
**Call identifier:** H2020-ICT-2016 - **Grant agreement no**: **732907**

**Topic**: ICT-18-2016 - Big data PPP: privacy-preserving big data technologies

# Deliverable 6.10

# Blockchain Analytics (3)

Due date of delivery: December 31th, 2019

Actual submission date: December 31th, 2019

**Start of the project:** 1st November 2016
**Ending Date**: 31st December 2019

Partner responsible for this deliverable: Gnúbila

Version: 1.0

**Document Classification**

| | |
|---|---|
| Title | Blockchain Analytics 3 |
| Deliverable | D6.10 |
| Reporting Period | 2nd |
| Authors | Alexandre Flament, Mirko Koscina, Jérôme Revillard |
| Work Package | WP6 |
| Security | Public |
| Nature | Report |
| Keyword(s) | |

**Document History**

| Name | Remark | Version | Date |
|---|---|---|---|
| Jérôme Revillard | Incremental update of the D6.8 | 1.0 | October 10th 2018 |
| Jérôme Revillard | Incremental update of the D6.9 | 1.0 | December 12th 2019 |

**List of Contributors**

| Name | Affiliation |
|---|---|
| Alexandre Flament | Gnúbila |
| Mirko Koscina | Almerys |
| Jérôme Revillard | Gnúbila |

**List of reviewers**

| Name | Affiliation |
|---|---|
| Mirko De Maldè | Lynkeus |
| | |

**DISCLAIMER : This document is based on the already submitted D6.9. It includes a new section (section 7.9) describing current limitations and future work.**

# 1 Scope of this document

This document corresponds to the Deliverable D6.10 Blockchain Analytics (3), based on the work done in T6.2 and T6.3. D6.10 is an update of the previously submitted D6.9 and will illustrate the blockchain usage thanks to illustrative and eye-catching analytics. It will serve the purpose of feeding the project Website and marketing materials.

The task *T6.2 Blockchain Assessment, Prototyping and Integration* consist into assess existing blockchain technologies and select an appropriate candidate for implementation. Particular emphasis will be brought to Ethereum and the Hyperledger technologies. Scalability will be a key point for the proposed infrastructure and several data sharing, data indexing and (in memory) parallel processing techniques will be considered.

On the other hand, the task *T6.3 Blockchain Infrastructure Deployment and Test* consist in the deployment of the technology selected in task T6.2 over the physical infrastructure of participating centres. More specifically, hospitals such as OPBG, Charité, UCL/GOSH will be installed and parameterized as the pillar nodes of the system. Others will join at a later stage as the network will propagate in the community. A software package will be released periodically that will allow external centres to join in. It will comprise the blockchain mining service, API, the Data Catalogue (PID indexing) and core libraries from WP4. Once the blockchain infrastructure is deployed, scale-up tests will be operated with synthetic data also from WP4, to validate the overall infrastructure robustness, performance, and an equation with the initially identified requirements.

This document is organized in seven sections. The section 1 corresponds to the scope of the document and section 2 is an introduction about what is blockchain and how we can obtain data from it. Then in Section 3 we describe the components of Hyperledger involved in transaction flow and data storage. In Section 4 we introduce the concept of chaincode to manage and develop applications on the blockchain. Then, in Section 5 we introduce a description of our implementation to feed the data analytics module.

In Section 7, a new tool named the Blockchain Explorer is presented: it provides different information about the blockchain content.

# 2 Introduction

A blockchain system is a distributed ledger where all the transactions taken place in the network are stored in blocks. These blocks are sorted and added to the network one by one forming a chain of block. The system is decentralized because the ledger is replicated within the nodes participating in the processing services. The security properties of this decentralized system are based on cryptographic techniques that make this chain of blocks immutable. The new blocks generation and the ledger replication process within the node of the network is governed by the consensus algorithm defined in the network. The consensus algorithm uses cryptographic functions and business rules in order to decide which node will add the next block and the replication of the last blockchain state to the rest of the nodes.

In any system it is crucial to know the current status of the infrastructure and how this is being used (operations, users, queries, among others) in order to measure the performance of the service that is running on it. In the case of a decentralized system, this can be more tricky than a central system because the status will depend on the nodes and there are several characteristics that make it more complex. In the case of blockchain, to measure the different status of the ledger will depend on multiples components that gives life to the decentralized system.

A simple query to get information from a block implies that we must connect to a specific channel (ledger) and then find the block. In the case of transactions, these operations can be more complex because it will depend on the status. If we would like to know the status of a simulated transaction, endorsed transaction or the committed transaction, we need to take in care the different modules of Hyperledger Fabric, such as: consensus (responsible of the transactional confirmation process and block generation), the state database to know the current status and membership service to know who is valid to perform specific instructions.

Hyperledger Fabric gives us framework with standard instructions to operate over the network called Chaincode. This is the one of the key elements of the blockchain implementations because it gives us a simple interface to communicate with different nodes and channel in the decentralized system. In addition, we have a Hyperledger Composer that is a suite to manage and develop new applications for the blockchain in a friendly user environment.

In the following sections we will describe the concepts introduced above and how all the components can interact by using chaincode and hyperledger composer.

# 3 Hyperledger Fabric Blockchain

Hyperledger project was founded in 2015 by The Linux Foundation to advance the blockchain technologies for multiple industries. The main goal is to facilitate and encourage the development of blockchain implementation beyond the widely known cryptocurrencies.

Hyperledger Fabric is a blockchain framework that let us implement smart contracts inside this private and permissioned ledger. The smart contracts are used to provide controlled access to the ledger. Using smart contracts, it is possible to encapsulate information and store it across the network and define business rules that can execute transactions automatically. The permission of the ledger is reached by using an enrolment service called Mermbership Service Provider (MSP). This facilitates the control over the network and also gives flexibilities for the consensus algorithm because of the lack of need to prove the honesty of the nodes by using expensive techniques like Proof-of-Work. The identity manager administrates the user's IDs and authenticates all participant members of the network. This service permits to parametrize different layers of permission of specific network operations, making possible to allow or block some users to invoke or develop new operations into the network. Additionally, Hyperledger Fabric offers privacy and confidentiality in the network by using private channels. These are restricted messaging paths that can be used to provide privacy for a specific group of users into the network.

The smart contract triggered in the network are represented as intangibles assets by the decentralized system. These assets are managed as a collection of key-values pairs, with state changes recorded as transactions on a channel ledger. The operations, control or modifications of these assets are governed by the chaincode. The chaincode corresponds to small programs that runs specific instructions in the ledger. With these we can manage network rules, configuration parameters, query the ledger, among others.

## 3.1 Hyperledger Fabric Components

Hyperledger Fabric is a comprehensive framework to implement customizable business blockchain networks. The Fabric model is based on six components: *Assets, Chaincode, Ledger, Channels, Security and Membership Service,* and *Consensus* [1]. All these components together make the full blockchain solution.

### 3.1.1 Assets

Hyperledger Fabric assets can range from intangibles like contracts and intellectual property to tangible like real estates or any goods. These assets can be modified by using special chaincode transactions. In Fabric, the assets are defined as a collection of key-value pairs with state changes recorded as transaction on the ledger.

### 3.1.2 Chaincode

Chaincode is an application or code that lets the administrator of the business blockchain define the assets and the operations (transaction instructions) to modify them. Chaincode functions runs over the current state database and is initiated through a transaction. As a result of the Chaincode execution, we obtain a set of key value writes to submit to the network and be applied to the ledger on all the nodes.

