This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 101017331

D4.5 Implementation of Advanced CPS-IoT RSM Features v1

Deliverable No.	D4.5	Due Date	28/02/2021
Description	D4.5 describes advanced features of the ODIN platform, specifically the Digital Ledger Technologies, for trustworthy resource federation, and the Resource Choreographer, for resource orchestration in use case workflows.		
Type	Report	Dissemination Level	PU
Work Package No.	WP4	Work Package Title	CPS-IoT Resource Management System
Version	1.0	Status	Final

Authors

Name and surname	Partner name	e-mail
Ilias Kalamaras	CERTH	kalamar@iti.gr
Petros Toupas	CERTH	ptoupas@iti.gr
Dimitrios Giakoumis	CERTH	dgiakoum@iti.gr
Konstantinos Votis	CERTH	kvotis@iti.gr
Pablo Lombillo	MYS	plombillo@mysphera.com
Pilar Sala	MYS	psala@mysphera.com

History

Date	Version	Change
14/12/2021	0.1	Initial draft containing table of contents.
04/01/2022	0.2	Update ToC
21/01/2022	0.3	Update ToC
25/01/2022	0.4	Draft content
14/03/2022	0.5	Collection of input from partners
21/03/2022	0.6	Content addition
22/03/2022	0.7	Content integration and finalization
30/03/2022	0.8	Integration of peer-review comments
30/03/2022	0.9	Version ready for quality check
05/05/2022	1.0	Version ready for submission

Key data

Keywords	Digital Ledger Technologies, blockchain, resource federation, resource choreography, resource workflow design
Lead Editor	Ilias Kalamaras (CERTH)
Internal Reviewer(s)	Ernesto Iadanza (UoW) Marcello Chiurazzi and Gastone Ciuti (SSSA) Alejandro Medrano (UPM)

Abstract

Deliverable D4.5 "Implementation of Advanced CPS-IoT RSM Features v1" describes advanced features of the ODIN platform, namely the Digital Ledger Technologies for trustworthy resource federation and the Resource Choreographer for resource orchestration. Together with D4.2 (and its further versions), they complete the descriptions of the core ODIN components regarding resource management and interaction. Digital Ledger Technologies is a key component that enables trustworthy resource federation, so that resources of one ODIN instance (e.g. a hospital) are visible and usable by remote ODIN instances, as if they were local, thus supporting the distributed nature of the ODIN platform. The Resource Choreographer allows the composition of the existing resources into operational pipelines and workflows that implement high-level logic, to support hospital applications and use cases. For both components, this deliverable describes architectural aspects, their envisioned operation and the tools to be used to implement their functionalities.

Statement of originality

Table of contents

List of tables

List of figures

1 Introduction

This deliverable describes advanced features of the ODIN platform, specifically the Digital Ledger Technologies, used for trustworthy resource federation, and the Resource Choreographer, used for resource composition into high-level operational workflows. Together with D4.2 “Implementation of Local CPS-IoT RSM Features v1” (and its subsequent versions D4.3 and D4.4), it completes the description of the core ODIN platform components responsible for resource management and interaction. While D4.2 focused on low-level resource management, such as semantic resource abstraction, resource communication and metric collection, D4.5 focuses on advanced features that enable the expansion of the ODIN platform into larger distributed ecosystems and custom high-level applications and domains.

The deliverable describes two components of the ODIN platform:

- The Digital Ledger Technologies (DLT) component. This component enables resource federation across ODIN instances in a trustworthy manner. Resource federation is a key idea of the ODIN platform, allowing its implementation to be distributed, with resources of one instance (e.g. a hospital, a cloud machine, etc.) being shared by other remote instances. Resources are valuable assets; therefore ODIN instances need to ensure that they are shared in a trustworthy manner. DLT technologies, powered by a blockchain infrastructure, allow all resource sharing requests and permissions to be handled and all transactions between ODIN instances to be securely audited, ensuring immutability and non-repudiation. Section 2 is devoted to the ODIN DLT components.
- The Resource Choreographer component. This component allows the composition of resources into operational workflows that implement high-level logic. Based on the semantic abstractions of each resource (i.e. the resource descriptors, described in D4.2, constructed according to the corresponding entities of the ODIN ontology, described in D3.2), each resource will possess, among others, a number of input and output data channels and types. The Resource Choreographer allows the user to design workflows by connecting the outputs of one resource to the inputs of another. These workflows are meant to represent high-level business logic for implementing hospital operations, such as the ones described in the ODIN use cases, by specifying how resources should communicate with each other. The Resource Choreographer component is described in Section 3.

For each of the above components, the current deliverable presents the relevant information in the following structure:

- Requirements review, i.e. which ODIN requirements are fulfilled by each component;
- Architecture review, i.e. how each component fits in the ODIN platform architecture;
- Applicable technologies for the implementation of each component;
- A comparison of the applicable technologies to select the most appropriate ones;
- A list of features to implement in the first version of each component.

At the current stage of the project (end of first year), the development of both components is in the design phase, with the specification of the requirements, architecture and overall functionality of each component. Implementation will start from the beginning of the second year.

1.1 Deliverable context

Table 1 provides an overview of the context of the current deliverable, in relation to the project objectives and foreseen results.

Table 1: Deliverable context.

PROJECT ITEM	RELATIONSHIP		
Objectives	The deliverable is relevant to ODIN's Objective 1, as it describes components that enable the secure and trustworthy decentralized and federated implementation of the ODIN platform, as well as components that facilitate the usage of the Key Enabling Resources in hospital applications, bridging the gap between KER suppliers and healthcare organizations.		
Exploitable results	• The ODIN Digital Ledger Technologies infrastructure for federated KERs • The Resource Choreographer component for resource composition and orchestration		
Workplan	D4.5 is attributed to the tasks of WP4 "CPS-IoT Resource Management System". Specifically, the tasks involved in the preparation of this deliverable are T4.4 "Digital Ledger Technologies Resource Federation and management framework" and T4.5 "Resource Choreographer module".		
Milestones	D4.5 is a key deliverable of the PREPARATION (MS1), PROCUREMENT PROCEDURE SIMULATION (MS2) and IMPLEMENTATION (MS3) phases of the project.		
Deliverables	D2.3	ODIN Platform catalogue	Regarding the available resources that can be potentially shared and orchestrated
	D3.2 – D3.3	Hospital Knowledge Base and ODIN semantic ontology	Regarding the semantic resource abstractions needed for the Resource Choreographer
	D3.4 – D3.6	Privacy Security and Trust report	Regarding security and trust concepts related to resource federation
	D3.7 – D3.9	Technical Support Plan and Operations	Regarding component documentation
	D3.10 – D3.12	ODIN platform	Regarding the integration of the components in the ODIN platform
	D4.1	CPS-IoT Resource Management System Specification	Regarding the requirements for the DLT and RC components
	D4.2 – D4.4	Implementation of Local CPS-IoT RSM Features	Regarding the resource descriptors and resource communication channels

			needed by the RC and DLT components
	D7.9	Pilot Studies Evaluation Results and sustainability	Regarding component evaluation results of unit/integration testing.
Risks	The following risks are relevant to this deliverable: • Technical problems during component/module development • Risk of time-consuming implementation and integration due to multiple technologies involved Both the DLT and Resource Choreographer components presented in this deliverable deal with communication between resources from a high-level perspective, either for sharing or for workflow composition. For this reason, the risk of delays due to the interconnection of diverse technologies is always present. The semantic resource abstraction and common communication through the resource gateway (see D4.2) offer mechanisms to mitigate this risk. On the other hand, this deliverable offers components that help mitigating the following risks: • Loss of compliance due to a low user acceptance of the system (by providing means that facilitate the creation of custom high-level applications and providing a trustworthy substrate for resource sharing) • Legal restrictions imposed in the execution of the trials (by providing secure and trustworthy means for resource sharing) • Failure to attract proposals for open call (by providing trustworthy means for sharing resources from open calls with ODIN instances, and connecting open call resources with native ODIN resources)		

1.2 Overview

An overview of the ODIN platform architecture, with the two components described in this deliverable, the Digital Ledger Technologies and the Resource Choreography, highlighted, is presented in Figure 1. At the bottom layer lie the Key Enabling Resources, i.e. the robots, IoT devices, AI services, databases, front-end and back-end services, etc., as well as the existing Hospital Information System. These are the elements that empower the design of high-level applications. Resources communicate with each other and with other platform components in two communication layers: either through the Enterprise Service Bus (ESB), after being semantically and syntactically transformed to a common data model and a common bus protocol, or through direct communication with each other, e.g. in intra-robot communication, or for large data transfer.

