform, the Blockchain is the ultimate source of truth, and every operation regarding data governance ends with a transaction evaluated by the data governance smart contracts, that will allow or deny the registration or consumption of the desired datasets.

6.4.2 Contribution to the Platform and the Platform's Objectives

The Data Governance component provides the platform with the mechanisms to define and manage the access rules policies to the different data sets managed by the platform and made available for exchange or consumption between data providers and data consumers.

The Data Governance component is implemented through a permissioned Blockchain network, and the four main subcomponents that are implemented in this Blockchain network through smart contracts are: Clearing House, IDS Broker, Identity Management and governance rules.

- Broker chaincode allows to register all metadata pertaining to the dataset that a data provider wishes to share, and that are stored on its premises

- Governance rules chaincodes allow to register access rules, to define who can access the datasets published in the broker and how, and to evaluate these access rules once a particular data consumer wants to access the dataset.
- The clearing house, whose function is to enable provenance and auditing of all requests for access to granted datasets/permissions, is implemented using the inherent functionality of the Blockchain.
- Identity management is inherent in any permissioned Blockchain implementation. Each Blockchain organization defines a Certificate Authority (CA) server, which is responsible for issuing cryptographic material for all entities in the network, including nodes (peers and orderers), and network clients (applications that connect to Blockchain with the help of Fabric SDK, or end users, who authenticate, connect and execute transactions with the help of these applications).

The data Governance service works in conjunction with the Semantic Interoperability component that registers the dataset metadata in the broker. The various data consuming services and components query the governance smart contracts from the connectors to determine their level of permission to the data.

The access point from the connectors to the data owners and consumers is through an exposed REST API that encapsulates the Hyperledger Fabric SDK for network access.

The Blockchain network is a generic component and part of the overall platform. In fact, it can be applied and replicated in any domain in which data is off-chain and Blockchain serves as a means for defining and enforcing access policies.

6.4.3 Addressed Requirements

The following table (Table 14) details the set of requirements that have been identified for proper development of the platform:

ID	Description	Category
R2.2	The DataPorts Platform must provide a secure interface framework for data exchange between itself and the potential data sources	Non-Functional
R2.4	The architecture must follow a federated data approach avoiding the “data-centric model”	Non-Functional
R2.7	The DataPorts Platform must provide resources (user guides, documentation, etc.) to support the provided functionalities of the platform	Non-Functional
R3.11	As a Blockchain organization I want to be able to interconnect with other organizations within my network in an agreed upon manner to facilitate trust and transparency	Non-Functional
R3.12	As a blockchain organization I would like to be able to onboard an existing Blockchain network and participate as a validating member	Functional
R3.13	As a participating Blockchain organization I would like to be able to agree on a consensus algorithm which will provide pilot appropriate transaction validation and ordering functionality for the network	Functional
R3.14	As a participating Blockchain organization I would like to be able to securely sign in my network components (peers, Cas) and clients (end-users/application clients) to a blockchain network	Non-Functional
R3.15	As a participating Blockchain organization I would like to be able to maintain my own copy of distributed ledger while being assured other organizations on the network have exact same replica of the ledger.	Functional

R3.16	As a participating Blockchain organization I would like to be able to endorse transactions by executing agreed upon business logic specified as a bounding contract between network organizations	Functional
R3.17	As a participating Blockchain organization I would like to be assured of immutability and permanent availability of the shared data	Non-Functional
R3.39	As a data provider, I want to retrieve the history of transactions, so that I am able to track the information flow	Functional
R4.1	As a participating Blockchain organization, I would like to be able to develop and run agreed upon business logic in a distributed manner and agree on results	Non-Functional
R4.2	As a participating Blockchain organization, I would like to share data in a privacy preserving manner with selected organizations within the business network	Non-Functional
R4.3	As a participating Blockchain organization, I would like to enroll my client applications and end users to access and share data in my Blockchain network in a secure manner using appropriate authentication	Non-Functional
R4.5	As a data provider, I would like to have an agreement with data consumer about data manipulation	Non-Functional
R4.6	As a data provider, I would like not only to grant access to data consumer, but also to revoke access	Non-Functional
R4.7	As a participant Blockchain organization, I want to set data governance rules, so that data access is specified	Non-Functional
R4.12	The DataPorts Platform must support the security and governance recommendations from the IDS reference architecture in order to share and transfer data securely	Non-Functional

Table 14 - Addressed requirements by the Data Governance component

6.4.4 Description of the Component

In order to fulfill the necessary requirements and incorporate overall platform functionalities, a Blockchain platform comprised of three separate networks is proposed. Each pilot will have their own network to test and deploy their use cases applications in the Blockchain. A third shared network, called the Blockchain Governance Network, will be common to all participants in the Blockchain platform and will contain smart contracts for global use cases and global scenarios. The data sharing and data governance components will be contained in the smart contracts, giving permission for access and consumption of datasets. For more information on the Hyperledger Fabric network design, refer to Chapter 7.2 in Deliverable D2.3 [6].

The Data Governance component was the focus of the MVP to be deployed in the Blockchain Governance Network in Q2 2021. The component will contain a REST API as well as a Fabric Client to communicate with the Blockchain network from other components of the DataPorts Architecture. Certain components and characteristics will be provided by the DataPorts platform, such as the Data Access Component, the Identity Management Component, or the Semantic Interoperability Component.

The Data Governance component of the DataPorts platform should meet the following expectations in order to comply with the previous requirements, detailed in Table 15:

ID	Description	Captured by requirement
F-2.4	The DataPorts platform (through Blockchain technology) implements all the authentication and authorization mechanisms to allow data sharing and trading in a secure and reliable way	See Section 5.3

ID	Description	Captured by requirement
F-2.6	The DataPorts platform provides Orion Context Broker and Blockchain component, registering the description of the data	See Section 6.2
F-2.7	The DataPorts platform enables the data owners to exchange data	See Section 5.1
F-2.8	The DataPorts platform enables data sovereignty	See Section 5.1
F-4.4	The DataPorts platform provides clear rules on how data will be accessed	R4.3, R4.4, R4.6
F-4.5	The DataPorts platform provides a flexibility of policies on data distribution	R4.5, R4.7
F-4.7	The DataPorts platform enables the data owners to exchange data	R4.5, R4.6, R4.7, R4.9, R4.11, R4.16

Table 15 - The functionalities implemented by the Data Governance component

6.4.5 Interactions of the Component

Interaction between components in the DataPorts platform must be achieved, and the Blockchain component should be able to communicate events to the necessary components. To be precise, most interactions with the Data Governance component will take place with the Semantic Interoperability Component, to take advantage of the platform ontology and metadata registration, and with the Data Access Component, in order to manage transaction proposals to be validated by the smart contracts in the Blockchain Governance Network.