### 3.1.3 Ledger

The blockchain ledger is a sequence of tamper-resistant record of all the state transactions in the system. The state transitions are produced by transactions (invocations) submitted by members of the network. There is one ledger per channel and each node store a copy of the ledger of which they are participating.

Within the ledger we can find the following features [1]:

- Query and update ledger using key-based lookups, range queries, and composite key queries
- Read-only queries using a rich query language (if using CouchDB as state database)
- Read-only history queries - Query ledger history for a key, enabling data provenance scenarios
- Transactions consist of the versions of keys/values that were read in chaincode (read set) and keys/values that were written in chaincode (write set)
- Transactions contain signatures of every endorsing peer and are submitted to ordering service
- Transactions are ordered into blocks and are “delivered” from an ordering service to peers on a channel
- Peers validate transactions against endorsement policies and enforce the policies
- Prior to appending a block, a versioning check is performed to ensure that states for assets that were read have not changed since chaincode execution time
- There is immutability once a transaction is validated and committed
- A channel’s ledger contains a configuration block defining policies, access control lists, and other pertinent information
- Channel’s contain Membership Service Provider instances allowing for crypto materials to be derived from different certificate authorities

### 3.1.4 Channels

Channels are used to maintain subnets in the blockchain implementation. Each channel will have one ledger with specific business rules. Fabric permits to configure the system according the business rules of the service, letting set one common shared ledger for all the members or more than one with restricted access to a selected group of members.

The implementation of multiples channels is used to isolate the transactions and the ledgers. This can be done by installing chaincode only on peers that need to access the asset state to read and/or write into the private ledger. Additionally, it is possible to encrypt the data by using common encryption scheme with their correspondent secret keys.

### 3.1.5 Security and Membership Service

Hyperledger Fabric is a blockchain implementation where all the members are known participants. In order to enrol and validate members, a Certificate Authority (CA) is implemented based on a Public Key Infrastructure (PKI) environment. The CA is responsible of the digital certificate issuance process for each member of the network. The certificates tied organization, network components, and end users or client application. With this scheme it is possible to govern the access to the entire network and on channel level by authenticating members according their digital certificate.

### 3.1.6 Consensus

The consensus algorithm is one of the backbone of any blockchain implementation due to their responsibility in the new blocks addition and the chain replication. In the case of Hyperledger Fabric, the consensus algorithm takes part in the entire transaction flow from the proposal and endorsement, to the ordering,

validation, and commitment. The consensus is achieved when the transactions have been ordered and passed through series of explicit policy criteria checks. These checks are part of the transaction lifecycle and include endorsement policies to establish which members must endorse the transaction (according their type), as well as system chaincodes to ensure the policies enforcement. Also, system chaincodes are used to validate that the transaction has been endorsed properly prior to commitment. Moreover, there is a versioning check where the ledger current state is agreed before adding a new block to the chain. In addition, there are identity verifications during all the transaction flow.

### 3.2 Chain

The chain corresponds to a transaction log where each transaction is recorded in block [2]. These blocks contain N transactions and are sorted in sequences. Each block is linked to its predecessor by adding the transaction hash value of the previous block in the header. In addition, the hash value of the transaction recorded in the block is also stored in the header. This permits to maintain the chain ordered in sequence and linked between consecutive blocks by using cryptographic functions.

The chain structure mentioned above is the responsible of the ledger immutability property. This means that the hash link makes that any change in a recorded transaction will affect the block hash value. This forces the reconstruction of the entire chain from the block where the change was made until the last block.

### 3.3 State Database

The state database is an indexed view of the transaction log and can be recovered at any time or automatically upon the peers started up [2].

The current state database corresponds to the latest values for all keys ever included in the transaction log. It is also known as World State because it represents all the latest key values known to the channel. The current state database is stored in the state database to make more efficient the operations from the chaincode. Within the databases supported are levelDB and couchDB.

### 3.4 Transaction Flow

The transaction flow starts when an application client sends a transaction proposal to an endorsing peer. The endorsing peer verifies the digital signature of the client, and then simulates a transaction by executing a chaincode function. From this process we get a set of key/value versions that were read in the chaincode (read set) and a set of key/value that were written in the chaincode (write set). The transaction proposal response is sent back to the client with the endorsement signature of the endorsing peer. Then, the client broadcast a transaction payload with the endorsement to an ordering service. This service is responsible to broadcast the ordered transactions to all the peers on the channel.

Once the peers have received the transaction, they will validate it according to the endorsing policy of the channel and will also authenticate the signatures with respect to transaction payload. This process ensures that the correct allotment of specific peer has signed the result of the transaction proposal. In addition, peers check the versioning of the read set in order to ensure the data integrity and protect against double-spend attack. Finally, for each valid transaction the write sets are committed to the current state database and the block is appended to the chain [2].

In Figure 1 we can see the transaction flow of Hyperledger Fabric [3].

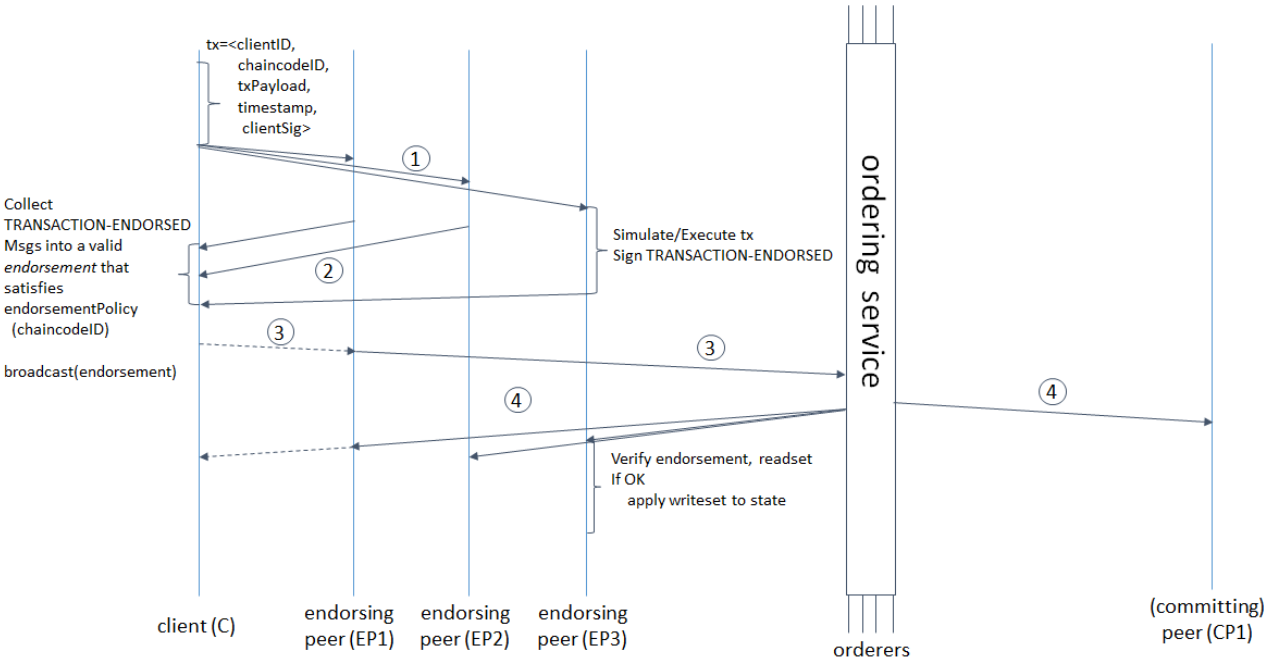



*Figure 1: Hyperledger Fabric Transaction Flow (copyright Linux Foundation – Hyperledger Project)*

# 4 Chaincode

Chaincode is an application that manages the business logic of the network [4]. From the practical point of view, chaincode can be considered as a smart contract because it is a business logic agreed by the network members. The chaincode runs in a secured Docker container isolated from the endorsing peer process and it initializes and manages the ledger state through the transactions sent by the applications.