The DLT and Resource Choreographer components lie above this layer, at the platform service level, together with core components such as the ODIN ontology, resource directory, metric collection and platform management. Both components are connected to the Enterprise Service Bus, for communicating with resources. Communication with the outer world always passes through the ODIN Application Programming Interface (API) gateway, which in turn passes through proper user/service authentication. Details about specific architectural elements of each component are provided below, in sections 2.2 and 3.2.

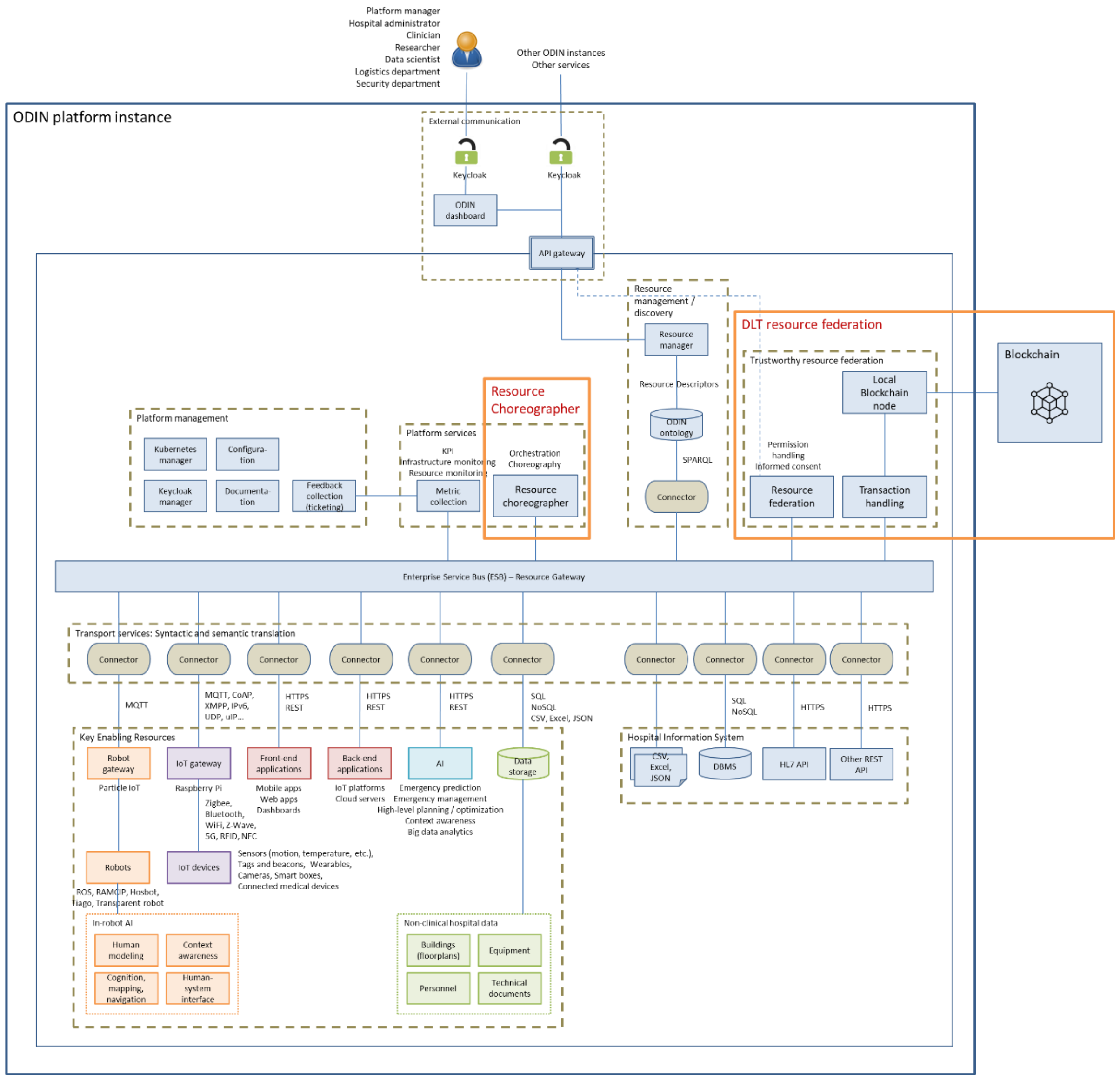

Figure 1: Positioning of the Digital Ledger Technologies and Resource Choreographer components in the ODIN platform architecture.

2 Digital Ledger Technologies

2.1 Requirements Review

The requirements for the Digital Ledger Technologies have been drawn from the ODIN Description of Action, from the requirement collection activities of WP2, from feedback from the project partners and pilot sites, and from the experience of the project partners in DLT technologies in other European projects related to health data sharing. More detailed information about these requirements can be found in D4.1 “CPS-IoT Resource Management System Specification”.

2.1.1 Hospital requirements

ODIN is to operate in an environment with numerous stakeholders where resources will be shared. For that reason, it is important for the ODIN platform to build trust, and a consequent requirement is to monitor log files and informed consents in a safe and encrypted manner. Additionally, all resource events should be auditable, traceable, and immutable for providing accountability and non-repudiation.

Additionally, the ODIN platform should consider how data should be handled. The platform will account for its performance and should opt for storing hash keys of the shared files instead of the data on the blockchain. The performance is fundamental for the ODIN platform along with the continuous and uninterruptable operation of resource sharing, as the auditing mechanism based on the blockchain should seamlessly work for five years.

The adoption of the ODIN platform should account for the scalability of the platform. Storing hashes and structuring a permission network can address scalability issues. Additionally, blockchain should be able to incorporate new members easily and quickly. Hindrances in including members can diminish entities' interest to join the network. Finally, the performance of the system is another point to focus on. The overhead of procedures should be as small as possible so that the platform can provide real-time applications.

2.1.2 Technical requirements

ODIN is to facilitate resource sharing between separate entities in the healthcare sector. As there are numerous entities with resources like data, software, and hardware, ODIN refers to these entities as “ODIN instances”. A functional requirement is to permit the secure direct share of resources in the ODIN platform. Moreover, data are resources with special focus to store in private either locally or on cloud infrastructure.

A vital component for resource sharing will be based on Decentralised Ledger Technologies (DLT) and blockchain. Essentially, the technologies will manage the federated resources after studying the state-of-the-art standards to address this issue. The ODIN platform will aim to develop a consortium-level permissioned blockchain with proper certificates. In this type of network, only authorised nodes will impact the state of the blockchain by adding blocks. Open-source blockchain platforms will be used for enhanced interoperability and auditability.

The blockchain platform will decrease the friction between the instances, as it will store every event taking place between them. Events vary based on the resource. For example, instances may require accessing data or using an AI model from another instance. Despite the specific resource, blockchain will store every event to allow traceability and auditability. Blockchain ledger can allow handling informed consent regarding data.

Smart contracts deployed on the blockchain can allow offering traceability and auditability. In detail, smart contracts are the encoded logic of processes for auditing, resource sharing, and permission handling. The auditing mechanism can make use of the transactions' timestamp to trace events.

Blockchain will bundle the transactions in blocks interconnected to create its chain. The blocks will consist of several transactions committed in a single batch. A REST API will be developed to permit querying blockchain without interfering or joining the network. Finally, a Graphical User Interface (GUI) will be in place to allow users to query the blockchain with ease and obtain the overview and status of the transactions. There is room to include visualisations and KPIs for users to extract knowledge out of the underlying data.

Overall, the requirements for the DLT component can be summarized as such:

- Functional requirements
    - Allow controlled and secure direct share of resources (data, AI models, services, KPIs, IoT, robots, knowledge) between different ODIN instances (e.g. hospitals).
    - Allow each organization (e.g. hospital) to store sharable data and resources in private local or cloud spaces.
    - Use state-of-the-art standards, such as Decentralized Ledger Technologies (DLT) and blockchain to access, manage and federate resources.
    - Develop ODIN's own private/permissioned blockchain network to manage assets, audit transactions and ensure secure data exchange.
    - Use open source platforms for the blockchain, such as Openchain and Hyperledger.
    - Every procedure that transfers health sensitive data and resources from one ODIN instance to another must be logged in the blockchain.
    - Informed consent regarding data/resource transfer across organizations should be stored in the blockchain.
    - Smart contracts should be used for auditing, resource sharing, permission management and informed consent.
    - The blockchain auditing mechanism should include a timestamp in any transaction.
    - The blockchain should allow several transactions to be committed in a single batch.
    - Allow old data/transactions to be purged/obfuscated after a certain amount of time, if required by relevant privacy regulations.
    - Allow queries to the blockchain through a REST API.
    - Provide a graphical user interface to allow users to query the blockchain, obtain an overview of its status and historical transactions, and visualize the blockchain status and KPIs.
- Non-functional requirements
    - Log files and informed consents should be encrypted in the blockchain.
    - All resource transactions across organizations should be auditable, traceable and immutable. Accountability and non-repudiation of the involved parties should be ensured.
    - Hash keys of shared files should be stored to the blockchain.