The Semantic Interoperability Component will provide the necessary specifications for metadata and data querying and registration, so that future implementations and modules can be compatible with existing components with little friction between them. This standardization process is necessary for the Data Governance Component to achieve consistency of the registered metadata, under a common format and framework.

The Data Access Component will serve as the link between the Blockchain Governance Network and the rest of the DataPorts platform, requesting access to metadata and, when successful, broadcasting the results to the relevant component that requested the blockchain transaction. Once the Blockchain peers in the Governance Network receive a transaction proposal for accessing a dataset, the smart contracts will evaluate the petition and allow or deny the consumption of the dataset according to the governance rules provided by the dataset owner. These interactions between components in the main platform architecture are shown in Figure 22.

Authentication between on-chain and off-chain components is achieved by assigning a unique identity on the Blockchain to every user in the DataPorts platform that needs to make use of the Data Governance component. Hyperledger Fabric Membership Service Providers (MSP) are the mechanisms that allow identification of components and users in the Blockchain, and therefore, every user must be associated with the relevant MSP to certify that the current user is allowed to perform operations and make transaction proposals in the Blockchain.

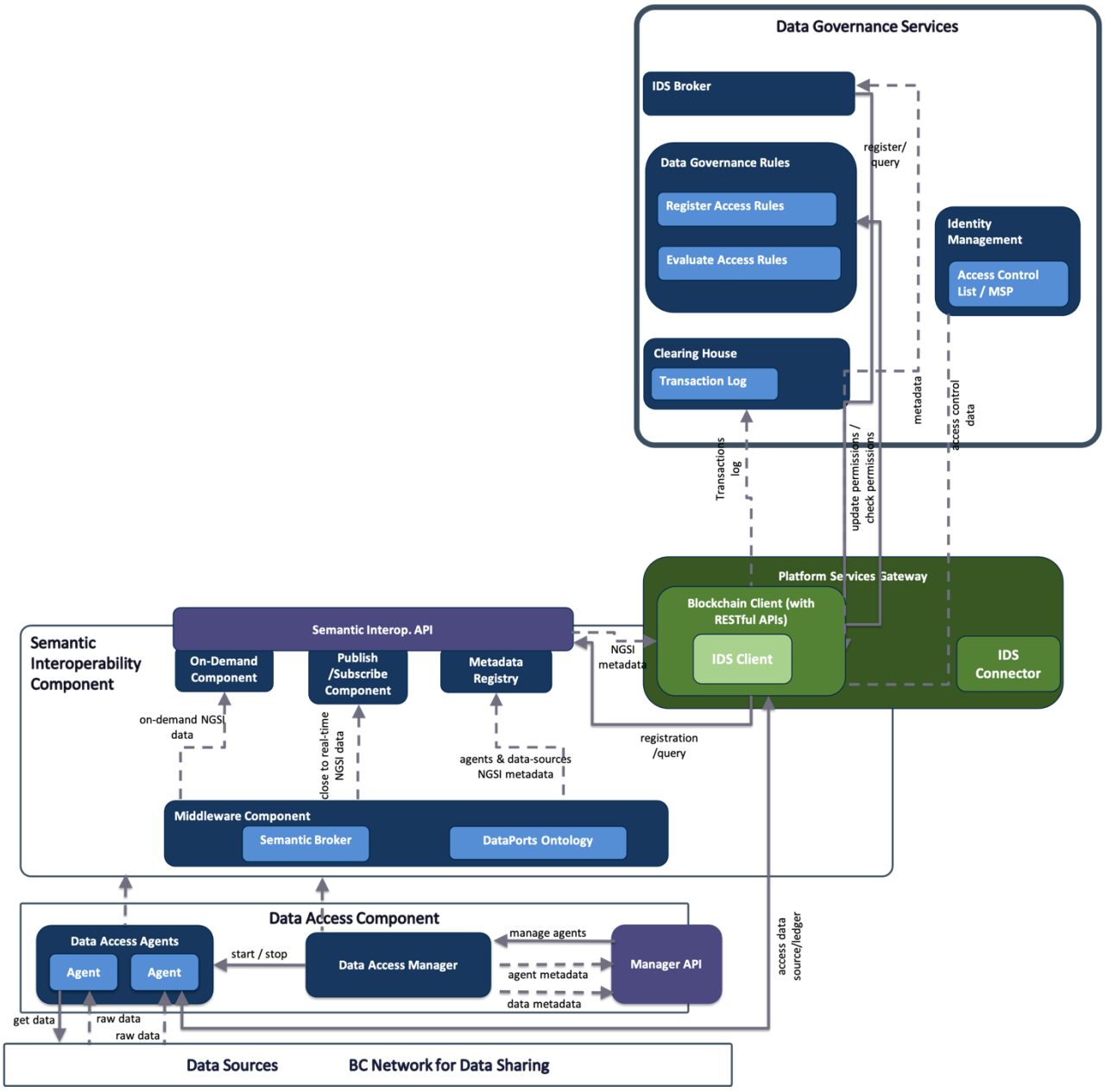

Figure 22 - Interactions between the Data Governance related components

6.4.6 Subcomponents / Tools

The Hyperledger Fabric network is made from several subcomponents and tools that helps it communicate with other components in the platform and generate transactions that update the ledger.

These subcomponents belong to the Hyperledger Fabric architecture itself and can be deployed in a modular way in order to ensure high availability, decentralization, and scalability.

- Storage state of transaction: CouchDB
- Access to the network through Fabric API/SDK
- Smart contracts to allow/deny access to data and store transactions: Golang Chaincode
- Transaction processing and validation: Orderer and Endorsing Peers

The different necessary components of the Hyperledger Fabric network are described below:

- Peer. They are nodes contributed by the entities - organizations - participating in the Blockchain (registry). These nodes store both the Blockchain and a key-value database of the current state of the chain. Hyperledger differentiates between several types of peers: committing peer, endorsing peer, leader peer and anchor peer. Every peer is a committing peer: storing the Blockchain and maintaining the World State (see State definition below). Any peer that has a chaincode installed is an endorsing peer. Then each organization can have one or more leader peers. These are responsible for propagating the blocks received from an Orderer (Ordering Service node) to all committing peers in the organization. An anchor peer is a peer used to communicate with peers from other organizations on the Channel, so that the Channel's membership is properly managed.
- Channel. It is a private sub-network between two or more peers, so that only the authorized participants can participate in it. Each channel maintains its own Blockchain, along with its Smart Contracts, and only channel participants can view that Blockchain and the transactions sent to it. In addition, the consensus on the next block, and the transactions included in it, is done by channel. Therefore, it can be understood that each Channel is equivalent to a different Blockchain. It is a mechanism that allows for a higher level of isolation between participants if deemed necessary. It should be noted that the same peer can participate in more than one Channel at the same time, keeping both isolated.
- Chaincode. It is the name that Smart Contracts receive in Hyperledger. These can be implemented in common technologies like Java, Node.js or Go. And they can interact with the state of the blockchain through a well-defined REST API. In Hyperledger there are two types of chaincode: user - programmed by the blockchain operators - and system - which provides the logic needed by Hyperledger to execute part of the transaction flow.
- Ordering Service. It is the service in charge of building the blocks for each Channel, executing for this the consensus phase of the transaction flow. This service is modular and has different consensus protocols. Said service may be made up of a set of nodes. In fact, to be fault tolerant it is interesting that it is not restricted to a single node.
- State. The state in each Channel is made up of its Blockchain (registry) and the World State, key-value database that maintains the current state of the registry. Thus, each of the peers that belong to a Channel maintains a local copy of both. In addition, the implementation of Hyperledger Fabric (from version 1.0) allows to use two different databases to store the World State, GoLevelDB or CouchDB. The first is a memory-embedded database, while CouchDB uses a client-server model through a REST API over HTTPS.
- Private Data Collection. Collection of private data that can only be stored by the peers of the subset of organizations of a Channel authorized to it. It is a mechanism that allows information to be exchanged between an organization subset of a Channel without the need to create another one. The fact that only the peers of the organizations authorized to do so can store the information makes the level of replication of the information stored in them less. The database in which it is stored is usually called SideDB.
- Client. It refers to a client application, which is authorized to participate in the Blockchain (Channel) and proposes transactions. This implies sending the transaction to one or more endorsing peers based on the endorsement policy established in said Channel. Once it has received enough responses with the appropriate approval, it sends the transaction along with the endorsements obtained to the Ordering Service, which includes it in a block and propagates it for subsequent validation and confirmation in the registry.

The communication with the chaincodes is achieved through the Hyperledger Fabric Client (HFC). The HFC client manages the query/invoke requests to the blockchain network, where the data governance chaincodes will be deployed.

These smart contracts (chaincodes) satisfy the need for application functionality relating to data governance through smart contracts:

- The chaincode which acts as a broker, and which guarantees secure access to immutable data evaluates access based on the granted/revoked access rights. The identity management is Blockchain based. Finally, using the Blockchain offers us an implementation of the clearing house component through the transaction log, using the transaction log as a clearing house component.
- Broker chaincode allows to register all the metadata pertaining to the dataset a data provider wishes to share, and which is stored at his premises, and provides capabilities of querying this metadata based on various search parameters.
- Governance rules chaincodes allow to register the access rules, to define who can access the datasets published in the broker and how, and to evaluate these access rules once a particular data consumer wishes to access the dataset. Upon evaluation of the access rules for a particular dataset and data consumer, the access is either granted or rejected. These chaincodes also allow to manage the lifecycle of the access permissions, allowing to query or revoke the granted access. Additionally, the chaincodes provide user and organization management, allowing to register and query users and organizations (the access rules are defined for dataset for particular users/organizations).
- As mentioned above, the clearing house, whose function is to allow provenance and audit on all dataset access requests/granted permissions is implemented using inherent built in Blockchain functionality. Blockchain retains history of all transactions executed on the ledger using chain of blocks, to which each new transaction is appended. This history is tamper-proof, immutable, and replicated to all the peers in the network, and therefore can serve as a single verifiable source of truth for audit. For each artifact on the ledger (in our case datasets and granted permissions) Fabric provides us with APIs to check when and by whom the artifact was accessed and modified. This serves as a perfect implementation for the functionality required by the clearing house component.
- Identity management is inherent in any permissioned Blockchain implementation. Each Blockchain organization defines a Certificate Authority (CA) server, which is responsible for issuing crypto material (certificates and public and private keys) for all entities in the network. These include the Blockchain nodes (peers and orderers), and the clients of the network (whether these are applications connecting to Blockchain with the help of Fabric SDK, or end-users, authenticating, connecting and executing transactions with the help of these applications). Each transaction on the chain, as well as all additional messages (such as, peers transaction evaluation responses and orderer messages to peers) are signed with the issuing entity crypto-material and validated (by using the public-private key) before being committed to the ledger. This mechanism is used to ensure secure and verifiable trail of updates of the ledger. Additionally, as part of the governance rules identity management, a user registration chaincode is introduced to allow storing organizational and organization user information on the chain, to allow for specification of governance rules and granting of permissions on user/organizational basis for all the shared datasets.

These components are deployed in a Linux environment in the form of Docker containers, which allows for better management of subcomponents and greater availability of the network itself.

The middleware will take care of abstracting the complexities of the client SDK by offering a friendlier REST API. A framework that facilitates the development of asynchronous applications will be chosen:

- REST API development is done through Spring Boot to facilitate development.
- Interaction with the Blockchain will be done through the java SDK provided in Fabric. The possibility of using Spring Reactor to facilitate asynchronous communication is also contemplated if the SDK does not offer the necessary support.

The middleware will also oversee creating “user sessions” and mapping them to the corresponding private keys that sign the transactions.

When a Fabric transaction proposal is created by a DataPorts user, the Middleware receives it and sends it

to the Endorsing Peers in the relevant Fabric Channel (the ledger that means to be updated by the transaction) with the help of the Fabric SDK deployed in its own Docker container. Endorsing peers then check the transaction proposal against the logic contained in the Chaincode installed in the Channel that receives the transaction. If successful, Peers then send the signed transaction proposal to the Ordering Service, which will once again check all digital signatures and update the current status of the ledger in the CouchDB database. After the write operation has taken place, the network broadcasts an event that allows the rest of components to know that the transaction has been performed successfully.

6.5 PROCESS-BASED ANALYTICS COMPONENT

The Process-based Analytics component aims at optimizing business process using machine learning techniques. A business process in the context of DataPorts can be the flow of vessels within the port's service area or transport (containers or goods) operation process. By proactively predicting the future states of the ongoing process, the component provides forward-looking perspectives for the users to make decisions. The Process-based Analytics component includes the following three main components and associated functionalities. By exploiting advanced data analytics techniques and machine learning, these components offer decision support for terminal and process operators, thereby facilitating proactive management of port processes:

- Ensemble Predictive Process Monitoring: This component uses ensembles of deep learning models (recurrent neural networks) to provide accurate predictions for each point during process execution, i.e., in a streaming fashion.
- Prescriptive Process Monitoring: Building on process predictions, Online Reinforcement Learning allows automating the process on whether and when to adapt a running process. We apply state of the art Reinforcement Learning algorithms to the problem of identifying the signs of possible failure early and accurately.
- Explainable Predictive Process Monitoring: This component aims at providing interpretations on why a certain prediction is made by a black-box predictive model, by the deep learning models used in the first component above. To generate highly accurate predictions and at the same time facilitate interpretability for predictive process monitoring tasks, we leverage the concept of model induction from interpretable machine learning (ML) research.

6.5.1 Position in the Architecture

Process-based Analytics component is part of the Advanced Big Data Analytics component of the DataPorts platform. It analyses business processes by using both historic and real-time data available inside the DataPorts platform to provide its predictive results to cognitive applications, which can inform the end-users about the predictions.