The chaincode has two perspectives, one for developers and the other for operators. The Chaincode for Developers is focused on the application development to add or control functionalities customized by the implementer. On the other hand, the Chaincode for Operators is oriented to manage the blockchain network.

## 4.1 For Developers

The chaincode programs are developed in Go, Java or Javascript languages and must implement the chaincode interface [5]. These interfaces are methods called in response to received transaction. In the case of the Init method, this is called when a chaincode receives an Instantiate or Upgrade transaction. With this instruction the chaincode performs any necessary initialization (including initialization of the application state).

On the other hand, the Invoke method is called in response to an invoke transaction to process the transaction proposal. In addition, there is a chaincode interface called ChaincodeStubInterface, which is used to access and modify the ledger, and to perform invocations between chaincodes.

## 4.2 For Operators

The chaincode for operations permits to administrate the system by using an API that can accessed by command line. The commands to manage the chaincode lifecycle are: package, install, initiate and upgrade. Once the chaincode is installed and initiated, it remains active to process transactions via invoke.

Additionally, the system chaincode runs within the peer process instead of running in an isolated container like normal chaincodes. These chaincodes are used to implement a number of system behaviour that can be modified or replaced by the system administrator. Currently, we have available the following system operations [6]:

- LSCC: Lifecycle system chaincode handles lifecycle requests described above.
- CSCC: Configuration system chaincode handles channel configuration on the peer side.
- QSCC: Query system chaincode provides ledger query APIs such as getting blocks and transactions.
- ESCC: Endorsement system chaincode handles endorsement by signing the transaction proposal response.
- VSCC: Validation system chaincode handles the transaction validation, including checking endorsement policy and multiversioning concurrency control.

For example, In Figure 2 we can see the chaincode swim-lane [7] of a transaction process in Hyperledger Fabric.

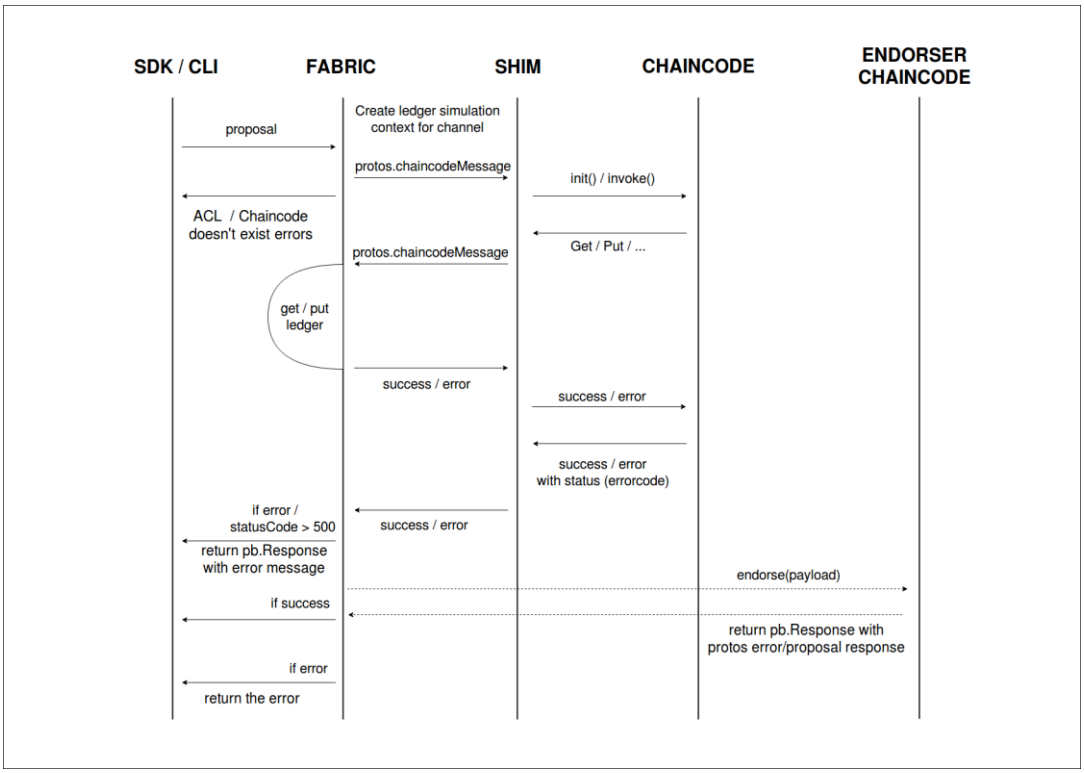



*Figure 2: Hyperledger Fabric Chaincode Swimlane (copyright Linux Foundation – Hyperledger Project)*

# 5 Implementation

The implementation of the module for data analysis for MHMD is based on an integration between the different components of the ledger. The assets represent the information that we have recorded in the ledger, the channel is the ledger to query, the database is the current status of the chain, the chaincodes are the functions that let us query the ledger and the transactions are the triggers to execute some operation on the chain.

To get access to the different components of the ledger and create a new business logic for this purpose, it is necessary a development platform to make simple the management and development on the blockchain. The tool selected to implement the programs to feed the data analytics module is Hyperledger Composer. With this tool we will operate the ledger and we will also develop the smart contracts and the business logic to feed the analytics module.

## 5.1 Hyperledger Composer

Hyperledger Composer is a comprehensive toolset to develop blockchain applications [8]. This suite let us develop an API that runs some business rules to get specific data to feed it to the data analytics module. Also, it is managed by Composer the access to different queries in order to maintain the restricted operations on the private channel (ledger). In Figure 3 we can see how Composer interacts with the Hyperledger Blockchain.