  - Ensure continuous and uninterruptable operation of resource sharing. The blockchain auditing mechanism should work without any interruption at least for 5 consecutive years.
  - Resource federation should be scalable to large datasets and large ODIN instance federations.
  - Once authorized, resource federation should operate in real-time, resembling local resource usage in performance.

2.2 Architecture Review

The positioning of the Digital Ledger Technologies components within the ODIN platform architecture can be seen in Figure 1, at the right part of the diagram. The DLT consists of the following sub-components:

- The Resource Federation component, which handles permissions for resource access. Each time a remote instance requests access to a local resource, the API gateway passes this request to the Resource Federation component, which checks if permission is granted, possibly through interaction with the hospital administrators.
- The Transaction Handling component, which logs all transactions between remote resources.
- The ODIN Blockchain, which is the place where all permission requests and transaction audits are stored, in an immutable manner. The ODIN Blockchain is primarily hosted in cloud-based blockchain nodes, but ODIN instances (e.g. hospitals) can also participate in the distributed ledger, if they wish to offer computational resources for this purpose.
- The Local Blockchain Node, which is the local node hosted by the ODIN instance, in case the instance participates as a node in the ODIN blockchain.

Blockchain will be a technology for building components to deliver specific functionalities in the ODIN platform. Blockchain technology fits well for the implementation of resource federation due to its suitability with the following requirements:

- Permission handling
- Dynamic consent
- Auditability
- Traceability
- Non-repudiation

Blockchain is a technology that establishes a network of connected peer nodes that maintain a shared ledger and run smart contracts. The fundamental concept in the architecture is the peer nodes, as shown in Figure 2. Popular networks like Bitcoin and Ethereum allow peer nodes to freely join or leave the network, as they operate as public and permissionless blockchains. While these blockchains strive for extreme decentralization, private entities construct private networks to run their chain. Nodes are bestowed on specified participants to interact with the blockchain. Essentially, organisations participating in the consortium are granted permissions to run peer nodes.

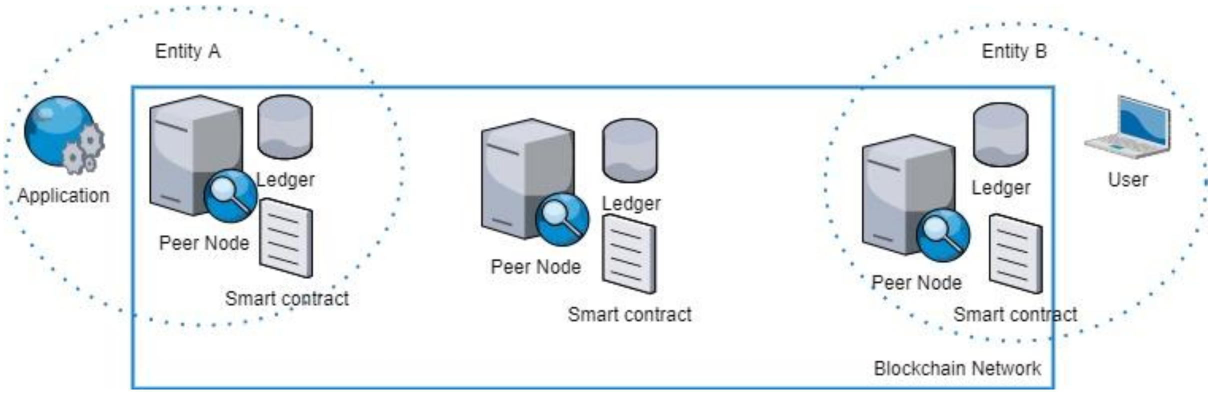

Figure 2: Blockchain overview.

Another concept in the blockchain architecture is the ledger distributed among the peer nodes. In a blockchain, the ledger is a replica of the data, meaning blocks and transactions, and acts as a multiparty database with no central authority. The data are encrypted prior to storing them on the ledger. Additionally, data are immutable as the ledger acts in an append-only way. The ledger is composed of blocks linked to each other, forming a chain. As depicted in Figure 3, blocks have a header with information, such as timestamp and other metadata, defining the block itself, while the block body stores all transactions.

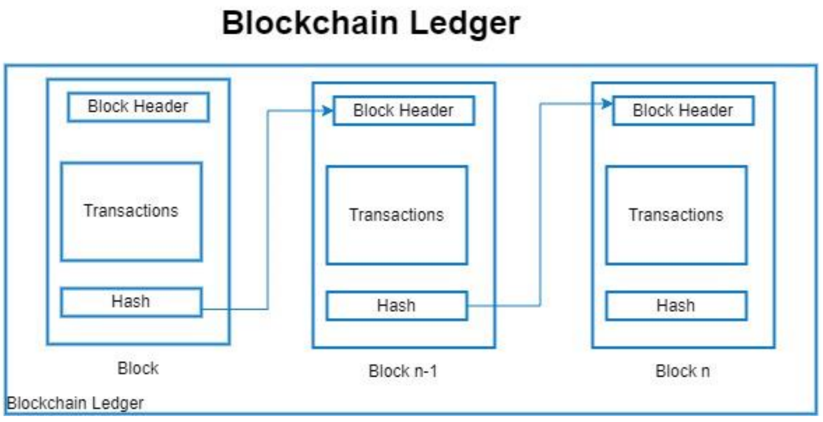

Figure 3: Ledger Overview.

Blockchain has gained popularity for its ability to run logical procedures with the deployment of smart contracts. Smart contracts are trusted applications run on the blockchain, and their output is stored in a transaction on the blockchain. There are numerous options in programming languages to express business logic.

There are more components to consider in the architecture of permissioned blockchain. The network is based on the verification of users, and rights are allocated to the users to perform their activities. A component for handling permissions is vital for this type of blockchain. Established blockchain solutions like Hyperledger Fabric refer to that component as Certificate Authority (CA)[1] (Figure 4). The task of CA is to register identities, issue certificates, and renew or revoke certificates.

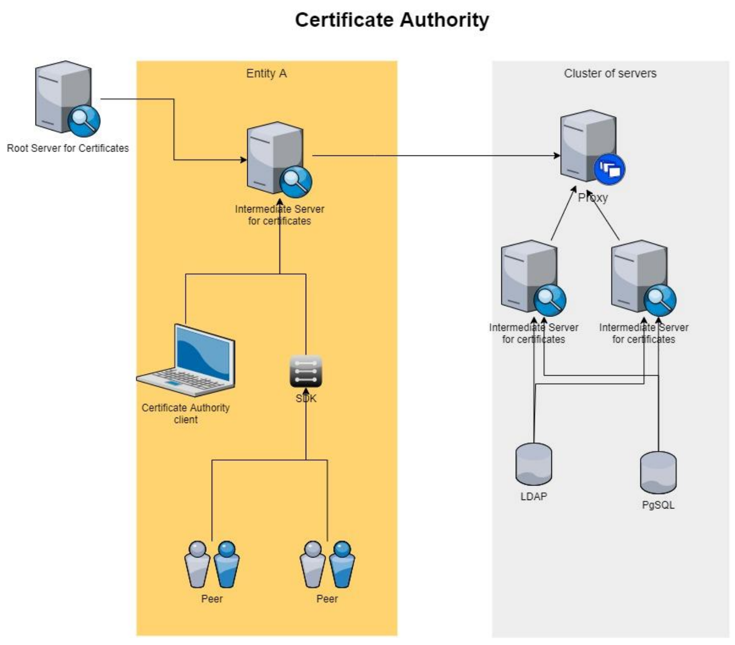
Certificate Authority
Root Server for Certificates
Entity A
Intermediate Server for certificates
Certificate Authority client
SDK
Peer
Peer
Cluster of servers
Proxy
Intermediate Server for certificates
Intermediate Server for certificates
LDAP
PgSQL

Figure 4: Certificate Authority in Hyperledger[1].