6.5.2 Contribution to the Platform and the Platform's Objectives

The Process-based Analytics component offers services to end-users of the DataPorts platform and data-scientists. Using data on the ports' ongoing and recurring processes, this service assists decision-making activities during the ports' operations. In a transport operation business process scenario, it can entail an accurate prediction for the process's delivery time, notifying process managers about potential delay and suggesting adaptations in order to avoid latencies.

The Process based Analytics component is dependent on two main Components: The Data Abstraction Virtualization Component and Semantic Interoperability Components. As shown in Figure 8, the Process-based Analytics component queries the Data Abstraction Virtualization Components about historic process monitoring data, i.e., event logs in order to create the Event log repository. Furthermore, it receives close to real-time NGSI data from the Semantic Interoperability Component.

The data received from these two sources compose the datasets, which are used to train the Ensemble predictive process Monitoring sub-component.

The output of this component is to offer advanced data analytics and ML services that optimize the ports' processes and facilitate proactive adaptions performed by process managers.

6.5.3 Addressed Requirements

Table 16 shows the requirements addressed by the Process-based Analytics Component.

ID	Description	Category
R3.27	As an end-user I want the platform to provide cognitive services in the Port domain for supporting my decision-making process so that I could improve my KPIs	Functionality
R3.28	As an end-user, I want software components based on State-of-the-Art Machine Learning (ML) algorithms, specifically customized to the ports domain, so that I could easily and automatically create models	Functionality
R3.29	As a data scientist I want software components based on state-of-the-art machine learning (ML) algorithms, specifically customized to the ports domain, so that I could easily create accurate models	Functionality

Table 16 - Addressed Requirements by Process-Based Analytics Component

6.5.4 Description of the Component

The Process-based Analytics component consists of three main components and functionalities, realizing the functionalities shown in Table 17.

ID	Description	Captured by requirement
F-3.11	The DataPorts platform provides the data owners data driven analytic services	R2.16, R3.28, R3.29
F-3.12	The DataPorts platform provides the consumers and end users new AI and cognitive applications	R2.17 R3.27
F-3.13	The DataPorts platform provides tools to help decision processes	R3.27. R3.29
F-3.4	The DataPorts platform establishes machine learning models	R3.32

Table 17 - Functionalities implemented by the Process-Based Analytics component

These three main components are detailed in Figure 23, where it is illustrated how they play together as part of the Process-based Analytics component.

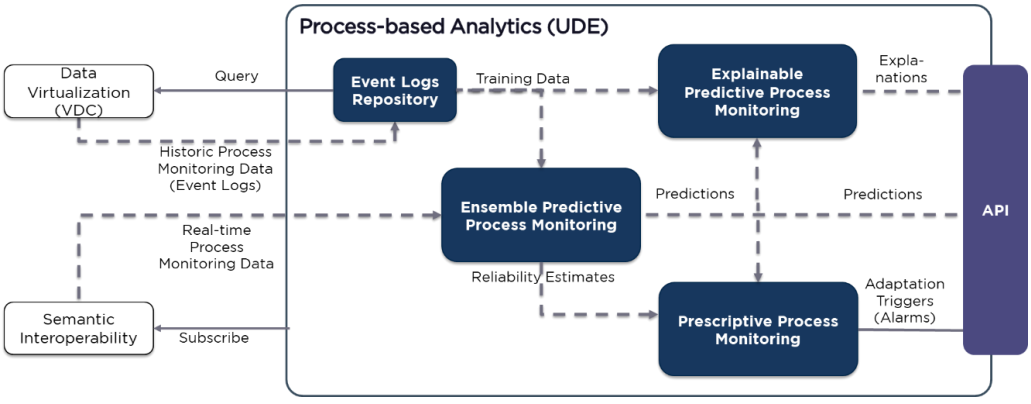

Figure 23 - Overview of Process-based Analytics Component

6.5.4.1 Subcomponent: “Ensemble Predictive Process Monitoring”

Failing business processes, like delayed delivery of cargo containers, can be costly to the entity conducting the business. To prevent business processes from failing and the costs associated with said failure, predictive business process monitoring predicts how an ongoing case will unfold. To this end, predictive business process monitoring uses the sequence of events produced by the execution of a business case to make predictions about the future state of the case. If the predicted future state of the case indicates a problem, the ongoing case may be proactively adapted, e.g., by re-scheduling process activities or by changing the assignment of resources.

There are two quality-criteria for these predictions. First, predictions should be accurate. When adaptation decisions are based on inaccurate predictions, this may imply unnecessary adaptations (e.g., if a delay is falsely predicted) or missed adaptations (e.g., if an actual delay is not predicted). Second, predictions should be produced early during process execution. Earlier predictions leave more time for adaptations, which typically have non-negligible latencies.

As a solution to address this trade-off, this component in addition to a prediction also computes a reliability estimate. Reliability estimates help operators distinguish between more and less reliable predictions on a case-by-case basis. Together with the earliness indicators, reliability estimates can help operators decide whether to trust an individual prediction enough to adapt the running process.

Thereby the main output of the component is the prediction about the future state of a monitored business process at each step of the process, as well as a reliability estimate of said prediction. The main input of the component is monitoring data about the monitored business process, which allows the component to make its predictions. This data about business processes must split into discrete time-steps, as outlined above, and should be of a high quality, that is conforming to the requirements R-3.42 and R-3.43.

Thereby, this component establishes machine learning models (F3.4) provides data owners data driven analytic services (F-3.12), consumers and end users (process operators in this case) new AI and cognitive applications(F-3.13).

6.5.4.2 Subcomponent: “Prescriptive Process Monitoring”

This component automates the decision on whether to adapt or not. To this end, the component learns when a prediction should be acted upon. To do so, a reinforcement learning agent analyzes the data available for the ongoing business process whenever a new event is added to the process log. It then decides whether an adaptation is necessary at the current point time or not.

At the end of a business process, the agent receives different rewards, depending on the adaptation decision it has made for this business process. Generally, these rewards reinforces both accurate and early adaptation decisions by the reinforcement learning agent, who permanently learns how to increase the rewards it receives. By in such a way providing useful adaptation decisions to the data owners, consumers and process operators as end-users, the component addresses the functionalities F-3.12 and F-3.14 of the DataPorts platform, as they are described in Table 17.

The main output of the component is the action the reinforcement learning agent decides upon, which indicates the need, or lack thereof, for an adaptation at the current time in the observed business process. The main input of the component is the prediction and its reliability, provided by the “Predictive Process Monitoring via Deep Learning Ensembles” component.

6.5.4.3 Subcomponent: “Explainable Predictive Process Monitoring”

Using black-box models without being able to interpret their decisions has potential risks. Such risks could hinder the acceptance and trust of users in adopting the predictive assistant system. Therefore, in addition to predictions, we provide the users with interpretations in this component.