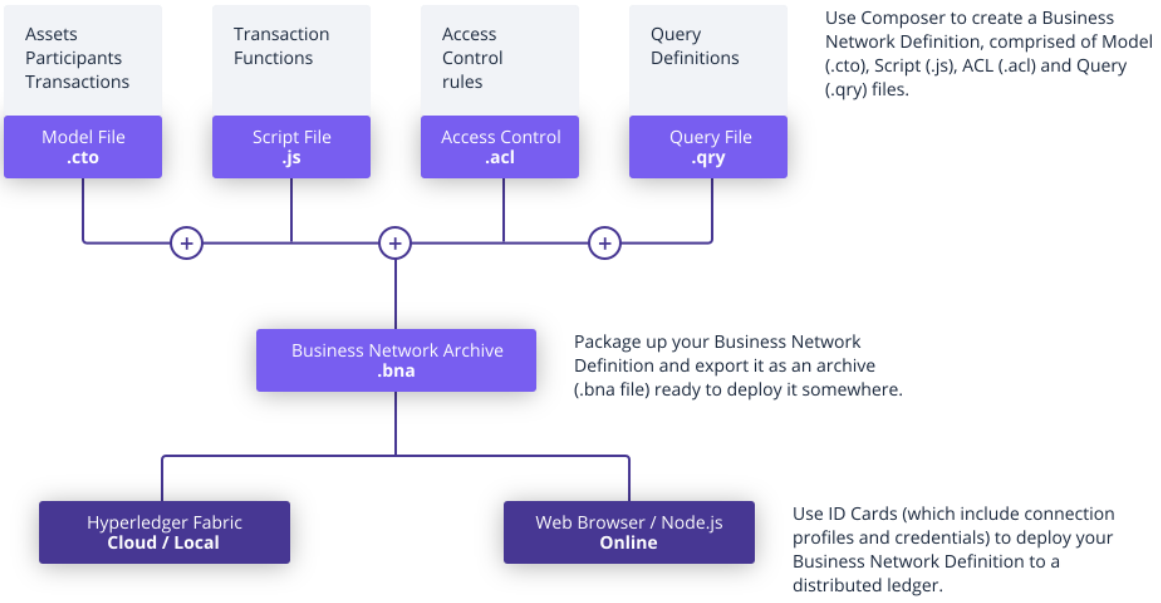
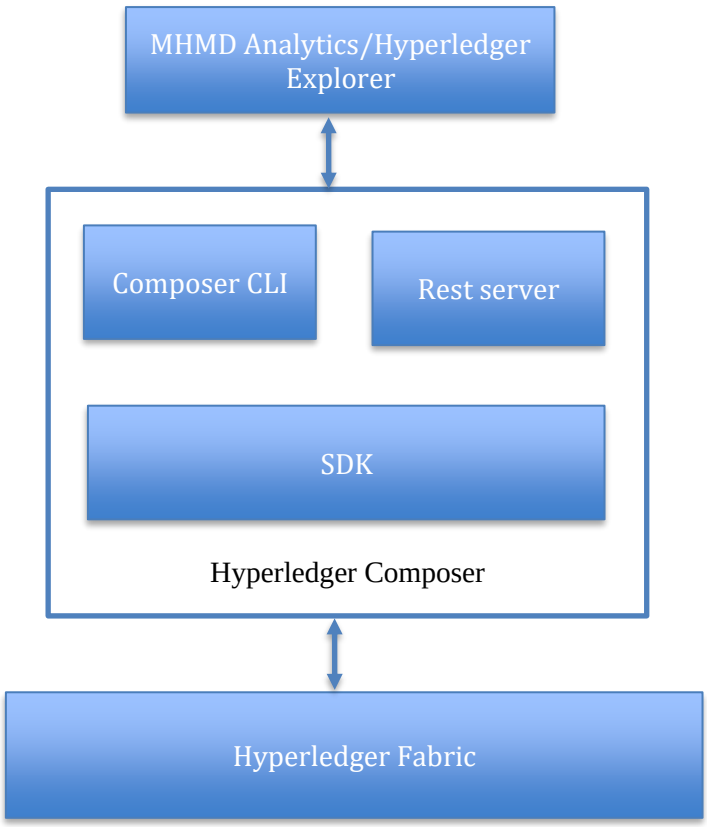



*Figure 3: Hyperledger Composer Diagram (copyright Linux Foundation – Hyperledger Project)*

## 5.2 Implementation for MHMD

The analytics module is based on Hyperledger Explorer that runs on Hyperledger Composer. The MHMD Analytics will interact with the Composer framework (Rest Server-CLI and SDK) to periodically query the ledger the number of transactions, number of blocks and number of nodes into the network.




## 5.3 MHMD Hyperledger Explorer

### 5.3.1 Transactions Query

The amount of transactions can be queried by reading the HistorianRecords of the channel. This record contains the information about the historical transactions and can query the full container and for specific transactions too. The implementation for MHMD is based on the query of the full HistorianRecord in order to get the total amount of transactions performed in the network.

The implementation of the transaction query is presented below:

```
.then(() => {
    return MHMDNetworkConnection.getHistorian();
}).then((historian) => {
    return historian.getAll();
}).then((historianRecords) => {
    console.log(prettyoutput(historianRecords));
})
```

### 5.3.2 Block Count Query

The number of blocks will give us the information about the growing rate of the chain and we can also calculate the performance of the consensus algorithm. With this we can measure the time it takes to the system to confirm a new transaction by adding it into a new block.

An implementation example of the block count query is presented below:

```
type BlockchainInfo struct {
    Height            uint64 `protobuf:"varint,1,opt,name=height" json:"height,omitempty"`
    CurrentBlockHash  []byte `protobuf:"bytes,2,opt,name=currentBlockHash,proto3"
json:"currentBlockHash,omitempty"`
    PreviousBlockHash []byte `protobuf:"bytes,3,opt,name=previousBlockHash,proto3"
json:"previousBlockHash,omitempty"`
}
```

### 5.3.3 Participants Count Query

The participants count gives us the status of the network according to the number of participants. This information can be presented as a total number of participants and/or according to the role into the network.

The implementation of the nodes count query is presented below:

```
const MHMDNetworkConnection = require('composer-client').MHMDNetworkConnection;
let MHMDNetworkConnection = new MHMDNetworkConnection();
return MHMDNetworkConnection.connect('hlfv1', 'digitalproperty-network', 'admin', 'adminpw')
    .then(() => {
        return MHMDNetworkConnection.getIdentityRegistry();
    })
    .then((identityRegistry) => {
        return identityRegistry.getAll();
    })
    .then((identities) => {
        identities.forEach((identity) => {
          console.log(`identityId = ${identity.identityId}, name = ${identity.name}, state = ${identity.state}`);
        });
        return MHMDNetworkConnection.disconnect();
    })
    .catch((error) => {
        console.error(error);
        process.exit(1);
    });
```

# 6 MHMD Analytics Mockup

The dashboard is meant to display, in an efficient and eye-catching way, every relevant data related to the blockchain. It must be appealing and yet insightful, using charts and graphical components to illustrate the data gathered by our software. It is meant to be a powerful and highly interactive interface, with many metrics to look at.

The mockup of the full dashboard is presented below:


*Figure 4: MHMD analytics Dashboard*

## 6.1 Block processing

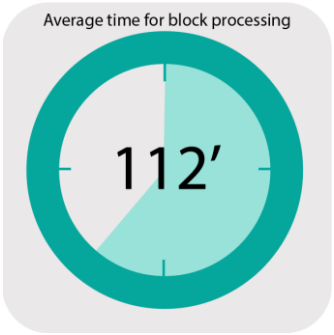



*Figure 5: Clock-like visualization for block processing time*

This component displays the average time for block processing in a clock-like visualization. It is measured in blocks per minutes.

## 6.2 Number of nodes

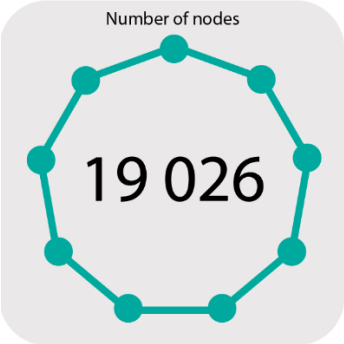



*Figure 6: Number of nodes visualization*

This component displays the total number of nodes in the system. The visualization is made with circles linked to each other to represent the nodes and the blockchain.

## 6.3 Textual space

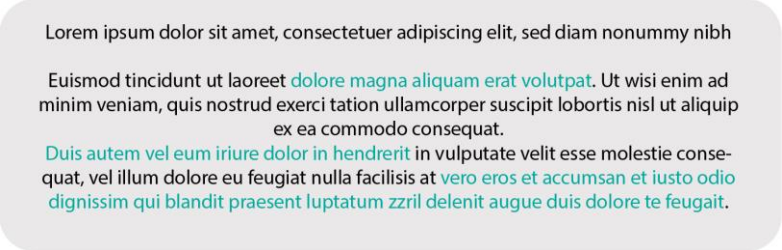



*Figure 7: Information text box*

A block for displaying information that is not related to any type of data or presenting how the blockchain works.