[1] Hyperledger Fabric. Fabric CA User's Guide. Source. Accessed on 22 March 2022.

2.3 Applicable Technology

In this section, the existing technologies for implementing the DLT components are examined, and the way they will be applied in the ODIN platform is presented in more detail.

2.3.1 Overview

Before diving into the technology, it is vital to understand the environment and challenges that the developed solution should address. Hospitals manage large volumes of data produced from patients meaning that the data are personal and sensitive. Moreover, hospitals are places where numerous stakeholders offer their services or equipment. The aforementioned aspects constitute an environment with data abundance and multi-stakeholders. Security and trust are important features for the developed platform to deliver. With this in mind, the ODIN project opts for using blockchain technology for achieving security, building trust, and offering the possibility of audit.

Blockchain is defined as a set of data forming a chain of linked data packets, named blocks, where multiple transactions are stored in each block [1]. The addition of new blocks grows the chain, which acts as a common ledger between the participants. The role of encryption is vital in the addition and linkage of the blocks. Essentially, new blocks are added after the approval of the network and are marked with a timestamp. The encryption mechanisms are in place to safeguard the chain's integrity through the first till the last block.

A noteworthy component of the blockchain structure is the consensus mechanism. The network follows a logical process to agree on the ledger's state that involves the network's majority, called consensus mechanism [2]. Generally, there are a plethora of consent mechanisms to select from, depending on the requirements. Notable consent mechanisms include Practical Byzantine Fault Tolerance, Proof of Work, and Proof of Stake.

Blockchain is a general technology that is applied in different sectors due to the benefits that it brings. The benefits from the application of blockchain are [3]:

- Decentralisation: there is no central authority that validates the transactions, as nodes are responsible for adding the blocks;
- Resilience: changes in the data can be detected due to the encryption and compromising the network requires to take control of the majority of the network;
- Anonymity: users are identified with their created addresses in the network, which do not include any identifiable information;
- Control: the transactions are recorded sequentially with the timestamp;

- **Single point of failure avoidance**: a blockchain is active as long as there is a node to support the network's operations.

2.3.2 Available Open-Source Projects

2.3.2.1 Ethereum

Ethereum[2] is a well-known and established blockchain that is decentralised and open-source, with the capability to execute smart contracts. The blockchain application was conceived in 2013 and formally described in a whitepaper. Specifications on the development and the virtual machines were announced in 2014 with the publication of the yellow paper. The Ethereum Foundation has been established as a non-profit organisation supporting Ethereum and similar projects.

Ethereum established a permissionless public network that achieves a consensus on its state by applying a Proof-of-Work algorithm. Miners follow the algorithm to extend the chain with the inclusion of new blocks and are rewarded for successfully completing the task. The consensus algorithm is expected to shift to a Proof-of-Stake with the upcoming update of Eth 2.0.

Ethereum is the infrastructure upon which decentralised applications and smart contracts are built. Essentially, it provides a Turing-complete language with the provision of a virtual system, the Ethereum Virtual Machine (EVM), to execute the code for smart contracts and decentralized applications (dApps). EVM's basic functionality is to abstract the executing code, and the executing machine as software portability is increased. It is possible to develop smart contracts and dApps in numerous programming languages like Go, C++, Python, Java, JavaScript, Ruby, Rust, and others.

2.3.2.2 Hyperledger Fabric

Hyperledger Fabric[3] is an open-source platform for enterprises to apply distributed ledger technology with differentiating features from other popular applications. While the most popular blockchains are public and permissionless, such as Bitcoin and Ethereum, Hyperledger Fabric is able to support the establishment of a network operated by permissioned users. The participants in the network are known entities with credentials to operate in the network. Another differentiating point is the absence of a native currency for incentivising mining and smart contract execution. In this way, cybersecurity hazards are avoided, and the latency in the transaction execution is diminished.

The platform's architecture is highly configurable and modular as other open-source solutions. Multiple sectors can apply the platform for solving open issues in their domain. The platform offers a degree of flexibility in consensus algorithms for the network. Essentially, a project can

[2] Ethereum, https://ethereum.org/en/

[3] Hyperledger Fabric, https://www.hyperledger.org/use/fabric

select a consensus algorithm depending on its specific needs. There are defined components in Hyperledger Fabric's architecture like ordering service, membership service provider, and more for achieving modularity.

The Fabric platform adopts an architectural approach for handling transactions named as "execute-order-validate", contrary to the "order-execute" approach. The suggested approach permits can parallelize the execution of the transactions, as only a subset of the network peer nodes should vouch for committing the transactions.

2.3.2.3 Hyperledger Sawtooth

Hyperledger Sawtooth[4] is introduced as a solution for enterprises to create, deploy, and operate distributed ledgers, additionally referred to as blockchain. Platforms in different sectors are based on the solution for their deployment, as the solution facilitates extreme modularity and flexibility for platforms. The modular approach of Hyperledger Sawtooth allows the network participants to decide on policies to enforce, while the solution's core architecture permits the selection of different transaction rules, permissions, and consensus algorithms. The solution is part of an active project by Hyperledger, the open-source collaborative effort by The Linux Foundation.

Hyperledger Sawtooth innovates in its architectural approach compared to other blockchain-as-a-service (BaaS) solutions due to its modularity. In detail, the application and core system levels are clearly separated to ease the deployment of applications. The built-in smart contract abstraction permits developers to select a programming language to write their logic. The solution offers different transaction processors to address the requirements unique to each application. There is a range of languages, like Python, JavaScript, Go, and more, for deploying the transaction processor SKDs to streamline the contract development in any programming language.

Furthermore, the solution supports numerous consensus algorithms for projects to select the most appropriate to their needs. Classifications of the techniques include Nakamoto-style consensus and variants to the Byzantine Fault Tolerance (BFT) algorithms are general classes of techniques along with innovative algorithms, like Proof of Elapsed Time and Sawtooth Raft, to build the network's consensus. An intriguing feature in Hyperledger Sawtooth is the abstraction of the core concepts of consensus and isolation from transaction semantics. In this way, it is possible to change the consensus algorithm without any significant overhaul to the network's operations.

Another feature of the solution is the ability to batch transactions into single units. The batches are atomic units that change the system's state, which means that a batch and the underlying transactions are either committed or not to the state. The batch approach solves the

[4] Hyperledger Sawtooth, https://www.hyperledger.org/use/sawtooth

dependencies between transactions. Apart from the batches adoption, Sawtooth defines transaction families that group transactions similar to each to allow their inclusion on the ledger.

2.3.2.4 Digital Ledger Technologies solution comparison

A comparison of the main characteristics of the above described technologies is presented in Table 2. All solutions provide a wealth of functionalities. However, the ODIN requirement for a permissioned (private) blockchain network leads to the selection of one of the Hyperledger solutions. The selection between Hyperledger Fabric and Hyperledger Sawtooth will be made during the next period.

Table 2: Comparison of solutions for the Digital Ledger Technologies implementation.

Attribute	Ethereum	Hyperledger Fabric	Hyperledger Sawtooth
Open-source	Yes	Yes	Yes
Smart contract support	Yes	Yes	Yes
Support for private/public blockchain	Public	Private/public	Private/public
Operating system support	All OSs	All OSs	All OSs
Programming language support for execution	High	High	High
Scalability	Medium	High	High
Extensibility	Medium	High	High
Community support	High	Medium	Medium
Ease of adoption	High	Medium	Medium
Performance of consent algorithms (speed, energy efficiency)	High	High	High
Client availability	High	Medium	Medium
Is what is needed	No	Yes	Yes

2.3.3 Usage in ODIN

The ODIN project is to implement blockchain to achieve its objective of connecting resources between hospitals in a secure manner. Essentially, components based on blockchain will deliver different functionalities for handling permission and auditing transactions. The two functionalities are related to each other, since permissions are the cornerstone for audits to provide the specificities.

There is extensive work in permission handling with blockchain as the public grows more aware of the matter, with regulations like General Data Protection Regulation (GDPR) aiding in public awareness. Blockchain in healthcare is used for data sharing with permissions at its core, as smart contracts are responsible for managing access to resources [4][5]. Other studies [6][7] suggest smart contracts for managing clinical trials with smart contracts determining the access rights.

While the literature focuses on data access, data are only a digital asset for hospitals and other digital and physical assets. Assets handled by the ODIN platform also include e-robots, IoT devices, applications, predictive models, and more. Permission handling for all assets follows the same logic as data and is enforced with smart contracts. Entities and resources have identifiers used in managing permissions, while procedures like granting or revoking access and using the resource are documented via transactions on the distributed ledger. Furthermore, the decentralised execution of smart contracts alleviates the risk of a compromised server and its malicious behaviour, as smart contracts are executed on nodes.