By providing interpretable models for the predictions as additional input, this component facilitates F-3.12, F-3.14, and F-3.4.

6.5.5 Interactions of the Component

Component requires historical data to train the predictive models and bootstrap the reinforcement learning algorithm. A probable source for this data is the "Data Abstraction & Virtualization" component. The "Data Abstraction & Virtualization" component will provide as a subcomponent the "Virtualized Data Repository software package", which will store historic data for the explicit purpose of providing it to the data analytics components like ours. Since the core of the "Virtualized Data Repository software package" subcomponent will be a MongoDB database, our component will directly query and receive the necessary data from this database.

In addition to the historical data, component requires access to real-time data on ongoing business processes. A probable source for this data is the "Semantic Interoperability" component. The "Semantic Interoperability" component will publish streams of data through a REST interface that allows component to subscribe an API-endpoint of our own to theses streams. The concrete messages that are exchanged between the components will be in the JSON-LD format. The specific data linked in these messages will then be queried by the component.

The output of Process-based Analytics component i.e., the predictions, explanations, and adaptation triggers will be communicated to novel AI/cognitive apps through our component's API. This interaction between our component and the end user, will be implemented using a REST Service, which will contain methods for retrieving and handling the results of our component.

6.5.6 Subcomponents / Tools

The "Ensemble Predictive Process Monitoring" component takes training data from the event log repository. During run time, the real-time process monitoring data provided by Semantic interoperability would be used as input and generates a prediction about the process' failure, together with a reliability estimate of the prediction. As shown in Figure 24, this is achieved by combining the individual prediction of each ML-Model.

The ensemble prediction for a check point is computed as a majority vote, while the corresponding reliability estimate is computed as the fraction of individual model that predicted the majority class. The prediction together with reliability estimates help operators decide whether to adopt changes on a case-by-case basis.

As a technological basis, the "Ensemble Predictive Process Monitoring" component uses the framework Tensorflow to implement its machine-learning functionalities. Therefore, the component is written entirely in the Python programming language.

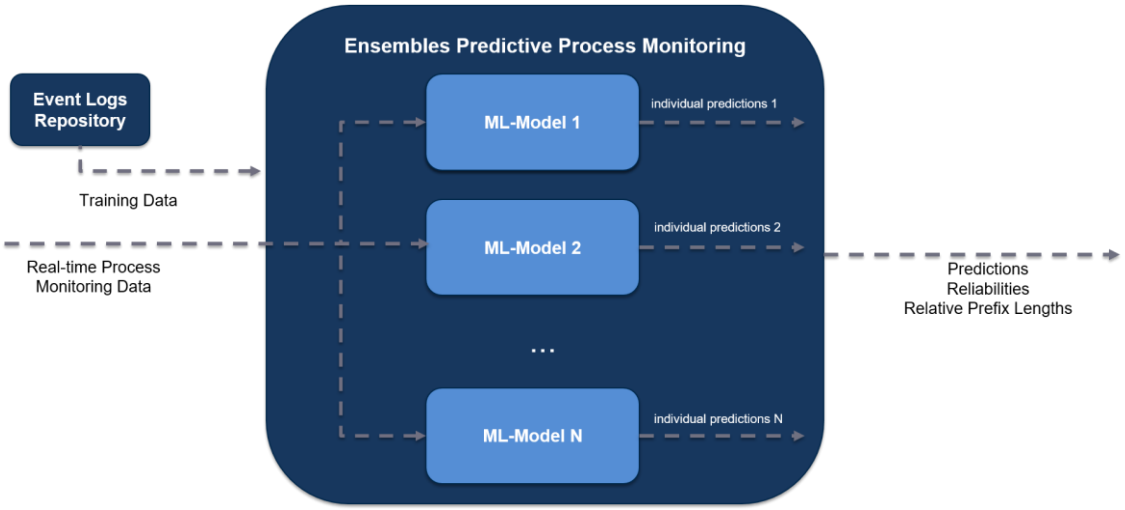

Figure 24 - Ensembles Predictive Process Monitoring

The “Prescriptive Process Monitoring” component’s main goal is to prescribe process adaptations to process operators, informing their adaptation decisions. Because the reliability estimates generated by the “Ensemble Predictive Process Monitoring” cannot be interpreted as the stochastic probability of the predictions being accurate, and because they can vary in variance and accuracy in of themselves, it is difficult for a human to interpret them correctly.

Mainly, the “Prescriptive Process Monitoring” component learns to interpret the reliability estimates in context, on a process-by-process basis, in order to propose adaptations as adaptation triggers in a beneficial manner. To do so, the component receives data on the monitored process, as well as the predictions and reliability estimates generated by the “Ensemble Predictive Process Monitoring” component. It outputs adaptation triggers, which prescribe to the process operator, when to adapt a given process case.

For the “Prescriptive Process Monitoring” component to be able to continuously learn from ongoing processes, even if they are altered based on the component’s outputs, it uses a reinforcement-learning technique. As such, the reinforcement-learning agents that the component trains can adapt, at least to certain extents, to changing conditions under which the processes are executed. For each different process that the component is used with, another reinforcement learning agent is trained. From the process monitoring data that the component receives as input is, to align with the conceptual reinforcement-learning framework, rewards are generated and used to train the reinforcement-learning agents.

Like the “Ensemble Predictive Process Monitoring” component, the “Prescriptive Process Monitoring” component is also implemented using the Tensorflow framework and written in python programming language.