## 6.4 GDPR compliance


*Figure 8: GDPR information box*

The total number of protected persons in the blockchain. Also, a reminder of the GDPR norms, and how MHMD is complying with these laws. Complementary can be displayed by hovering the circles. The three main axis are:

- Compliance journey
- Right to be forgotten
- Transparency client

## 6.5 Social networks data

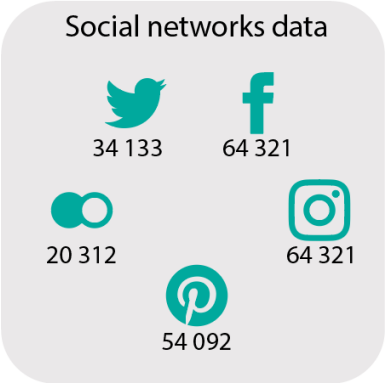



*Figure 9: Social network data box*

This block displays the data gathered by DigiMe. It could be images, text or metadata from Twitter, Facebook, Flickr, Pinterest and Instagram.

## 6.6 Blockchain data


*Figure 10: General blockchain data box*

This block is used to display data related to the blockchain. It could be specifications of the blockchain: advantages, costs, transparency, decentralization… It can also display some data like: Number of participants by types (hospitals, research labs, Digime…), number of transactions…

The nodes have different logos and reveals their information on hover, with a soft popup.

## 6.7 Repartition of studies

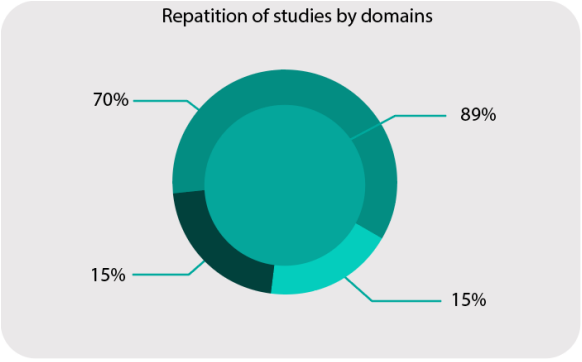



*Figure 11: Studies distribution box*

A detailed sunburst graph is used to display the repartition of studies per types: Hospitals, research labs… It is interactive and can be zoomed in or out.

## 6.8 Transactions

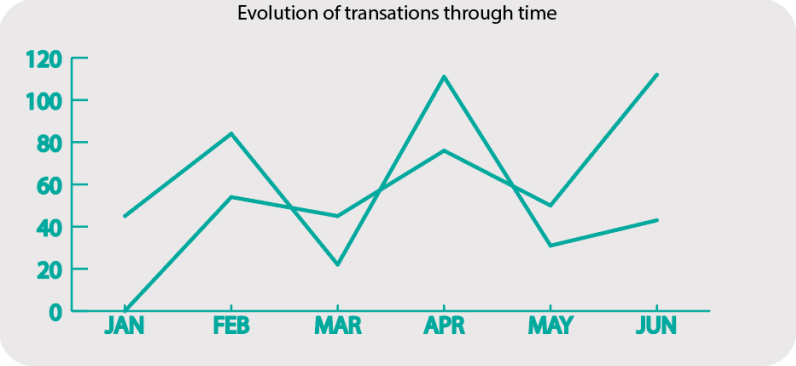



*Figure 12: Transaction chart box*

A detailed chart graph using blockchain data to display the evolution of transactions through time. It can be relative to any period of time.

## 6.9 FedEHR data

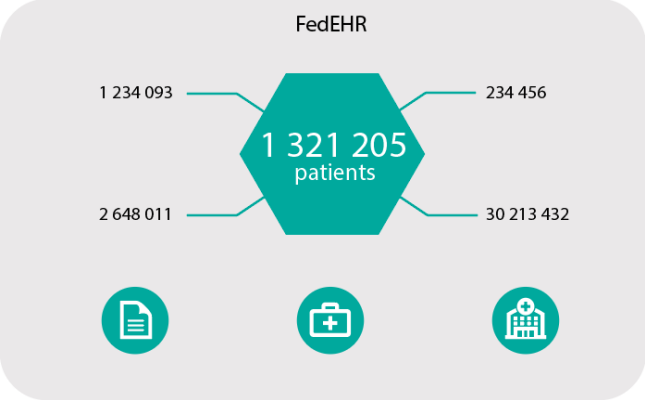



*Figure 13: FedEHR data box*

This block contains all the relevant data gathered in FedEHR like:

- Number of patients
- Number of clinical variables
- Number of medical events
- Number of hospitals

# 7 MHMD Blockchain Explorer

Some of the presented metrics have been Implemented with a tool that is called MHMD Blockchain Explorer. This tool is based on the Hyperledger Explorer [9]. As explained, the Hyperledger Explorer is a blockchain module and one of the Hyperledger projects hosted by The Linux Foundation. Designed to create a user-friendly Web application, Hyperledger Explorer can view, invoke, deploy or query blocks, transactions and associated data, network information (name, status, list of nodes), chaincodes and transaction families, as well as any other relevant information stored in the ledger. Hyperledger Explorer was initially contributed by IBM, Intel and DTCC.

Hyperledger explorer consists in a backend application that runs on the top of Hyperledger composer. The communication between the backend application and the client is through a REST API that allows us to invoke the functions implemented in the different chaincodes loaded in the peers connected to the network. Additionally, Hyperledger explorer includes pre-defined functions to show graphically information about the network status such as: blocks generated, number of transactions settled, numbers of organisations, numbers of nodes, numbers of chaincodes, among others.

This tool is intended to be deployed in all the MHMD sites and will provide the sites administrators a way to see what occurs into the blockchain.

In the following sections, we will go through the different functionalities.

## 7.1 The top banner

The web interface is composed of a top banner as shown in the following figure


*Figure 14: the top banner*

This banner is the main navigation entity of the interface. It allows to display the following pages:

- Dashboard: corresponds to the main page where is shown general information about the blockchain services.
- Blocks: page with the list of blocks generated in the network
- Transactions: page to display the list of transactions
- Chaincodes: page with the list of chaincode loaded in the peers of the system
- Channels: page with list of channels where the peers are connected
- Network: page with the list of peers connected to the channel
- MHMD: pages with the list of data item registered in MHMD and the list of studies created into the network

## 7.2 Dashboard

The main page is a dashboard which provides high level information on the MHMD blockchain network. The dashboard shows the information of the status and statistics of the MHMD service.

In the upper side of the page, we displayed a summary of the number of blocks generated, the number of transactions settled, the number of nodes connected to the network, and the chaincodes installed in the peers. In the middle side we show a pie chart to display the amount of transactions per organization. In the analytics charts, we display a dynamic graph to show the evolution of the number of blocks per hour, blocks per minutes, transactions per hour, and transactions per minutes. In the end of the page, we present the list of the recent blocks created and the status of the peers connected to the network.