As smart contracts will lay the ground for handling permissions, the following process is to support auditing mechanisms. The auditing mechanism will rely on the transactions stored on the blockchain to provide reports on the activities based on the resources. Essentially, the report on transactions can provide the overview for the timeline between granting and revoking access and the transactions taking place in-between. The expectations for a secure system upkeeping the user’s privacy are that no transaction is realised outside of the access period. Apart from the time, the report should prove that only the authorised entity accesses and uses the resource. An additional point to consider is that transactions related to the resource access should document the activity, as access rights are bestowed for specified use, and no other action should occur on the resource. The ODIN platform will deliver the aforementioned functionalities with the application of blockchain.

2.3.3.1 Smart contracts

Smart contracts are defined by IBM [8] as digital contracts stored on a blockchain and executed under meeting predetermined conditions, while Ethereum [9] refers to smart contracts as simply a program. Essentially, smart contracts are logical structures (source code) executed on the blockchain. Smart contracts were initially defined by cryptographer Nick Szabo [10] with the benefits of security and cost reduction. Generally, smart contracts offer benefits in improving speed, efficiency, and accuracy, transparency due to the third party’s absence, security due to their encryption, and cost reduction. Smart contracts have limitations [11][12] that the ongoing research aims to resolve and suggest mitigation procedures.

ODIN project is to structure a blockchain network and make use of the smart contracts to deliver functionalities and automate procedures. The two main functionalities have been described in detail in the overall design, and these functionalities are handling permissions for the authorised entities and the possibility of auditing the activities within the distributed ledger. Overall, smart contracts will facilitate the interactions between the stakeholders in the network, as data and resources have to be used. The smart contracts will enhance security and transparency, as the participants will execute the same logic producing a deterministic output.

As the network will operate as long as a single node is active, the operations should be undisturbed and ongoing all the time.

As smart contracts can be written in different languages, the ODIN project will select a programming language based on the selected blockchain open-source solution and the team experience and familiarity.

2.3.3.2 Flow of operations

The ODIN platform aims to facilitate the cooperation between two entities in the healthcare system. A simplistic overview of the system is composed of two instances, the Keycloak server intervention and the ODIN platform, as depicted in Figure 5. Instances can request to access resources or consume their output to perform their task. The Keycloak server and blockchain intervene between the two instances to secure the communication channel and build trust between the entities.

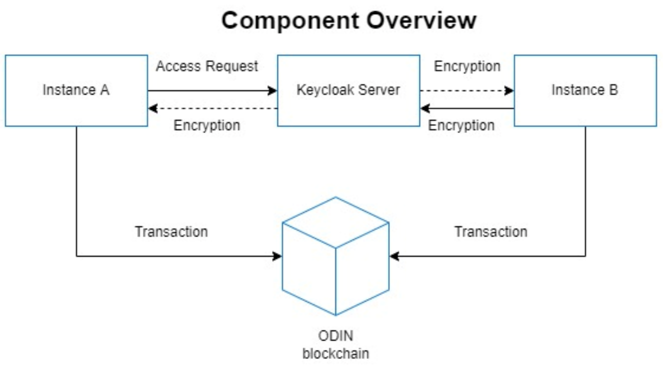

Figure 5: Overview of the interaction between ODIN instances.

Figure 6 depicts the architectural components involved during resource sharing between two ODIN instances. In the example, a backend application in the in instance A wishes to use an AI application accommodated in instance B. The request for resource use is originally made between the API gateways of the two instances. The API gateway of instance B informs the Resource Federation component about this request. The latter decides whether to grant or not permission for access, either using pre-defined consents or after notifying the hospital administrator and waiting for approval. Whatever the result, the request is logged in the ODIN blockchain.

Once permission is granted, the API gateway of instance B opens a direct channel of communication to the exposed API of the requested resource (e.g. the AI service). The two resources can now communicate, with the back-end service of instance A sending data and requesting analysis results from the AI service of instance B, directly through their APIs, but always through the instances' API gateways. Whenever a transaction is made, i.e. an API request, both API gateways inform the Transaction Handling components of both instances of the transaction and the transaction is audited in the blockchain.

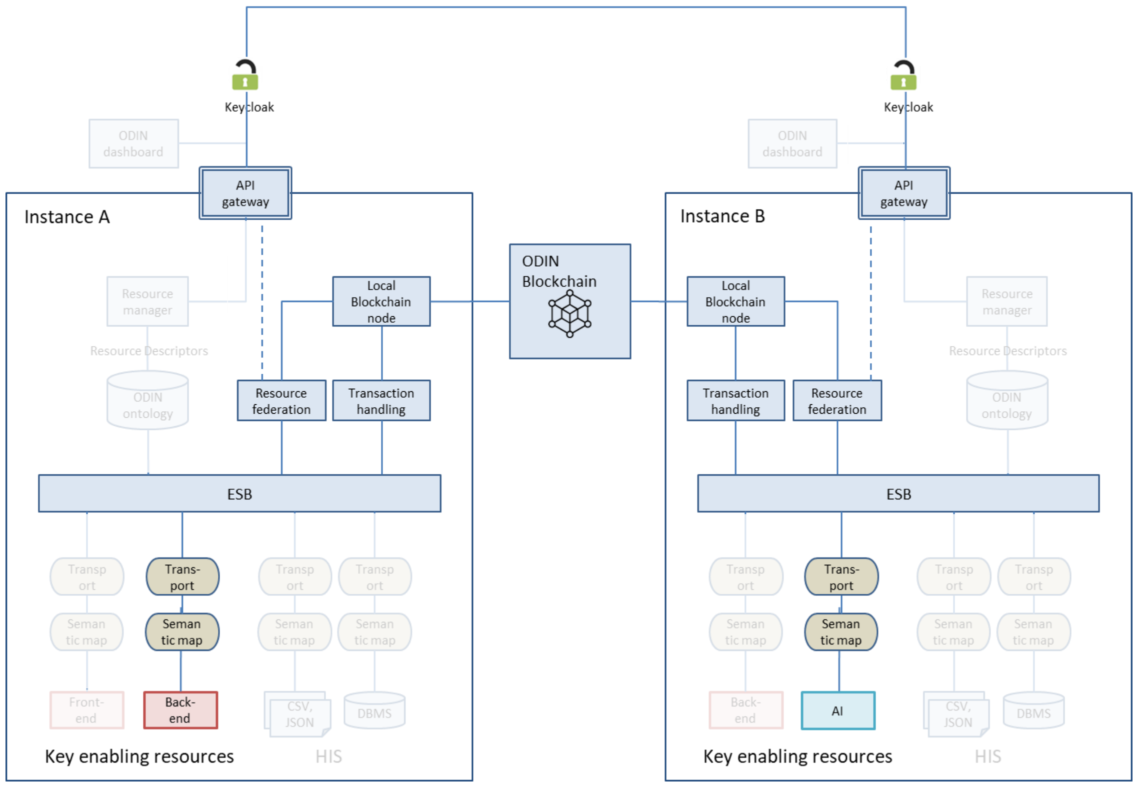

Figure 6: Architectural view of resource sharing between two ODIN instances.

A sequence diagram of the above procedure is depicted in Figure 7, with a detailed description of the actions between two instances.

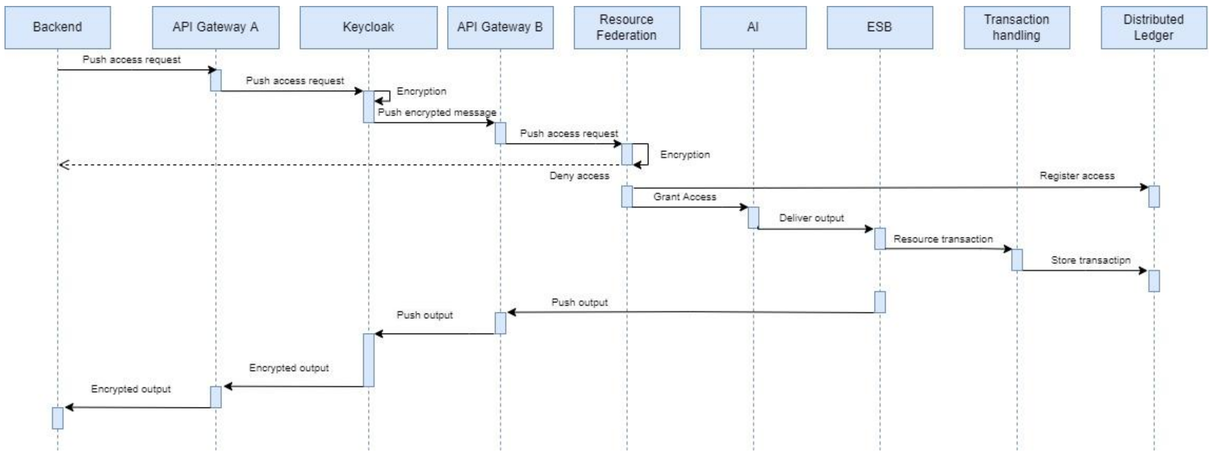

Figure 7: Sequence diagram for an example of resource federation.