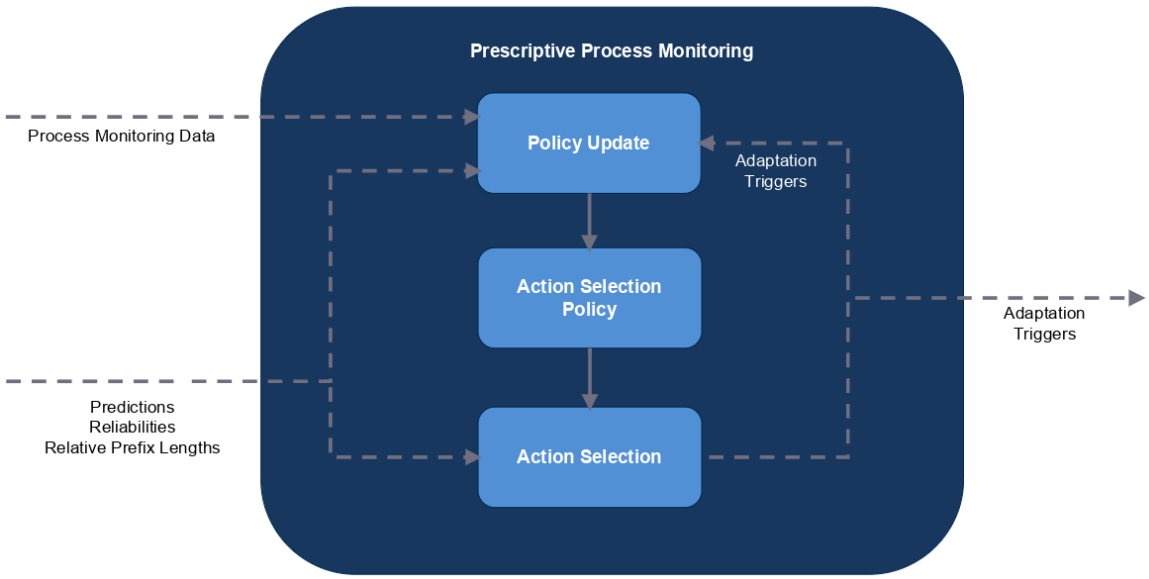

Figure 25 - Prescriptive Process Monitoring

In the field of predictive process monitoring, non-interpretable models perform consistently better than their interpretable counterparts. However, non-interpretable models provide only predictions to the users. Using black-box models without being able to interpret their decisions has potential risks. To generate highly accurate predictions and at the same time facilitate interpretability, we leverage the concept of model induction from explainable artificial intelligence (XAI) research.

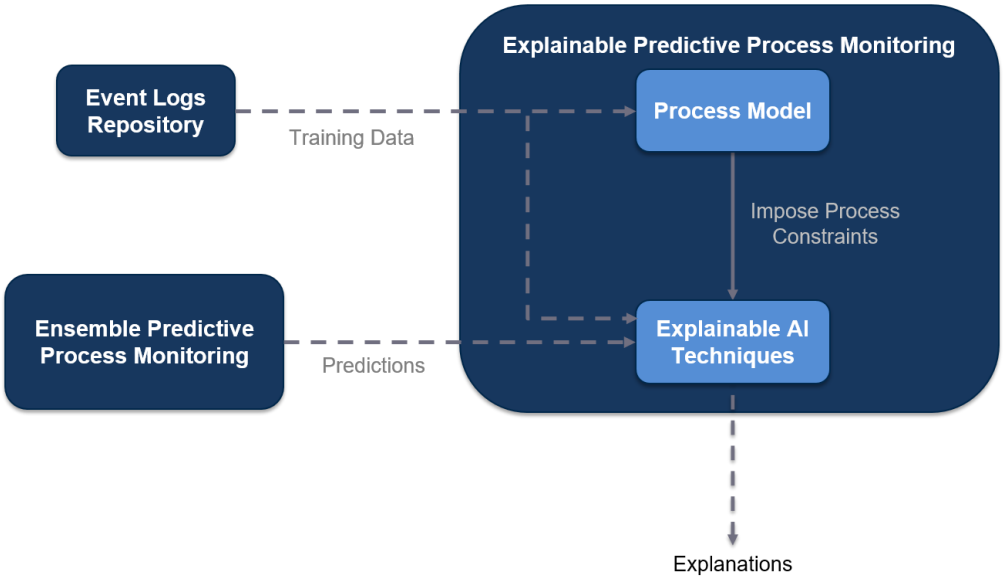

Figure 26 - Explainable Process Monitoring

Figure 26 shows the sub-components and tools that are used to realize the Explainable Predictive Process Monitoring component. A process model is mined by the process discovery algorithm from the process execution data. We use the knowledge of the process model to impose domain-specific constraints on explainable AI techniques. This ensures that the generated explanations follow the domain knowledge of BPM. For example, for a prediction point we would like to provide interpretations, we exploit the process model to subtract trace prefixes (S) that have similar patterns. This is achieved by identifying the prefixes that reach the same marking in the process model as the instance of interest. Then, the subset S is used to query the non-interpretable model (Deep Learning Ensembles in this case) to get the corresponding predictions (P) from the black box. In the last step, the subset of the prefixes together with the corresponding predictions (P) from the non-interpretable model are used to induce interpretable models for generating explanations to the users.

6.6 AUTOMATIC MODELS TRAINING ENGINE

This component will use the data available in the DataPorts federated platform to train a set of specific ML models to make predictions over ports-related data. Hence, according to the initial requirements and datasets provided by the pilots, we foresee that the most suitable ML models to train are time series forecasters. Following a low-code approach, in which no previous expertise in ML is required, the end-user will be able to select a wide range of supervised time series forecasting services, along with the desired service configuration parameters, such as the target variable to be forecasted (for instance, the ETA of a container), the desired features from the available training datasets and the time horizon ahead to make the forecasts. Accordingly, the training engine will automatically generate a processing workflow depending on the computing infrastructure available, thus performing the training of all the time series ML models disposed. Finally, the best trained model amongst them, which offers the best quality metric to make new predictions, will be selected to be deployed as a service, thus being potentially used to build a cognitive service.

6.6.1 Position in the Architecture

This component is a result of T3.4 Data analytic sand AI services for cognitive applications and an architectural component of the Advance Big Data Analytics block from the platform point of view. According to the IDS Reference model, we consider this component as a Data App as it uses the available data for processing purposes, i.e., building a ML model. This component will be wrapped inside an IDS connector to ensure that

the shared data is only used for analytics aims. The main data producers of this component are the Semantic Interoperability component (T3.2) and the Data Abstraction and Virtualization component (T3.3). This component also interacts with the IDS Broker to retrieve the available datasets at the DataPorts platform and get a formal description (metadata) regarding the data structure. The core functionality of the cognitive services developed in the context of the pilots are the deployed models by this component. The interaction with such models is expected to be through a REST API.

6.6.2 Contribution to the Platform and the Platform's Objectives

This component will contribute to the platform by offering ready-to-use cognitive services to the end-user of DataPorts as explained below in the interactions of the component (Section 6.6.5) and in concordance with the related uses cases of Valencia Port: Tracking of Transport Operations and Port Authority Data Sharing and Analytics Services. The trained cognitive service will provide its outcomes through a REST API.

The cognitive services focus on specific ports oriented KPIs, directly related with the available datasets and identified by the Port of Valencia such as the amount of received goods that will occur in a certain time horizon, the number of containers delivered to a specific Spanish region, the occupancy of a particular port or terminal in a future time slot, etc.

To train a cognitive service, the end-user will interact with the component throughout a web dashboard specifically developed for that matter, to configure and personalize the forecasting pipeline that will process the desired available data and select the best predicting model for the chosen cognitive service.

The configuration of the pipeline will allow the user to select a set of different personalization parameters for each service, for instance: port to predict the occupancy, import or export of a certain good… Those parameters will be available through the interaction with the IDS broker which will provide the metadata of the available datasets previously registered into the DataPorts data space.

To receive real-time data, when required in any of the use cases, the component will interact with the semantic interoperability component which deploys a Context Broker as a subcomponent. For that, the component subscribes to the broker to obtain continuous data previously transformed and aggregated. As a result, the component will outcome predictions over time updated with the new incoming data.