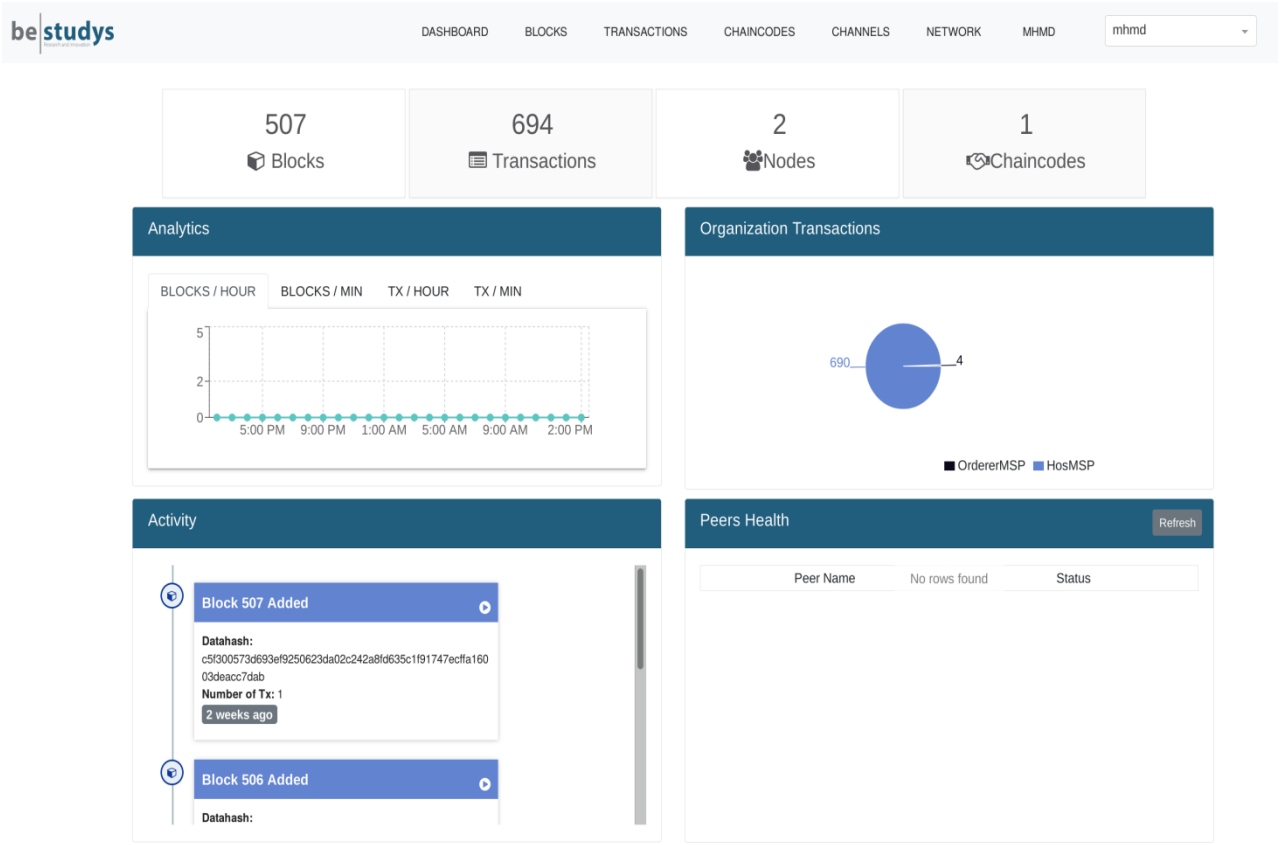
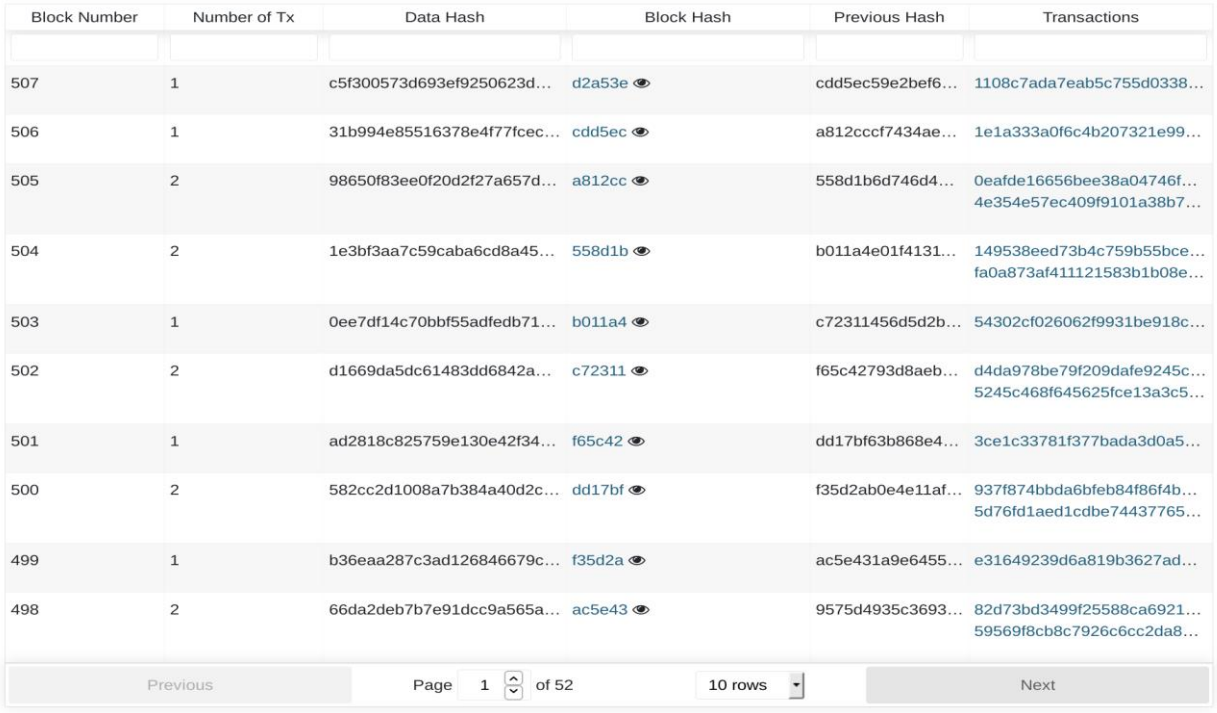



*Figure 15: Landing page*

## 7.3 Blocks

The blocks section aims at giving detailed information of the different blocks which were added into the blockchain. The block list is sorted in descending order and for each block is shows: the block number, the number of transactions included in the block, the hash value of the data, the block hash, the previous block hash and the transaction ID of the transactions included in this block.