The API gateway facilitates the communication between the underlying components within the instance and the rest of the world. In detail, the gateway pushes and receives messages and outputs. The Keycloak Server enhances the security of the communication by establishing communication channels and encrypting messages. The Resource Federation's role is to evaluate the access request, either deny access concluding the communication or grant access and store the access in the ledger. Finally, ODIN's Enterprise Service Bus (ESB) is responsible for tracking the use of the resources and pushing every access to the ledger to be stored as a transaction.

Blockchain's ledger accommodates transactions on the decision and access on the resource. In this way, it will be possible to audit the transactional history of the resource. Essentially, the transactions should be within the time period that the access was granted. The data stored on the blockchain are reliable for extracting insights as no change takes place on the data due to the blockchain's immutability.

2.4 Features Implemented in the First Version

In the ODIN project, the first version for the application of DLT technology will set the foundations to achieve auditability, traceability, and permission handling. Blockchain is an application of the DLT technology and will be implemented for constructing the distributed ledger. Moreover, the blockchain network will be based on the permissioned type as entities will be allocated authorisation certificates to interact with the blockchain.

The aforementioned points are concepts to drive the future blockchain development for the project. In the following steps, smaller and sequential goals are set to guide the development and gradually deliver the platform.

Initially, there are different blockchain services that can be included for structuring the blockchain network. ODIN will opt to create a permissioned network for its consortium blockchain. For that purpose, the Hyperledger Foundation offers different options to choose from. The choice will be based on the matching requirements of the project.

Once the choice on the blockchain solution is made, the developing phase can start establishing the blockchain. The blockchain will be deployed on the ODIN cloud platform striving for undisrupted service availability. Some initial smart contracts will be deployed on the chain as part of the blockchain network. These contracts will handle basic functionalities, with indicative cases being audit logging and transactions recording.

Finally, a vital component for the architecture, to be included in the first version, is the development of an API. The API will be based on the REST (Representational State Transfer) framework for facilitating the communication of ODIN components with the blockchain.

3 Resource Choreographer

3.1 Requirements Review

The Resource Choreographer (RC) is in charge of orchestrating all the resources and services to perform and achieve business logic objectives. Usually in an orchestration, this is performed through rules and workflow engines that execute tasks over the rest of the services and also react to and create events.

The Resource Choreographer not only has to orchestrate the services and resources but allow at least to setup the orchestration templates or workflows. Available tools that offer means to define and design the workflows will be selected. Whenever possible, the Resource Choreographer must delegate the interpretation and leading of a sequence of events or tasks to the services themselves, giving pass to a choreography approach.

The requirements for the Resource Choreographer are reported in more detail in D4.1 "CPS-IoT Resource Management System Specification". A short summary is presented below:

- Functional requirements
    - The RC should enable the coordination, in the form of business choreography services of the available resources accessible from the Resource Gateway and upper layers of ODIN platform.
    - The RC should allow controlling the service orchestration using configuration.
    - The RC should allow listing active scripts to control which processes are happening.
    - The RC should allow viewing which resources are available to be managed in a process choreography so that the process is fully compliant with the resources.
    - The RC should allow the dynamic generation of messages to trigger changes.
    - The RC should allow the dynamic generation of states and tasks to automate orchestration and self-initiate communication among resources intervening in the process.
    - The Resource Choreographer should have the capability of importing/exporting workflows, so that they can be used in other hospitals, or they can be used by other more complex workflows.
- Non-functional requirements
    - The Resource Choreographer should follow an orchestration approach but Choreography should be used when possible.
    - The Resource Choreographer should use formal workflow notation to express the processes compatible with most common used tools.
    - The Resource Choreographer should have a workflow engine to execute the business processes.
    - The Resource Choreographer should allow non-technical users to easily create, edit and apply workflows in a way that are independent of programming language.

3.2 Architecture Review

The positioning of the Resource Choreographer component within the ODIN platform architecture is depicted in the middle part of Figure 1. It is connected to the ESB for communicating with resources at the bottom layer.

Communication with resources means sending the appropriate messages for resources to operate (e.g. commands to a robot) or subscribing/unsubscribing resources to each other using the ESB messaging protocol to guide messages from one resource to another.

Important services or components of the architecture proposed for the Resource Choreographer is the Resource Choreographer itself, as it is not a composed component. On the other hand, as it has been stated in the requirements, if the choreography approach has to be followed, also the rest of the services must implement some sort of functionality to allow it, may be it through the messaging protocol or other mechanisms described in D4.1 CPS-IoT Resource Management System.

The Resource Choreographer depends on the Service Discovery component and the Resource Catalogue, in order to get the list of available resources in the current ODIN instance and their characteristics and configuration. However, a more detailed description of these components is out of the scope of this deliverable since they are not part of the RC itself.

In the following sections, an overview of concrete solutions of existing technologies for resource choreography will be carried on, discussing the best options.

3.3 Applicable solutions for Resource Choreographer

3.3.1 Drools

Drools[5] is an Open-Source Business Rules Management System (BRMS) under RedHat community development. It has a Business Rules Engine (BRE), a web designer and rules management tools (Drools Workbench), in order to support for Decision Model and Notation (DMN) models and an Eclipse IDE plugin in case someone wants to extend Drools core functionality.

DMN is a standard by Object Management Group[6] (OMG) which standardises a conventional and visual notation which any user and persona can understand. It can be compared to BPMN (Business Process Model and Notation), but DMN was created for business decision and BPMN was created for business process management. Drools supports all of the DMN standardised

[5] Drools, https://www.drools.org/

[6] Object Management Group, https://www.omg.org/

components and relations so it can define complex diagrams to express the decision rules. BPMN is also available to create compex Business Processing Management rules. An example Drools diagram can be seen in Figure 8.

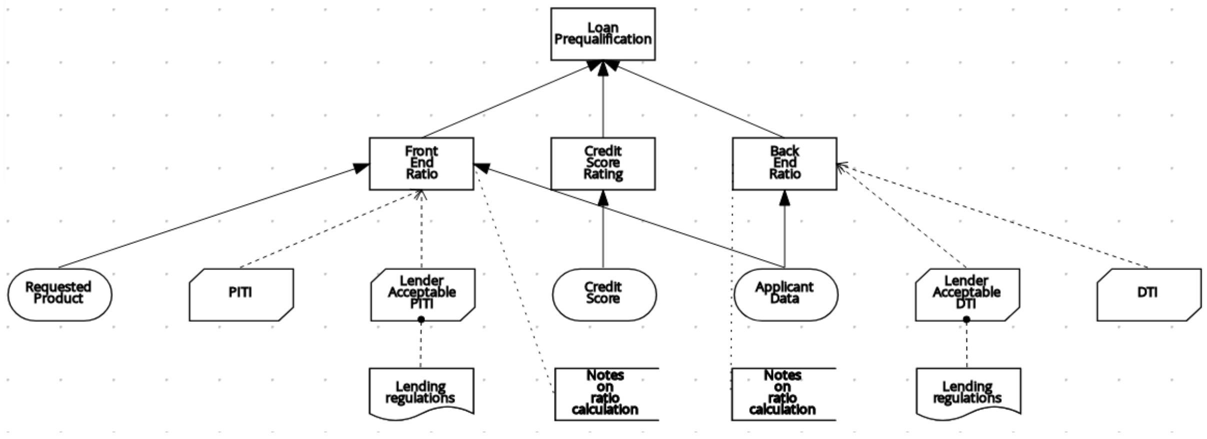

Figure 8: Example of Drools diagram.

DMN can also be expressed using XML language, creating complex definitions for rules and processes and it also has a programming editor using Java. An accompanying tool is the Drools Fusion complex event processor, which in other words, is used for detecting and selecting the interesting events from an event fog, finding their relationships, and inferring new data from them and their relationships. Using Drools, a wide range of applications can be developed to control the logic.