To retrieve batch data for the cognitive services that demand it, the component will interact with the data virtualization component which provides historical data consistent with the quality criteria specified in the DataPorts platform in parquet files system. Hence, the component will ingest such data to be temporally stored in a distributed filesystem for parallel processing purposes in different computing nodes.

The component might be used independently to provide trained cognitive services based on the historical data from the port-oriented and publicly available datasets, identified in the use cases and downloaded from AEAT and/or Valencia PCS. Thus, by accessing to a local repository containing the data and following the same procedure explained above, a potential the end-user could train a specific and personalized cognitive service interacting with the web-based user interface of the component. As a result, the predictions offered by the service will be provided through a REST API accessible from any point if Internet connection is available.

6.6.3 Addressed Requirements

Table 18 shows the requirements addressed by the Automatic Models Training Engine:

ID	Description	Category
R2.16	DataPorts will follow the Big Data Analytics as a Service (BDAaaS) paradigm, providing a level of abstraction to application developers about the implementation and set-up details of the data platform, thus simplifying deployment	Portability

ID	Description	Category
R3.26	As an end-user I want the platform to provide cognitive services in the Port domain for supporting my decision-making process so that I could improve my KPIs	Functionality
R3.28	As a data scientist I want software components based on state-of-the-art machine learning (ML) algorithms, specifically customized to the ports domain, so that I could easily create accurate models	Functionality
R3.29	As a data scientist I want a distributed AI platform so that huge data-volumes could be processed	Scalability
R3.30	As a data scientist I want to use continuous data streams, so that the platform provides predictions in near real-time.	Functionality, Performance Efficiency
R3.31	As a data scientist, I want the platform to deal with the original data sources heterogeneity, real-time (streaming) or persistent data, relational or non-relational databases (NO-SQL), so that I can use the data without taking into account the underlying storage system	Interoperability
R3.48	IDEs and APIs for new applications development	Performance Efficiency

Table 18 - Addressed requirements by Automatic Models Training Engine

6.6.4 Description of the Component

The Automatic Models Training Engine component consists of four main functionalities:

ID	Description	Captured by requirement
F-3.4	The DataPorts platform establishes machine learning models	R3.28 R3.29
F-3.11	The DataPorts platform provides the data owners data driven analytic services	R2.16, R3.28, R3.29
F-3.12	The DataPorts platform provides the consumers and end users new AI and cognitive applications	R2.17 R3.27
F-3-14	The DataPorts platform provides smart API for cognitive services	R3.18

Table 19 - Functionalities implemented by Automatic Models Training Engine

The functionalities define a Big Data Analytics as a Service approach from a generic point of view. This approach defines a pipeline process to create a ML model as results. How this process is realized in the context of the project is detailed next:

- Training configuration: As a first step, the training process must be configured by the end-user to establish some overall parameters. For instance, the forecasting window of the target variable to predict (the next hour, the next week, etc.) the data variables to use for training the model, the expected accuracy threshold to discard a model, or the computing infrastructure to use (the number of processing nodes to train models). This configuration will be provided using a web-based user interface. This step fulfills the functionality F-3.15.
- Historical data import and distribution: To improve the performance of the training process to create the analytical models, we import the required historical data, perform some transformations and store it in a distributed repository. Thanks to this approach, several computing nodes could share the resulting dataset for training purposes, reducing the training time required.

- Data preparation (Optional): Analytical models require the input data to be compliant with some requirements: every data row must be synchronized with the same time period (every minute, hour, etc.), quantitative values must be normalized, and categorical values must be codified by using data analytics techniques such as one-hot encoding, and so on. Generally, data coming from original sources does not consider such constraints; therefore, a common solution is the implementation of an optional data preparation phase, so a data engineer will be able to implement a custom data preparation process by using their own preferred script.
- ML Techniques evaluation: By default, the engine includes a set of ML techniques to train models and evaluate their accuracy. These techniques will be selected from initial tests performed by using the datasets and business goals provided by the use cases and, consequently, to support functionality F3.5. In this evaluation phase, our goal is to detect the technique that best performs with port's domain data. Some examples of such techniques are Autoregressive integrated moving average (ARIMA), Recurrent neural networks (RNN) such as Long-short term memory (LSTM) networks, Gradient boosting trees (GBM) and Support vector regression (SVR) within others.
- Hyperparameters optimization: As our training engine does not require manual intervention, the most suitable training parameters for each ML technique must be automatically generated. To reduce the number of required models to train in order to find those parameters, this component uses hyperparameters optimization libraries to avoid the exploration of every parameter combination. This optimization simplifies the development in line with F3.22
- Distributed training execution: As a result of using several ML techniques and different parameter combinations, the model training process should be distributed into a set of computing nodes, so that several models could be simultaneously trained thus meeting the requirement R3.29. Therefore, this component dispatches and monitors the ongoing training processes. Since the training is distributed, the dataset must also be available (in memory or in disk) on the computing nodes.
- Trained models logging: Our component will log all the trained models for comparison purposes: the technique that performs best, the training time, the evaluation metrics for each model and so on. This log will be presented visually to the user according to the functionality F3.15.
- Automatic model deployment: The best model from the training process, according to an evaluation metric such as RMSE or MAPE, will be automatically deployed in order to be used by the cognitive apps. This model deployment will provide a standard REST API to send incoming data and to generate novel predictions. This last step is related with functionalities F3.22 and F3.25.

6.6.5 Interactions of the Component

The expected external interactions with the Automatic Models Training engine are shown in Figure 27. Next, these external interactions are detailed:

- End-user: the end-user interacts with our training engine by means of the user interface specifically provided. This UI is a web dashboard for configuring the training process according to the end-user requirements.
- IDS broker: Our training engine requires the data available in the ecosystem as main input. The IDS Broker provides the metadata of the available datasets previously registered into the DataPorts data space. On demand, the training engine requests such metadata for enabling the selection of specific data variables.
- Novel AI/cognitive apps: these apps require the resulting ML models generated by the training engine. A REST Service implements such interaction providing methods for sending data to trained models and retrieving predictions or anomalies.
- Semantic Interoperability (broker): The semantic interoperability component deploys as subcomponent a Context Broker. Specifically, data access component publishes to this broker the received data after is transformed following the DataPorts ontology. When the ML Framework of the training engine requires real-time data, for instance to continuously provide predictions, a subscription to the Context Broker implements the required behavior.

- Data virtualization: Data virtualization component provides historical data consistent with the quality criteria specified in the DataPorts platform. The training engine requires such historical data to be temporally stored in a distributed filesystem to enable the simultaneous access from several computing nodes. The main interaction is then to import such data as parquet files.