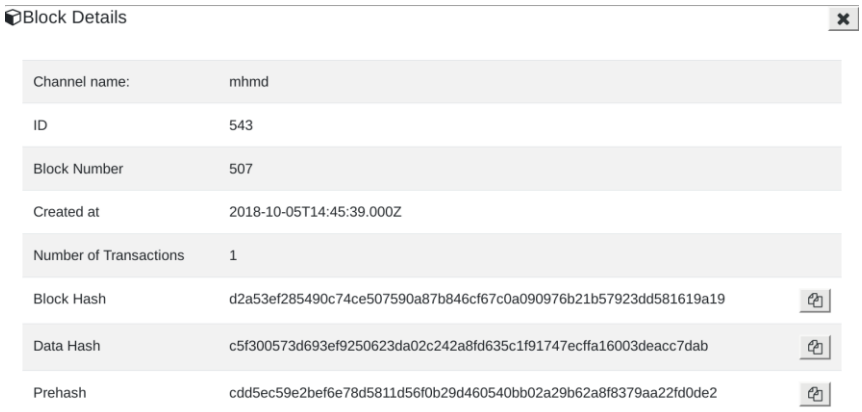


| Block Number | Number of Tx | Data Hash | Block Hash | Previous Hash | Transactions |
|---|---|---|---|---|---|
| 507 | 1 | c5f300573d693ef9250623d… | d2a53e | cdd5ec59e2bef6… | 1108c7ada7eab5c755d0338… |
| 506 | 1 | 31b994e85516378e4f77fcec… | cdd5ec | a812cccf7434ae… | 1e1a333a0f6c4b207321e99… |
| 505 | 2 | 98650f83ee0f20d2f27a657d… | a812cc | 558d1b6d746d4… | 0eafde16656bee38a04746f…<br>4e354e57ec409f9101a38b7… |
| 504 | 2 | 1e3bf3aa7c59caba6cd8a45… | 558d1b | b011a4e01f4131… | 149538eed73b4c759b55bce…<br>fa0a873af411121583b1b08e… |
| 503 | 1 | 0ee7df14c70bbf55adfedb71… | b011a4 | c72311456d5d2b… | 54302cf026062f9931be918c… |
| 502 | 2 | d1669da5dc61483dd6842a… | c72311 | f65c42793d8aeb… | d4da978be79f209dafe9245c…<br>5245c468f645625fce13a3c5… |
| 501 | 1 | ad2818c825759e130e42f34… | f65c42 | dd17bf63b868e4… | 3ce1c33781f377bada3d0a5… |
| 500 | 2 | 582cc2d1008a7b384a40d2c… | dd17bf | f35d2ab0e4e11af… | 937f874bbda6bfeb84f86f4b…<br>5d76fd1aed1cdbe74437765… |
| 499 | 1 | b36eaa287c3ad126846679c… | f35d2a | ac5e431a9e6455… | e31649239d6a819b3627ad… |
| 498 | 2 | 66da2deb7b7e91dcc9a565a… | ac5e43 | 9575d4935c3693… | 82d73bd3499f25588ca6921…<br>59569f8cb8c7926c6cc2da8… |

Previous | Page 1 of 52 | 10 rows | Next

*Figure 16: The Blocks page*

The block hash includes a link to see the details of the block. The information displayed include the channel name, block ID, block number, time stamp when was created, number of transactions, block hash, data hash and previous block hash (see figure 17).



Block Details

| | |
|---|---|
| Channel name: | mhmd |
| ID | 543 |
| Block Number | 507 |
| Created at | 2018-10-05T14:45:39.000Z |
| Number of Transactions | 1 |
| Block Hash | d2a53ef285490c74ce507590a87b846cf67c0a090976b21b57923dd581619a19 |
| Data Hash | c5f300573d693ef9250623da02c242a8fd635c1f91747ecffa16003deacc7dab |
| Prehash | cdd5ec59e2bef6e78d5811d56f0b29d460540bb02a29b62a8f8379aa22fd0de2 |

*Figure 17: block detail*

The transaction column lists the transaction ID of all the transactions included in the block. Each ID has a link to show the transaction details (see figure 18). The information displayed corresponds to the transaction ID,

the member service provider that created the transaction, the endorser that has endorsed the transaction, the chaincode used in the operation executed in the transaction, the transaction type, the time stamp, and chaincode used for the read/write operation included in the transaction. In the case of MHMD, the lscc chaincode manage the lifecycle of the chaincode running on each peer, and the mhmdcc is responsible to execute the special function developed to create studies and enforce GDPR.

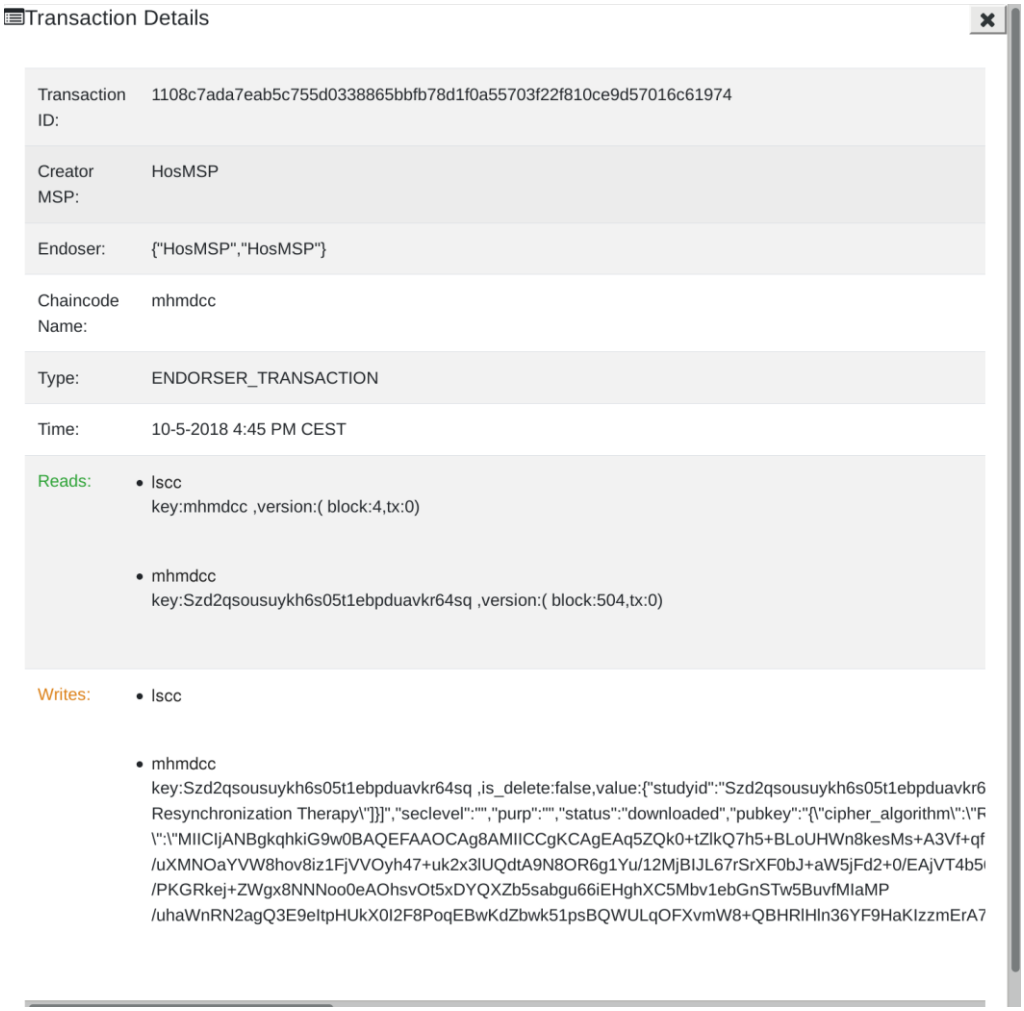



*Figure 18: block transaction details*

## 7.4 Transactions

The transactions page lists the transactions that have been settled in the blockchain. The list includes information about the organization that created it, the transaction ID, the transaction type, the chaincode invoked by the transaction, and the transaction timestamp. The transaction ID has a link to show more details about it as was presented in section 7.3 (figure 18).

| Creator | Tx Id | Type | Chaincode | Timestamp |
|---|---|---|---|---|
| HosMSP | 1108c7 | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 4:45 PM CEST |
| HosMSP | 1e1a33 | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 3:33 PM CEST |
| HosMSP | 0eafde | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 3:29 PM CEST |
| HosMSP | 4e354e | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 3:29 PM CEST |
| HosMSP | 149538 | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 12:15 PM CEST |
| HosMSP | fa0a87 | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 12:15 PM CEST |
| HosMSP | 54302c | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 11:04 AM CEST |
| HosMSP | d4da97 | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 9:26 AM CEST |
| HosMSP | 5245c4 | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 9:26 AM CEST |
| HosMSP | 3ce1c3 | ENDORSER_TRANSACTION | mhmdcc | 10-5-2018 6:21 AM CEST |