3.3.1.1 Characteristics

Security

Drools delegates its security to the container application where it is deployed. Usually this is handled by the Kie Server which uses a Wildfly web server that can delegate the authentication and authorization services for example to a Keycloak Server to control who and which applications can interact with Drools deployed services. This integration eases and decouples the functions and responsibilities, but forces the developer to take into account the implementation details and the configuration.

Metrics

Drools has several kinds of metrics. First it is compatible with Wildfly monitoring tools, so it is easy to be integrated using JMX to read base metrics from JVM MBeans and from the management model subsystem and deployment subtrees.

Through the KIE Server, it offers integration to Prometheus and Graphana which also help reading the metrics about the execution of business rules, processes, Decision Model and Notation (DMN) models, and other Drools assets.

With those tools, it is easy to control the infrastructure status. On the other hand it was expected to have profiling tools to measure the performance of the processes but currently those measures have to be done using concrete tests.

Deployment and setup

Drools is available to be deployed in Kie Server as many other Red Hat projects, which at the end fall into a Wildfly server. On the other hand a typical Drool setup can be packed into a Docker image to be used with Docker Composer or other orchestrating tools such as

Kubernetes. The problem around Drools is that each image is very big, so although it can work in clusters, the amount of resources wasted is considerable.

Beyond the deployment, Drools is a tool that it is not easy to manage or develop with. Most of the work has to be carried through Drools Workbench to create a workflow, and then load the workflow using Kie Server. This task, which is quite common, can take a lot of time, so it stresses operational tasks that may be quite slow.

3.3.1.2 Usage in ODIN

Drools can be used to deliver an orchestrator for concrete tasks that must be handled by several services. With Drools, a workflow can be deployed that upon some events thrown by users or other services, it can drive the sequence of processes that must be taken to achieve higher level business objectives.

Drools covers all the needs requested from the requirements. Tasks, data types, states, events, commands, and other interactions can be modelled using Drools to drive the processes. The main concern about Drools it is the learning curve, and operational management.

3.3.2 Kogito

Kogito[7] is a cloud-native oriented business automation solution for building intelligent applications. It is also an open source project under the Red Hat umbrella, and it can be seen as an evolution of Drools. Kogito derives its name from K of Kubernetes, which is its base environment to be deployed and from “Cogito” as in "Cogito, ergo sum" ("I think, therefore I am"). Kogito uses a cloud minded approach, so cloud-based solutions can be used with it, such as the ones shown in Figure 9.

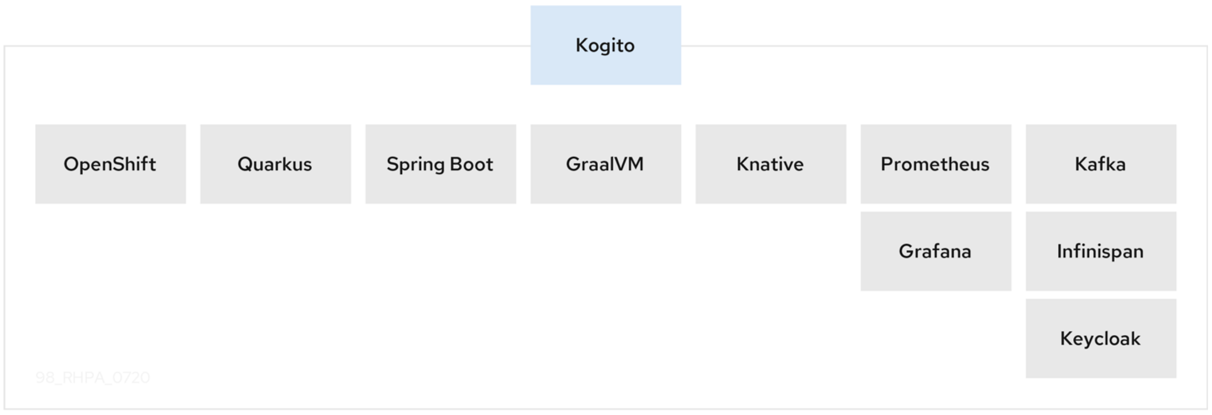

Figure 9: Solutions compatible with Kogito.

[7] Kogito, https://kogito.kie.org/

Kogito includes a graphical BPMN and BMN modeller with the suite of applications, easing the tasks to deploy a workflow. Moreover, Kogito includes support for Serverless Workflow specification[8] which supports declarative (no-code) specification of rules to orchestrate micro services. It is focused on orchestrating specific functions and is less complex and powerful than BPMN.

Kogito can be very useful in some scenarios such as the ones depicted in Figure 10. In the example of this figure, an order can be created from a custom UI or a messaging integration from another service. The order is dropped to a Kafka bus, which trigger new Serverless Workflow instances that evaluate the order. In this case, the Serverless Workflow just evaluates if the order can be auto approved or requires human interactions. Depending on the result, the order is forwarded to the Rule Service or the Human Task Service.

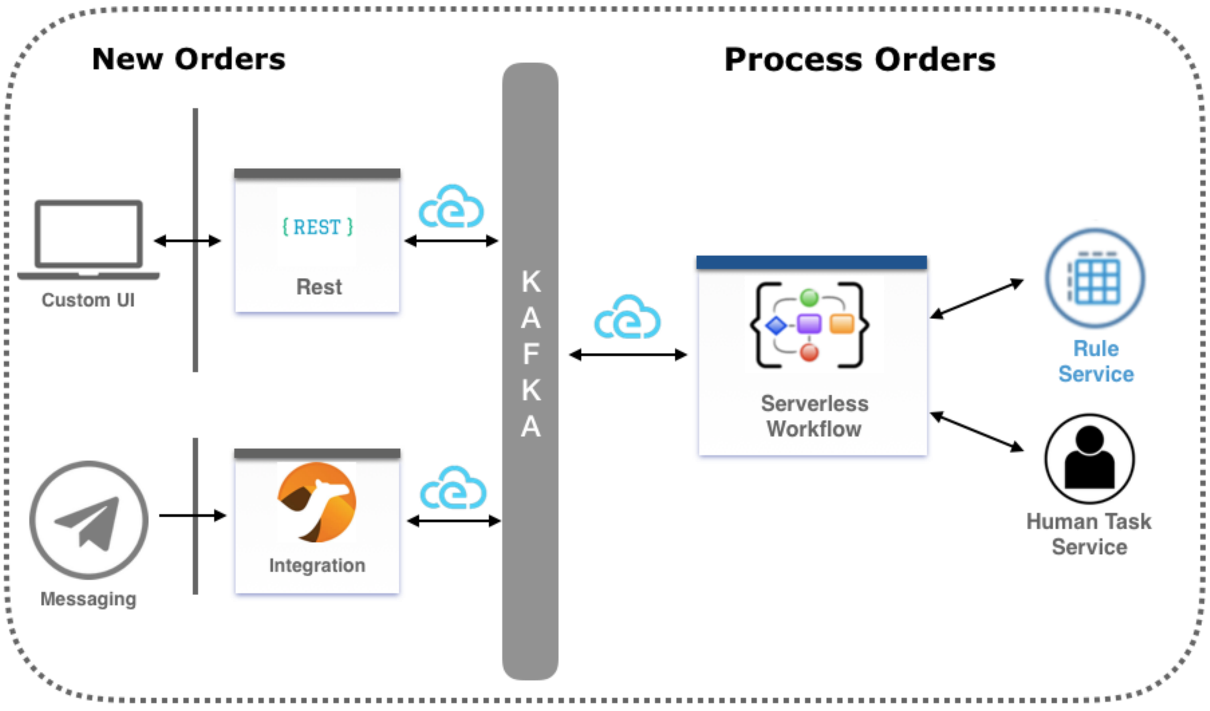

Figure 10: Serverless workflow.

As a Decision Model Notation engine, Kogito can use BMN specification, decision rules, decision tables and other kinds of graphical representation to deliver its power. Indeed Kogito supports Drools rule language to implement decisions. Processes are another strong point to be

[8] Cloud native computing foundation, https://www.cncf.io/

modelled using Kogito, as it supports BPMN 2.0, workflow design using Kogito BPMN Modeller, with support of Visual Code, or of importing a workflow from another tool.

3.3.2.1 Characteristics

Security

Security is assured with authorization and authentication mechanisms such as OAUTH, and it is also possible to delegate the security into Kubernetes and Keycloak solutions. This eases handling security attributes and decouples the business logic from the security.