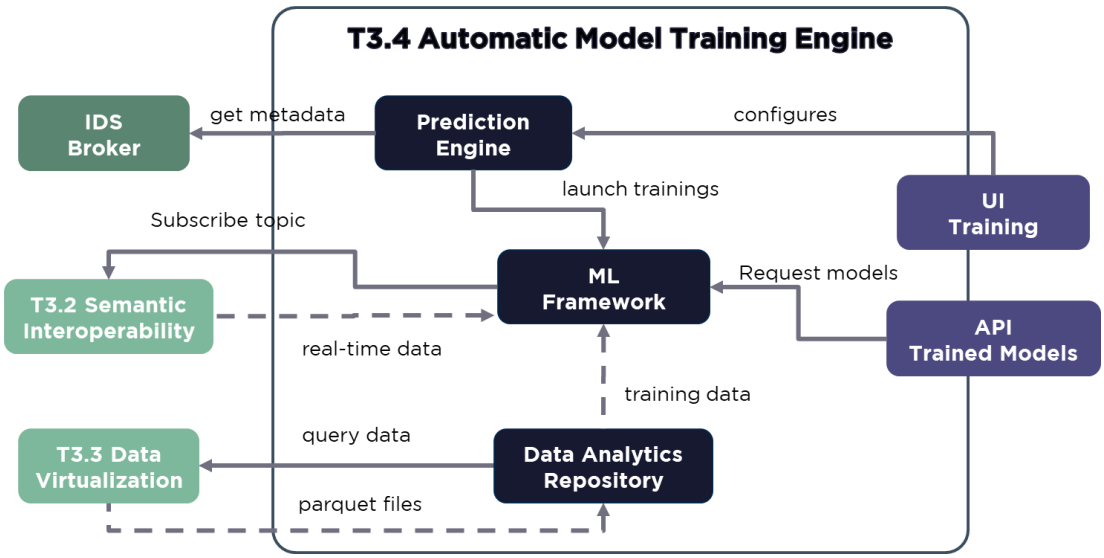

Figure 27 - Automatic Model Training Engine

6.6.6 Subcomponents / Tools

To accomplish the functionalities, the following subcomponents are used:

- Prediction Engine: This subcomponent starts and monitors the training process configured by the user. Its main responsibility is to request the data schema and access information from the IDS broker, and then dispatch a training request to the ML Frameworks. Resulting models will be stored for future access.
- UI Training Interface: this interface is a SPA (Single Page Application) using web technologies, which implements a dashboard. For this task, we will use a JavaScript framework such as Vue.js or Angular.
- ML framework: the ML Techniques expected to be used in the pilots are already implement in wide popular ML frameworks. For classical approaches, we use Scikit-learn, whereas for defining neural networks, we will use Pytorch.
- Data Analytics Repository: The selected datasets for training are stored using the Parquet format, a columnar storage format specifically defined for the Hadoop ecosystem. Data will be distributed using a specific filesystem framework such as HDFS or CEPH. On top of such files, this component provides a query execution engine, Hive or DeltaLake, with additional SQL-capabilities. We will provide a Jupyter web environment in order to implement analytical queries and transformations to adapt the data to the training process. To implement some of these transformations this component includes popular Python libraries and frameworks.
- API for trained models: An interface for retrieving the trained models and additional information about then such as the ML techniques applied and the accuracy indicators.

7 CONCLUSIONS

This document specifies the design and the architecture of the DataPorts platform. Various components of the data platform are introduced, and their interactions with other components are described. This document and the architecture of the platform are the results of a two-year process where several versions of the document and the architecture were iteratively created. In each iteration several meetings and discussion rounds were organized in order to gather feedback and contribution of all the project partners in order to improve the design of the platform and the resulting document.

Furthermore, this document lists the finalised functionalities of the platform and describes how the functionalities cover the requirements identified in deliverable 2.1. In the context of each component, the corresponding functionalities and requirements are specified.

In Section 5.3, we introduce how security is addressed in the DataPorts platform. A list of security mechanism is provided, and it is described how the security related functionalities are covered by various functionalities of the data platform.

This document serves as a reference to finalise the design and the implementation of the architectural components. Furthermore, the list of functionalities of the platform can be referenced in various documents of the DataPorts project.

In addition, to evaluate the architecture and the technical decisions, an internal (in the context of DataPorts consortium) working group will be created to perform these tasks. This working group is a combination of several software system architects and technical experts on Big Data technologies. In the context of WP3, each architectural component will be evaluated concerning its design and the technical decisions, and the results of this evaluation will be documented in the relevant deliverables of WP3.

Furthermore, in the context of WP6 and WP7 different surveys and questionaries are planned and designed in order to gather feedback from potential customers of the DataPorts platform. The results of such questionnaires and surveys will be analysed in order to evaluate the data platform from the end users' perspective.

8 REFERENCES AND ACRONYMS

8.1 REFERENCES

[1] DataPorts' Consortium, “DataPorts' Grant Agreement,” 2019.

[2] DataPorts' Consortium, “Deliverable 2.1 Industrial Data Platforms and seaport community requirements,” 2021.

[3] DataPorts' Consortium, “Deliverable 5.1 Integration, software quality assurance and deployment plan,” 2021.

[4] International Data Spaces Association, “International Data Spaces Association Reference Architecture Model,” 2019.

[5] V. a. C. G. a. K. V. a. D. A. a. D. N. a. L. G. a. F. T. a. V. T. a. M. E. a. K. G. a. K. Moulos, “A Robust Information Life Cycle Management Framework for Securing and Governing Critical Infrastructure Systems,” *Inventions,* 2018.

[6] DataPorts' Consortium, “Deliverable D2.3 Blockchain design specification M09,” 2020.

8.2 ACRONYMS

Acronym List	
AI	Artificial Intelligence
API	Application Programming Interface
ARIMA	Autoregressive integrated moving average
CA	Certificate Authority
DAC	Data Access Components
DAM	Data Access Manager
DAV	Data Abstraction and Virtualization
DB	Database
DoA	Description of Action
EC	European Commission
ETA	Estimated Time of Arrival
ETD	Estimated Time of Departure
EU	European Union
GA	Grant Agreement
GBM	Gradient boosting trees
GDPR	General Data Protection Regulation
HFC	Hyperledger Fabric Client
IDS RAM	International Data Spaces Reference Architecture Model
IDSA	International Data Spaces Association
IIRA	Industrial Internet Reference Architecture
JSON	JavaScript Object Notation
LSTM	Long-short term memory
ML	Machine Learning
MoSCoW	MUST, Should, Could, Won’t
MSP	Membership Service Provider

MVP	Minimum Viable Product
PaFS	Processing and Filtering Software
PC	Project Coordinator
PKI	Public Key Infrastructure
PMS	Port Management System
RNN	Recurrent neural networks
SPA	Single Page Application
SSL	Secure Socket Layer
SVR	Support vector regression
UI	User Interface
VDC	Virtual Data Container
WP	Work Package
XAI	Explainable Artificial Intelligence
XML	eXtensible Markup Language

Table 20 – Acronyms