Previous Page 1 of 70 10 rows Next

*Figure 19: Transactions detail*

## 7.5 Chaincodes

This page displays the different chaincodes installed in each peer connected to the MHMD blockchain network. The chaincode list represents the list of smart contracts running in the network. In the final MHMD set up, we will have specific smart contracts for each study purpose.

| Chaincode Name | Channel Name | Path | Transaction Count | Version |
|---|---|---|---|---|
| mhmdcc | mhmd | github.com/hyperledger/fabr… | 689 | 0 |

Previous Page 1 of 1 5 rows Next

*Figure 20: Chaincodes list*

## 7.6 Channels

This page displays the different channels available into the blockchain. In the MHMD case, we will have one channel responsible to keep the tracking of the data sharing process lifecycle and to enforce the security policies established by GDPR.

| ID | Channel Name | Channel Hash | Blocks | Transactions | Timestamp |
|---|---|---|---|---|---|
| 3 | mhmd | 3ba26a8c07862e5fb1c3518669f… | 486 | 664 | 6-21-2018 12:55 PM CEST |

Previous Page 1 of 1 5 rows Next

*Figure 21: Channels list*

## 7.7 Network

The network page displays the list of peers connected to the network. The peers can be displayed by organisation, role or all togethers. The MHMD set up consider two peers per each organization.

| Peer Name | Request Url |
|---|---|
| | grpc://peer0.hos.mhmd.com:7051 |
| | grpc://poc-system-handle.notrust.almerys.gnubila.fr:7051 |

Previous Page 1 of 1 5 rows Next

*Figure 22: Network page*

## 7.8 MHMD

The MHMD page shows the status about the service offered by the network regarding the data sharing process lifecycle. Through this page we can see the list of studies created into the network and the list of items registered in the data catalogue.

In the upper side of the page, we display the status of the study:

- Defined: once the study is created by the researcher and the request was recorded in the blockchain
- Started: once the member of the network has communicated their participation in the study request
- Access granted: once the members has confirmed that they have fulfilled all the security requirements to share the data (i.e. patient consent)
- Ready: once each member has made available their data package to be shared
- Downloaded: once the data requestor has successfully downloaded the full data package

In the lower side of the page we display the status history by date and hour.


*Figure 23: Studies list*

The GDRP registry shows the list of data items that have been registered in the MHMD platform including the data source, the data item ID, the hash of the item registered, the hash of the data item, and the bitmap offset to map the individual data items with the blockchain hash proof of existence.

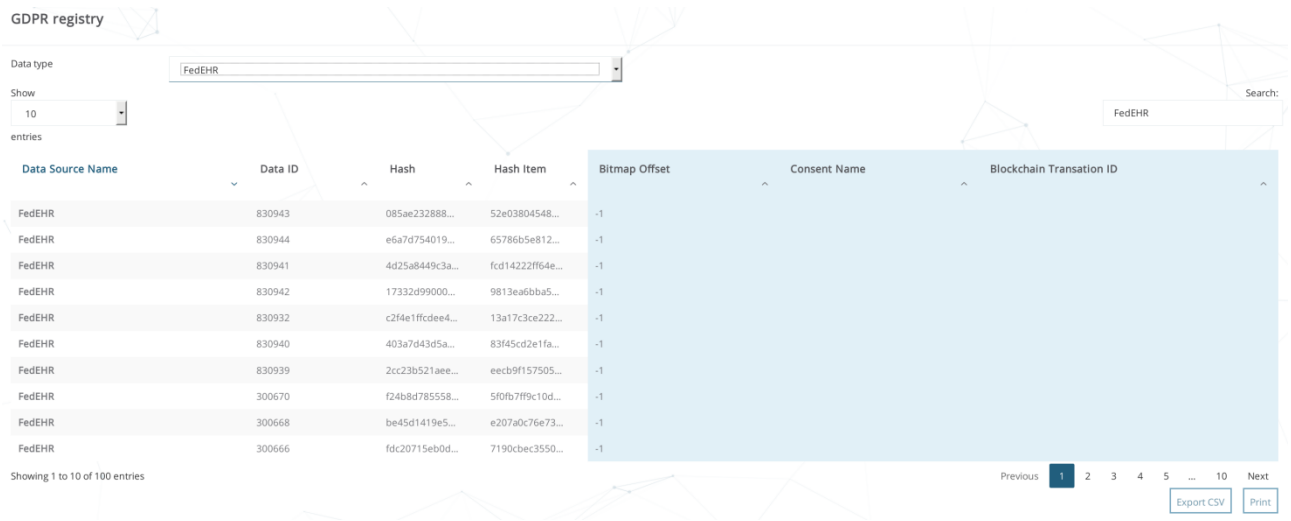



*Figure 24: Data items list*

## 7.9 Actual limitations and future works

During this last year, one of the main objectives was to update the MHMD Blockchain explorer in order to be able to connect to the blockchain which was setup to use TLS connection. The MHMD blockchain explorer was based on a quite old version of the Hyperledger Explorer tool: v0.3.6 (which was the latest version available when we started to work on it). In order to be able to connect a TLS protected blockchain, a

migration was updated to v0.3.7. This migration was not straightforward as a different customisation was done already.

This version has many stability issues which can be problematic for a production ready product. Anyway, the Hyperledger Fabric version which is currently used in the project is v1.2, so it is not possible to upgrade to the latest version of the Hyperledger Explorer tool. Some bug fixing could be done but it would be much more efficient to update to a more recent version of the Hyperledger Explorer tool. As this implies the update of the blockchain version itself, it will have a lot of impacts everywhere so its has not yet been planned.

Another limitation which is problematic is that in the GDPR registry (see previous section), the full list of data items is retrieved from the server and paginated on the Blockchain Explorer client. This is an issue as soon as the number of data items increase: it takes time to retrieve and it could also create out of memory issues. The future version should be able to do the pagination server side. This modification is not so quick to implement as it also has an impact on the filtering and the ordering functionalities. This has not yet been done as for the moment, we are far from reaching the limit but this is important to keep in mind in order to put a strong product in production.

# 8 References

[1] Hyperledger-fabric.readthedocs.io. ”Hyperledger Fabric Model”, Hyperledger-fabricdocs master documentation, 2017

[2] Hyperledger-fabric.readthedocs.io. ”Ledger”, Hyperledger-fabricdocs master documentation, 2017

[3] Hyperledger-fabric.readthedocs.io. ”Architecture Explained”, Hyperledger-fabricdocs master documentation, 2017

[4] Hyperledger-fabric.readthedocs.io. ”Chaincode Tutorials”, Hyperledger-fabricdocs master documentation, 2017

[5] Hyperledger-fabric.readthedocs.io. ”Chaincode for Developers”, Hyperledger-fabricdocs master documentation, 2017

[6] Hyperledger-fabric.readthedocs.io. ”Chaincode for Operators”, Hyperledger-fabricdocs master documentation, 2017

[7] Hyperledger-fabric.readthedocs.io. ”Chaincode Swimlanes”, Hyperledger-fabricdocs master documentation, 2017

[8] https://hyperledger.github.io. ”Welcome to Hyperledger Composer”, Hyperledger Composer documentation, 2017

[9] https://www.hyperledger.org/projects/explorer. ”Hyperledger Explorer”, documentation, 2018