Metrics

Kogito shines where Drools falls, that is in operational metrics. Kogito support Prometheus integration to deliver a full range of metric collection of Business Process Model and Notation process models, business rules, and Decision Model decision models. Access to the stored metrics is granted through a REST API, also using the Prometheus expression browser, or using a data-graphing tool such as Grafana.

To mention a few, Kogito supports:

- Total number of created instances
- Total number of running instances
- Total number of completed instances
- Total number of instances that violated SLAs
- Average process execution time, counted as the delta between the completion instant and the creation instant

Those KPIs allow business managers to act over the situations to counter problems and deliver the best of the business. Common metrics are also available for infrastructure and service monitoring.

Deployment and Setup

Kogito in cloud enabled from the roots. It is designed to be deployed directly into OpenShift, which is a concrete solution of Kubernetes, so Kubernetes is also supported. Kogito supports all the deployment tasks through its operators in a guided fashion, but it also support for a Command Line Interface to perform all the tasks.

Using this approach, all the Kogito services are deployed as pods into the cloud infrastructure that can be monitored to ensure its good behaviour in the cloud using probes.

Beyond the deployment, the configuration allows to tailor specific services to be running.

- Kogito supports the following key middleware infrastructure services:
  - Infinispan persistence
  - Apache Kafka reactive messaging
- Kogito also provides the following dedicated services:
  - Kogito Data Index Service indexing and querying
  - Kogito Jobs Service job scheduling
- The Kogito runtime supports various configuration options for these supporting services and for other capabilities, such as the following examples:
  - Custom event listeners

- Prometheus metrics monitoring
- Process instance management

3.3.2.2 Usage in ODIN

Since Kogito covers all functionalities offered by Drools and offers even more, it can be used in ODIN as the technology behind the Resource Choreographer implementation. The metric collection functionalities can be especially useful for monitoring the running processes and examining possible optimizations.

3.3.3 Resource Choreographer solution comparison

In the following table, a comparison among the solutions is performed in order to discuss which one is better for implementing the Resource Choreographer, in scales from 1 (poor) to 10 (great).

Table 3: Comparison of solutions for the Resource Choreographer implementation.

Specific requirement	Drools	Kogito	Reason
Is what is needed	8	10	Due to the Kubernetes infrastructure for the ODIN platform, Kogito is closer to the solution needed. Moreover, Kogito has more tools to fill several needs.
Ease of adoption	4	8	Mysphera developers and support team have reported the difficulty to learn and manage Drools over Kogito. Operational tasks are far easier with Kogito. For example, loading a workflow may take 4 hours using Drools, stopping services and with high risk of errors, while Kogito just takes 2 hours without stopping services.
Scalability	5	10	Kogito is cloud and elastic oriented while Drools image wastes lots of resources.
Reliability	6	10	Both can implement redundancy but Drools cannot take the same advantage as Kogito in terms of service monitoring.
Security (Kerberos, OAuth, etc.)	10	10	Both can use Keycloak.
Data integration	5	10	Kogito supports Kafka integration out of the box.
Support	5	8	Drools documentation is quite complex and bad organized.

			Moreover, most of the Droll project seems frozen as there is not much activity in the web page or the development team. We assume the Drools team has focused on Kogito which has more activity and support.
Deployment facilities	5	10	Kogito has operators to deploy its services over Kubernetes. Drools is a do it yourself path.
Size of project to be used	5	10	Kogito is aimed for mid to big projects where a big infrastructure using Kubernetes must be used. Drools is aimed to small to mid-sized projects.
Cost	10	10	Both are free.
Licence	10	10	Both are open-source.
Number of projects using it	-	-	No information about the number of projects found.
Number of languages	1	1	Java
Monitoring	7	10	Kogito has support for application and business monitoring while Drools does not.
Hard dependencies on other projects or solutions	2	-	Drools depends on Kie server and Wildfly server.
Innovation impact	8	10	Using any of the solutions as service orchestrator can be an innovative approach, but Kogito is newer and its service approach is most innovative.

The comparison leads to a clear result, that Kogito is more desirable for the ODIN purposes.

3.4 Features Implemented in the First Version

For the first version of the platform, the list of features to be implemented are:

- A list of the workflows that should be controlled by the Resource Choreographer
- A list of the situations or interactions that could be solved using choreography by the services without Resource Choreographer interaction.
- The definition of the messages, states, tasks and data types to be used to orchestrate the services with the RC.
- The definition of the messages to be used to choreograph the services

- The deployment and setup of the tool selected.

4 Conclusion and next steps

Two of the major objectives of the ODIN project are decentralization of the platform implementation and narrowing the gap between resource providers and healthcare organizations. The components described in this deliverable attempt to address these two objectives:

- The Digital Ledger Technologies component provides a secure and trustworthy substrate for resource sharing between remote ODIN instances, enabling resource decentralization. At the same time, it facilitates the introduction of external components (e.g. developed by open callers) or new ODIN instances, both from a technological and a regulatory point of view.
- The Resource Choreographer component provides a way for users to design custom application workflows that make use of the available resources and target specific use case scenarios. This narrows the gap between resource providers and healthcare organizations, singe new resources can readily be embedded in new applications designed by the healthcare authorities.

This deliverable describes the design of the two components, covering requirements, architectural elements, functionality and technology survey. Next steps include the actual implementation of the first versions of these components, by adopting the selected technologies and developing the core elements that allow the operation of the components in the ODIN platform. These developments are the focus of the second year of the project and will be reported in the second version of this deliverable (D4.6), while development will be concluded after further refinements in the third version of the deliverable (D4.7).

In parallel to the implementation, next steps also include the definition of use cases and scenarios in which the DLT and RC components play a significant role. With regard to DLT-based resource federation, scenarios will be examined in which hospitals of different countries (e.g. France and Denmark) share resources by exploiting existing DLT cross-border smart contracts that the NHSs of these countries may have. With regard to resource choreography and orchestration, scenarios in which choreography actively helps in improving hospital operations will be examined, e.g. by adapting and exchanging workflows across hospitals, perhaps through a kind of marketplace.

References

[1] Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. Business & Information Systems Engineering, 59(3), 183-187.

[2] Swanson, T. (2015). Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. Report, available online.

[3] Monrat, A. A., Schelén, O., & Andersson, K. (2019). A survey of blockchain from the perspectives of applications, challenges, and opportunities. IEEE Access, 7, 117134-117151.

[4] Theodouli, A., Arakliotis, S., Moschou, K., Votis, K., & Tzovaras, D. (2018, August). On the design of a blockchain-based system to facilitate healthcare data sharing. In 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) (pp. 1374-1379). IEEE.

[5] Azaria, A., Ekblaw, A., Vieira, T., & Lippman, A. (2016, August). Medrec: Using blockchain for medical data access and permission management. In 2016 2nd international conference on open and big data (OBD) (pp. 25-30). IEEE.

[6] Benchoufi, M., & Ravaud, P. (2017). Blockchain technology for improving clinical research quality. Trials, 18(1), 1-5.

[7] Nugent, T., Upton, D., & Cimpoesu, M. (2016). Improving data transparency in clinical trials using blockchain smart contracts. F1000Research, 5.

[8] IBM. What are smart contracts on blockchain? Source. Accessed on 18 March 2022.

[9] Ethereum. Introduction to smart contracts. Source. Accessed on 18 March 2022.

[10] Szabo, N. (1997). Formalizing and securing relationships on public networks. First monday.

[11] Mik, E. (2017). Smart contracts: terminology, technical limitations and real world complexity. Law, Innovation and Technology, 9(2), 269-300.

[12] Nzuva, S. (2019). Smart contracts implementation, applications, benefits, and limitations. School of Computing and Information Technology, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya.

Appendix A Acronym glossary

Acronym	Definition
AI	Artificial Intelligence
API	Application Programming Interface
BFT	Byzantine Fault Tolerance
BMN	Business Modeling Notation
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
BRE	Business Rule Engine
BRMS	Business Rules Management System
CA	Certificate Authority
CPS	Cyber Physical System
dApp	Decentralized Application
DLT	Digital Ledger Technologies
DMN	Decision Model and Notation
ESB	Enterprise Service Bus
EVM	Ethereum Virtual Machine
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
IDE	Integrated Development Environment
IoT	Internet of Things
JMX	Java Management eXtensions
JVM	Java Virtual Machine
KER	Key Enabling Resource
OAUTH	Open Authorization
OMG	Object Management Group
RC	Resource Choreographer
REST	Representational State Transfer
UI	User Interface
XML	eXtensible Markup Language